

Universidad de Jaén

Escuela de Doctorado

Tesis Doctoral

NUEVAS HERRAMIENTAS PARA LA MODELIZACIÓN DE DATOS PROCEDENTES DE SENSORES

PRESENTADA POR:

Alfonso López Ruiz

DIRIGIDA POR:

Dr. Francisco Ramón Feito Higueruela
Dr. Carlos Javier Ogayar Anguita

Jaén, mayo de 2023



Universidad de Jaén

Escuela de Doctorado

PhD Dissertation

NEW TOOLS FOR MODELLING SENSOR DATA

AUTHOR:

Alfonso López Ruiz

SUPERVISORS:

Dr. Francisco Ramón Feito Higueruela
Dr. Carlos Javier Ogayar Anguita

Jaén, May 2023

I do not know how long he kept on like this, hating to go on, not daring to stop, on, on, until he was tireder than tired. It seemed like all the way to tomorrow and over it to the days beyond.

– J. R. R. Tolkien, The Hobbit

Alfonso López Ruiz: *New tools for modelling sensor data*, © May 2023

El **Dr. Francisco Ramón Feito Higueruela**, Catedrático de Universidad, y el **Dr. Carlos Javier Ogayar Anguita**, Profesor Titular de Universidad del Departamento de Informática de la Universidad de Jaén,

CERTIFICAN:

Que la presente memoria, titulada **Nuevas herramientas para la modelización de datos procedentes de sensores**, ha sido realizada bajo su dirección y autorizan su presentación y defensa para optar al grado de Doctor por la Universidad de Jaén con mención de Doctor Internacional.

Dr. Francisco Ramón Feito Higueruela
Departamento de Informática
Universidad de Jaén

Dr. Carlos Javier Ogayar Anguita
Departamento de Informática
Universidad de Jaén

Alfonso López Ruiz
Ingeniero en Informática

Jaén, mayo de 2023

ABSTRACT

The objective of this thesis is to develop a framework capable of handling multiple data sources by correcting and fusing them to monitor, predict, and optimize real-world processes. The scope is not limited to images but also covers the reconstruction of 3D point clouds integrating visible, multispectral, thermal and hyperspectral data. However, working with real-world data is also tedious as it involves multiple steps that must be performed manually, such as collecting data, marking control points or annotating points. Instead, an alternative is to generate synthetic data from realistic scenarios, hence avoiding the acquisition of prohibitive technology and efficiently constructing large datasets. In addition, models in virtual scenarios can be attached to semantic annotations and materials, among other properties. Unlike manual annotations, synthetic datasets do not introduce spurious information that could mislead the algorithms that will use them.

Remotely sensed images, albeit showing notable radiometric changes, can be fused by optimizing the correlation among them. This thesis exploits the Enhanced Correlation Coefficient image-matching algorithm to overlap visible, multispectral and thermal data. Then, multispectral and thermal data are projected into a dense RGB point cloud reconstructed with photogrammetry. By projecting and not directly reconstructing, the aim is to achieve geometrically accurate and dense point clouds from low-resolution imagery. In addition, this methodology is notably more efficient than GPU-based photogrammetry in commercial software. Radiometric data is ensured to be correct by identifying the occlusion of points as well as by minimizing the dissimilarity of aggregated data from the starting samples.

Hyperspectral data is, on the other hand, projected over 2.5D point clouds with a pipeline adapted to push-broom scanning. The hyperspectral swaths are geometrically corrected and overlapped to compose an orthomosaic. Then, it is projected over a voxelized point cloud. Due to the large volume of the resulting hypercube, it is compressed following a stack-based representation in the radiometric dimension. The real-time rendering of the compressed hypercube is enabled by iteratively constructing an image in a few frames, thus reducing the overhead of single frames.

In contrast, the generation of synthetic data is focused on LiDAR technology. The baseline of this simulation is the indexing of scenarios with a high level of detail in state-of-the-art ray-tracing data structures that help to rapidly solve ray-triangle intersections. From here, random and systematic errors are introduced, such as outliers, jittering of rays and return losses, among others. In addition, the construction of large LiDAR datasets is supported by the procedural generation of scenes that can be enriched with semantic annotations and materials. Airborne and terrestrial scans are parameterized to be fed with datasheets from commercial sensors. The airborne scans integrate several scan geometries, whereas the intensity of returns is estimated with BRDF databases that collect samples from a gonio-photometer. In addition, the simulated LiDAR can operate at different wavelengths, including bathymetry, and emulates several returns.

This thesis is concluded by showing the benefits of fused data and synthetic datasets with three case studies. The LiDAR simulation is employed to optimize scanning plans in buildings by using local searches to determine optimal scan locations while minimizing the number of required scans with the help of genetic algorithms. These metaheuristics are guided by four objective functions that evaluate the accuracy, coverage, detail, and overlapping of the LiDAR scans. Then, thermal infrared point clouds and orthorectified

maps are used to locate buried remains and reconstruct the structure of a poorly conserved archaeological site, highlighting the potential of remotely sensed data to support the preservation of cultural heritage. Finally, hyperspectral data is corrected and transformed to train a convolutional neural network in pursuit of classifying different grapevine varieties.

RESUMEN

El objetivo de esta tesis es desarrollar una metodología capaz de controlar múltiples fuentes de datos, incluyendo imágenes y nubes de puntos 3D, así como capaz de corregir y fusionar dichas fuentes para aplicarlas a la monitorización, predicción y optimización de procesos del mundo real. No obstante, trabajar con datos adquiridos mediante sensores es tedioso en muchos aspectos, entre los que se incluyen las fases de adquisición, marcado de puntos de control o la clasificación de puntos. Para evitar estas tareas, se propone generar datos sintéticos a partir de escenarios modelados de manera realista, evitando así adquirir tecnología con un elevado coste y construyendo conjuntos de datos de gran tamaño de manera muy eficiente. Además, los modelos de estos escenarios se pueden relacionar con etiquetas semánticas y materiales, entre otras propiedades. A diferencia del etiquetado manual, los conjuntos de datos sintéticos no incluyen información errónea que puede inducir a error a los algoritmos que utilicen dichos datos.

Las imágenes procedentes de teledetección, a pesar de presentar diferencias de intensidad notables, se pueden fusionar maximizando la correlación entre ellas. Esta tesis emplea el algoritmo de registro de imágenes conocido como *Enhanced Correlation Coefficient* para el registro de imágenes multiespectrales, térmicas y en el espectro visible. Después, la proyección de información multiespectral y térmica al 3D se hace transfiriendo estos datos a una nube de puntos RGB muy densa reconstruida con fotogrametría. El objetivo que se persigue utilizando una proyección en lugar de una reconstrucción directa es obtener nubes de puntos densas y con una geometría precisa a partir de imágenes de muy baja resolución. Además, esta metodología es notablemente más eficiente que la que integran algunas soluciones comerciales. La calidad de los datos radiométricos se garantiza teniendo en cuenta la oclusión, y minimizando la disilimitud entre la agregación de un conjunto de datos y dichos datos.

Por otra parte , los datos hiperespectrales se proyectaron a nubes de puntos voxelizadas (2.5D) mediante una metodología adaptada al escaneo de tipo *push broom*. En primer lugar, se corrige la geometría de las imágenes hiperespectrales y se solapan para componer un ortomosaico. Después, éste se proyecta en una nube voxelizada. Debido al gran volumen del hipercubo resultante, se comprimen los datos radiométricos mediante una representación basada en *stacks*. Finalmente, se renderiza en tiempo real el hipercubo comprimido mediante la construcción de la imagen en múltiples fotogramas, de manera iterativa.

En comparación con los datos previos, los datos sintéticos se centran en la tecnología LiDAR. La base de esta simulación es la indexación de escenarios con un alto nivel de detalle mediante estructuras de datos diseñadas para resolver rápidamente intersecciones entre rayos y triángulos. Sobre esta base, se incluyen errores sistemáticos y aleatorios, tales como puntos anómalos, ruido en la trayectoria de un pulso o pérdidas de retornos, entre otros. Además, se generan escenarios procedurales enriquecidos con datos semánticos y materiales, que posibilitan la construcción de grandes conjuntos de datos LiDAR. Los escaneos aéreos y terrestres se parametrizan para poder incluir un amplio número de sensores comerciales. Además, integran múltiples patrones de escaneo y simulan la intensidad utilizando muestras obtenidas con un goniofotómetro, publicadas en una base de datos de BRDFs. Por último, también se contemplan longitudes de onda diferentes en el sensor LiDAR, incluyendo el escaneo batimétrico, y se simulan múltiples retornos.

Esta tesis concluye mostrando los beneficios de los datos fusionados y generados de

manera sintética mediante tres casos de estudio diferentes. La simulación LiDAR se emplea para optimizar el escaneado de edificios mediante búsquedas locales, que mejora las posiciones de escaneo, y algoritmos genéticos, que minimizan el número de escaneos necesarios. Estas metaheurísticas son guiadas por cuatro funciones objetivo que evalúan precisión, cobertura, nivel de detalle y solapamiento de los escaneos propuestos. Después, nubes de puntos térmicas y mapas ortorectificados, también térmicos, se emplean en la identificación de posibles restos enterrados y la reconstrucción de la estructura de un yacimiento arqueológico. Se muestra así el potencial de la teledetección para preservar el patrimonio histórico. Finalmente, los datos hiperespectrales se corrigen y transforman para entrenar una red neuronal convolucional con el objetivo de clasificar variedades de vid.

PUBLICATIONS

The following works are already published unless indicated otherwise. These are organized into journals and peer-reviewed conferences; the first are sorted according to the JRC quartile and impact factor, while the latter are classified according to the conference rating from GII (Group of Italian Professors of Computer Engineering), GRIN (Group of Italian Professors of Computer Science), and SCIE (Spanish Computer-Science Society):

- **JCR-indexed journals**

  - **Alfonso López-Ruiz** and Juan-Manuel Jurado-Rodríguez and Carlos-Javier Ogayar Anguita and Francisco-Ramón Feito-Higueruela, 2021. An optimized approach for generating dense thermal point clouds from UAV-imagery. ISPRS Journal of Photogrammetry and Remote Sensing, volume 97, pp. 78-95. doi: 10.1016/j.isprsjprs.2021.09.022.

    Journal impact factor: 11.774. Q1 (1/50) in *Geography, Physical*, Q1 (6/202) in *Geosciences, Multidisciplinary*.

  - **Alfonso López-Ruiz**, Carlos-Javier Ogayar-Anguita, Juan-Manuel Jurado-Rodríguez & Francisco Ramón Feito, 2023. Metaheuristics for the optimization of Terrestrial LiDAR set-up. Automation in Construction, volume 146, pp. 104675. doi: 10.1016/j.autcon.2022.104675.

    Journal impact factor: 10.517. Q1 (3/68) in *Construction & Building Technology*, Q1 (1/138) in *Civil Engineering*.

  - **Alfonso López-Ruiz** and Carlos-Javier Ogayar-Anguita and Juan-Manuel Jurado Rodríguez and Francisco-Ramón Feito-Higueruela, 2022. A GPU-accelerated framework for simulating LiDAR scanning. IEEE Transactions on Geoscience and Remote Sensing, volume 60, pp. 1-18. doi: 10.1109/tgrs.2022.3165746.

    Journal impact factor: 8.125. Q1 (29/276) in *Electrical & Electronic Engineering*, Q1 (5/87) in *Geochemistry & Geophysics*, Q1 (4/34) in *Remote Sensing*, Q1 (5/28) in *Imaging Science & Photographic Technology*.

  - Juan-Manuel Jurado-Rodríguez, **Alfonso López-Ruiz**, Luís-Filipe Machado-Pádua & Joaquim-João Sousa-Moreira, 2022. Remote sensing image fusion on 3D scenarios: A review of applications for agriculture and forestry. International Journal of Applied Earth Observation and Geoinformation, volume 112, pp. 102856. doi: 10.1016/j.jag.2022.102856.

    Journal impact factor: 7.672. Q1 (5/34) in *Remote Sensing*.

  - **Alfonso López-Ruiz**, Juan-Manuel Jurado-Rodríguez, Carlos-Javier Ogayar-Anguita & Francisco-Ramón Feito-Higueruela, 2021. A framework for registering UAV-based imagery for crop-tracking in Precision Agriculture. International Journal of Applied Earth Observation and Geoinformation, volume 97, pp. 102274. doi: 10.1016/j.jag.2020.102274.

    Journal impact factor: 7.672. Q1 (5/34) in *Remote Sensing*.

  - **Alfonso López-Ruiz**, Carlos-Javier Ogayar-Anguita, Juan-Manuel Jurado-Rodríguez & Francisco-Ramón Feito-Higueruela, 2023. Efficient generation of occlusion-aware multispectral and thermographic point clouds. Computers and Electronics in Agriculture, volume 207, pp. 107712. doi: 10.1016/j.compag.2023.107712.

    Journal impact factor: 6.757. Q1 (4/59) in *Agriculture, Multidisciplinary*, Q1 (23/112) in *Computer Science, Interdisciplinary Applications*.

- **Alfonso López-Ruiz**, Juan-Manuel Jurado-Rodríguez, Juan-Roberto Jiménez-Pérez & Francisco-Ramón Feito-Higueruela, 2022. Generation of hyperspectral point clouds: Mapping, compression and rendering. Computer & Graphics, volume 106, pp. 267-276. doi: 10.1016/j.cag.2022.06.011.

  Journal impact factor: 1.821. Q3 (68/110) in *Computer Science, Software Engineering*.

► **Peer-reviewed conferences**

- **Alfonso López-Ruiz**, Juan-Manuel Jurado-Rodríguez, Carlos-Javier Ogayar-Anguita and Francisco-Ramón Feito-Higueruela, July 17-22, 2022. GPU-based Mapping of Thermal Imagery for Generating 3D Occlusion-Aware Point Clouds. IGARSS 2022 (2022 IEEE International Geoscience and Remote Sensing Symposium), Kuala-Lumpur (Malaysia). doi: 10.1109/igarss46834.2022.9884240.

  Conference rating: B-, Class 3.

- **Alfonso López-Ruiz**, Juan-Manuel Jurado-Rodríguez, Emilio-José Padrón, Carlos-Javier Ogayar-Anguita & Francisco-Ramón Feito-Higueruela, September 22–24 2021. Comparison of GPU-based Methods for Handling Point Cloud Occlusion. 2021 Spanish Computer Graphics Conference (CEIG), Málaga (Spain). doi: 10.2312/ceig.20211364.

- **Alfonso López-Ruiz**, Carlos-Javier Ogayar-Anguita & Francisco-Ramón Feito-Higueruela, September 22–24 2021. A GPU-accelerated LiDAR Sensor for Generating Labelled Datasets. 2021 Spanish Computer Graphics Conference (CEIG), Málaga (Spain). doi: 10.2312/ceig.20211360q.

- **Alfonso López-Ruiz**, Juan-Manuel Jurado-Rodríguez, Carlos-Javier Ogayar-Anguita & Francisco-Ramón Feito-Higueruela, June 26-28 2019. Multispectral registration, undistortion and tree detection for Precision Agriculture. 2019 Spanish Computer Graphics Conference (CEIG), San Sebastián (Spain). doi: 10.2312/ceig.20191209.

► **Pending publications**

- **Alfonso López-Ruiz**, Carlos-Javier Ogayar-Anguita, Rafael-Jesús Segura-Sánchez, 2023. Enhancing LiDAR point cloud generation with BRDF-based appearance modelling.

- **Alfonso López-Ruiz**, Joaquim-João Moreira-de Sousa, Carlos-Javier Ogayar-Anguita & Francisco-Ramón Feito-Higueruela, 2023. Classification of vineyard varieties using UAV hyperspectral imaging.

- Carolina Collado, Carmen Enríquez-Muñoz, **Alfonso López-Ruiz**, Carlos Enríquez & Juan-Manuel Jurado-Rodríguez, 2023. Detection of landscape features with visible and thermal imaging.

Research is what I am doing when I do not know what I am doing.
– Wernher von Braun

Acknowledgments

Con esta tesis culminan tres duros años de trabajo con demasiadas emociones encontradas. Tres años de desafíos, alegría, frustración y aprendizaje en los que muchas personas han aportado su granito de arena para que todo esto fuera posible.

En primer lugar, me gustaría expresar mi más sincera gratitud a mis directores de tesis, Francisco Feito y Carlos Ogayar, por darme la oportunidad de investigar y trabajar con ellos. Jamás hubiera pensado que podría investigar, o que me gustaría, y lo que no me imagino en este momento es no continuar haciéndolo. Muchas gracias por confiar en mí y permitirme llevar a cabo trabajos que han derivado en tantos éxitos. También me gustaría agradecer a aquellos profesores que, con su labor docente, despertaron en mí el interés por la informática gráfica, especialmente a Francisco Conde.

Por otro lado, también quiero agradecer a mis padres y mi hermano por apoyarme durante todo este camino y por todas las oportunidades que me han dado. Todas las emociones que he experimentado en estos tres años, y en especial las emociones negativas, también las han sufrido ellos.

Igualmente, quiero transmitir mi gratitud a todos los compañeros de departamento y del grupo de investigación de Informática Gráfica y Geomática que me han acompañado en esta aventura: Rafael, Juanma, Ángel Luis, Lidia, Isabel, y en especial me gustaría agradecer a Roberto por sus valiosos consejos en estos últimos meses.

Muchas gracias a todos los compañeros del A3-102, A3-103 y A3-155 por ayudarme a dejar a un lado mis problemas durante media hora de charla y café, y en especial a Dani, Alicia, Jesús, David, María, Inma, Gema, José Antonio, José Negrillo, José Luis Cárdenas y Juan Carlos. Igualmente, me gustaría agradecer a mis compañeros de laboratorio, Diego, Andressa, Álvaro, Wen y Bruno, por aguantarme cada día, y especialmente a Diego por hacer más amenas demasiadas tardes de trabajo.

Finalmente, me gustaría transmitir mi agradecimiento a Joaquim por permitirme realizar mi estancia de investigación en la Universidad de Trás-os Montes e Alto Douro, en Vila Real, donde me sentí como si llevara toda una vida.

Esta tesis ha sido financiada por el Ministerio de Ciencia, Innovación y Universidades mediante un contrato predoctoral FPU (FPU19/00100) y una ayuda de movilidad dentro de este mismo programa (EST22/00350).



Contents

Contents **xvii**

Acronyms **1**

1 Introduction **9**
1.1 Historical background and technology details . . . 13
1.2 Aims and objectives . . . 16
1.3 Organization . . . 17

2 Fundamentals of imaging sensors **19**
2.1 Optical imaging systems . . . 19
2.1.1 Scan geometry . . . 21
2.1.2 Multispectral imaging . . . 22
2.1.3 Thermography . . . 26
2.1.4 Hyperspectral imaging . . . 29
2.1.5 Active sensing . . . 31
2.1.6 Principles of physical measurement . . . 34
2.2 Data types, processing and indexing . . . 39
2.2.1 Results from Remote Sensing . . . 39
2.2.2 Data processing . . . 42
2.2.3 Data indexing . . . 50

3 Simulation, fusion and analysis of sensor data **55**
3.1 Fusion of information . . . 55
3.1.1 Motivation . . . 55
3.1.2 Image matching . . . 57
3.1.3 3D reconstructions . . . 60
3.1.4 Multispectral imaging . . . 64
3.1.5 Hyperspectral imaging . . . 65
3.2 Simulation of Remote Sensing data . . . 66
3.2.1 Motivation of synthetic LiDAR simulation . . . 67
3.2.2 Public LiDAR datasets . . . 68
3.2.3 LiDAR simulation . . . 69
3.2.4 Time efficiency . . . 70
3.3 Optimization of LiDAR scans in indoor environments . . . 70
3.3.1 Motivation . . . 70
3.3.2 Planning for Scanning . . . 71
3.3.3 Metaheuristics . . . 72
3.4 Analysis of Remote sensing data . . . 74
3.4.1 Vineyard phenotyping . . . 74
3.4.2 Inspection of archaeological remains . . . 79

4 Materials and software **85**
4.1 UAS platforms . . . 85
4.1.1 DJI Matrice 210 . . . 85
4.1.2 DJI Matrice 600 Pro . . . 85

4.1.3 DJI Phantom 4 . . . 86
4.2 Sensors . . . 86
4.2.1 Parrot Sequoia . . . 86
4.2.2 DJI Zenmuse XT2 . . . 88
4.2.3 Nano-Hyperspec . . . 93

5 Fusion of UAS-based image datasets **95**
5.1 Matching multispectral imagery . . . 95
5.2 Matching RGB and multispectral imagery . . . 97
5.3 Matching RGB and thermographic images . . . 98
5.4 Segmentation of individual trees . . . 99
5.5 Results and discussion . . . 100
5.5.1 Multispectral registration . . . 101
5.5.2 RGB and multispectral image-matching . . . 102
5.5.3 Thermal and RGB image-matching . . . 104
5.6 Conclusions and future work . . . 105

6 Generation of thermal point clouds **109**
6.1 On the generation of thermal point clouds . . . 109
6.2 Estimation of RGB point clouds . . . 110
6.3 Image-matching . . . 111
6.4 Thermal image mapping . . . 113
6.5 Occlusion . . . 114
6.5.1 Visibility test . . . 114
6.5.2 Occlusion test . . . 115
6.6 Visualization . . . 116
6.7 Aggregation of image samples . . . 118
6.8 Thermal characterization of vegetation . . . 119
6.9 Results and discussion . . . 121
6.9.1 Study area . . . 122
6.9.2 Performance of point cloud reconstruction . . . 122
6.9.3 Number of points . . . 123
6.9.4 Area and point density . . . 124
6.9.5 Absolute and normalized response time . . . 124
6.9.6 Aggregation of thermal data . . . 125
6.9.7 Analysis of thermal point cloud . . . 128
6.9.8 Visualization of the point cloud . . . 128
6.10 Conclusions and future work . . . 129

7 Generation of multispectral point clouds **131**
7.1 On the generation of point clouds augmented with multiple data sources . . . 131
7.2 Correction and fusion of image datasets . . . 132
7.3 Projection in 3D point clouds . . . 133
7.4 Occlusion . . . 133
7.4.1 OpenGL's rendering . . . 134
7.4.2 Compute shaders . . . 135
7.4.3 CUDA . . . 136

7.5 Point cloud sorting . . . 137
7.6 Alignment of heterogeneous data sources . . . 138
7.7 Results and discussion . . . 140
7.7.1 Study area . . . 141
7.7.2 Image processing . . . 141
7.7.3 Response time . . . 142
7.7.4 Point cloud size and normalized response time . . . 144
7.7.5 Overall comparison . . . 145
7.7.6 Data shuffling in GPU . . . 146
7.7.7 Visual results . . . 147
7.7.8 Binary data . . . 148
7.8 Conclusions and future work . . . 149

8 Generation of hyperspectral point clouds 151
8.1 On the reconstruction of hyperspectral point clouds . . . 151
8.2 Hyperspectral alignment and rectification . . . 152
8.3 Generation of hyperspectral orthomosaics . . . 153
8.4 Projection of hyperspectral orthomosaics . . . 155
8.5 Hyperspectral data compression . . . 155
8.5.1 Point cloud rendering . . . 157
8.6 Results and discussion . . . 158
8.6.1 Study area and data acquisition . . . 158
8.6.2 Performance analysis . . . 158
8.7 Conclusions and future work . . . 161

9 Simulation of LiDAR sensors 165
9.1 On the generation of LiDAR datasets . . . 165
9.2 Modelling of synthetic environments . . . 166
9.2.1 Forestry environments . . . 167
9.2.2 Urban environments . . . 168
9.3 Spatial indexing . . . 170
9.4 Labelling of triangle meshes . . . 170
9.5 LiDAR simulation . . . 171
9.5.1 Pulse modelling . . . 171
9.5.2 Collision solver . . . 177
9.6 Results and discussion . . . 182
9.6.1 LiDAR response time . . . 182
9.6.2 Generation of synthetic datasets . . . 184
9.6.3 Visual results . . . 185
9.7 Conclusions and future work . . . 189

10 Estimating radiometric data from synthetic LiDAR 191
10.1 On the estimation of LiDAR intensity . . . 191
10.2 Material modelling . . . 193
10.3 Analytical BRDFs . . . 194
10.4 BRDF database . . . 197

10.5 Results and discussion . . . 198
10.5.1 Analytical BRDFs . . . 198
10.5.2 BRDF database . . . 199
10.6 Conclusions and future work . . . 200

11 Optimization of LiDAR scans **205**
11.1 On the optimization of TLS scans . . . 206
11.2 Synthetic scenarios . . . 207
11.3 Solution encoding . . . 207
11.4 Metrics . . . 208
11.4.1 Level of accuracy, coverage and resolution . . . 208
11.4.2 Level of overlap . . . 210
11.4.3 Point sampling . . . 212
11.4.4 Solution initialization . . . 212
11.5 Spatial search . . . 213
11.5.1 Solution neighbourhood . . . 213
11.5.2 Greedy algorithm . . . 214
11.5.3 Simulated annealing . . . 214
11.5.4 Tabu search . . . 214
11.6 Genetic algorithm . . . 215
11.7 Interactive tools . . . 216
11.7.1 Picking of scannable building levels . . . 216
11.7.2 Pipeline design . . . 217
11.8 Results and discussion . . . 217
11.8.1 Performance of local searches . . . 218
11.8.2 Pipeline performance . . . 221
11.8.3 Level of overlap . . . 222
11.8.4 Response time . . . 223
11.8.5 Variable scan height . . . 224
11.9 Conclusions and future work . . . 225

12 Detection of archaeological remains with thermal imaging **227**
12.1 Historical background of Castle of Arenas . . . 227
12.2 On the inspection of archaeological remains . . . 228
12.3 Study area . . . 230
12.3.1 Acquisition of ground control points . . . 231
12.4 Reconstruction of RGB point cloud . . . 232
12.5 DEM and orthomosaic . . . 232
12.6 Thermal modelling . . . 232
12.7 Anomaly detection . . . 233
12.8 Experimental results and discussion . . . 234
12.9 Conclusions and future work . . . 236

13 Classification of grapevines using UAS-based hypercubes **239**
13.1 On the classification of grapevine varieties . . . 239
13.2 Spectrometer analysis . . . 240
13.3 Processing of UAS data . . . 241

13.4 Automated training . . . 243
13.4.1 Dataset . . . 243
13.4.2 Implementation . . . 244
13.4.3 Architecture and training . . . 246
13.4.4 Data sampling and regularization . . . 249
13.5 Experimentation and analysis . . . 250
13.5.1 Study area . . . 251
13.5.2 Material . . . 252
13.5.3 Impact of window size . . . 252
13.5.4 Ablation study . . . 252
13.5.5 Training time and network capacity . . . 254
13.5.6 Separability . . . 256
13.5.7 Confusion matrices . . . 256
13.5.8 Analysis of errors . . . 256
13.5.9 Training over satellite imagery . . . 257
13.6 Conclusions and future work . . . 258

14 Conclusions **271**
14.1 Summary of contributions . . . 271
14.2 Future work . . . 274

A Documentación en castellano **279**
A.1 Contexto histórico y tecnología . . . 286
A.2 Propósitos y objetivos . . . 289
A.3 Organización del documento . . . 290

B Conclusiones **293**
B.1 Resumen de las contribuciones presentadas . . . 293
B.2 Trabajo futuro . . . 296

Bibliography **299**

List of Figures

1.1 Cumbre Vieja volcano eruption observed from Landsat-8 [4]. 9
1.2 Number of manuscripts related to different RS platforms. 10
1.3 Quadcopter Matrice 300 RTK coupled with a dual RGB-thermal sensor (Zenmuse H20T). 10
1.4 Overall procedure to acquire data with remote sensing platforms and sensors. Firstly, sensors are coupled on spaceborne, airborne and terrestrial vehicles or operated by humans, either by carrying them as a backpack or in hand. Then, sensor products can be grouped into graphical and numerical results, though most of them produce both kinds of data. Finally, the sensing products are processed and interpreted to provide users with valuable analyses. 11
1.5 Point cloud with 2.4M points, reconstructed using 900 images from the Hintze Hall (Model uploaded by *Thomas Flynn* in *Sketchfab*). 13
1.6 Photograph of San Francisco after the 1906 earthquake, taken from a camera held by seven kites. 14
1.7 Three shortwave infrared bands acquired by Sentinel-2 (band 9, 935-955 µm, band 11, 1567-1658 µm, and band 12, 2114-2889 µm). 15

2.1 Mechanism involved in a) push broom and b) single-snapshot scanning geometries. 21
2.2 Image captured during a flight planned with *nadir* view direction. 22
2.3 Architecture of a simple neural network with a single layer, aimed at learning the weights of importance of every spectral band. 25
2.4 Some applications of infrared imaging [33]. From left to right, and from top to bottom: night vision, volcano monitoring, surveillance and detection of incorrect thermal insulation. 26
2.5 Measurement stand for reading temperature from radiation with a single thermometer [47]. . . . 27
2.6 Example of thermogram originated from a wet piece of paper and targetted radiation. 27
2.7 Number of manuscripts related to different IR intervals. 28
2.8 Scheme of hyperspectral images, where vegetation and ground surfaces have a different spectral signature. 30
2.9 Different types of LiDAR and scanning geometries: a) terrestrial LiDAR, b) airborne LiDAR with a zig-zag scan pattern, and c) the same mission than b) using an ellipsoidal scan pattern. 33
2.10 Parameters that intervene in the measurement of radiance with a passive sensing tool. 35
2.11 Subsurface scattering simulated over a triangle mesh. 35
2.12 Parameterization of the BRDF response. 36
2.13 Insight into micro facets of a surface. The striking energy reflects into the hemisphere formed on the basis of the normal vector. Below, the same object is rendered as a polished and rough surface. Therefore, the second present microfacets that lead to a higher energy scattering. 37
2.14 Comparison of analytical BRDFs: a) Cook-Torrance ($f_0 \leftarrow 1$, $\alpha \leftarrow 0.077$), b) Phong ($ns \leftarrow 32$) and c) Oren-Nayar ($\alpha \leftarrow 0.75$). 39
2.15 Vector data in a geographic information system within an Android application. 40
2.16 Comparison of a) raster and b) vector representation of Vila Real district in Portugal. 40
2.17 Number of manuscripts related to different sensors. 41
2.18 Transition from triangle mesh to point cloud representation, from top to bottom, with the second solely showing the triangles' vertices. 43

2.19 On the left part, colouring using a mixed representation of surrounding points and on the right side, using a sole point colour for each pixel. This image was obtained using the procedure described by Schütz and Wimmer [122]. . . . 44
2.20 Shading of a 3D model and depth obtained from the z-buffer. . . . 45
2.21 Overview of occlusion, with the left model occluding the right one from the perspective of a camera placed on the left side. . . . 46
2.22 Different points of view from terrestrial imagery, which can be later used to estimate a 3D point cloud with photogrammetry. . . . 46
2.23 Overall procedure to compute the penalty value of aggregation functions. . . . 48
2.24 Ray-tracing rendering of a 3D modelled room. . . . 51
2.25 Rendering of the BVH data structure (left side) for indexing the conference room [145] on the right side. . . . 51
2.26 Response time in ms required for constructing a BVH for triangle meshes with a wide range of polygons. . . . 52
2.27 Node clustering at different BVH levels. . . . 52
2.28 From left to right: regular grid, octree, kd-tree and BVH. Part of the image was obtained from Ogayar et al. [10]. . . . 53

3.1 Available spectra from three different devices utilized for collecting UAS-based imagery. . . . 55
3.2 Number of publications related to visible, multispectral, hyperspectral and thermographic 3D modelling, from 2000 to the current year. . . . 57
3.3 Comparison of edge-based feature descriptors frequently used in image registration algorithms. These are less likely to work well over vegetation without relevant features. *ks* refers to kernel size, the ratio is presented as *r*, and *k* is a free parameter on the Harris detector. The brightness of the last two results is increased to improve the visualization. . . . 58
3.4 Systems with multiple sensors are calibrated with the aim of projecting multiple data sources into a single coordinate system. . . . 58
3.5 Comparison of two traditional feature descriptors and matching algorithms: SIFT and ORB. In this example, both are used to match multi-view sequences of the same scene. . . . 59
3.6 Two different approaches for reconstructing 3D IR models. First, SfM is used over visible and IR datasets, whereas the second procedure estimates the rigid transformation that aligns two point clouds. . . . 61
3.7 Two different methods for reconstructing 3D point clouds. First, images are mapped into a baseline point cloud once 2D and 3D features are correlated and used to estimate pose parameters. In the second approach, features are only sought in 2D to compute the transformation matrices between RGB imagery and other data source. . . . 62
3.8 Non-metal-coated GCPs made of plastic under two different scenarios. GCPs are barely noticeable in the second scenario since the surroundings have a similar temperature. In the first scenario, they are significantly more visible to the naked human eye since the surroundings are bare soil. . . . 63
3.9 Projection of an orthomosaic over a 2.5D point cloud. . . . 63
3.10 2.5D point clouds are generated by projecting images from any data source into a baseline point cloud. . . . 64
3.11 Example of a metropolitan LiDAR point cloud in the SemanticKITTI dataset [246], composed of a sequence of scans recorded from a vehicle coupled with a Velodyne LiDAR. . . . 68
3.12 Overview of the data collection proposed by Piadyk et al. [256], where LiDAR is captured along with audio and video feed. . . . 68
3.13 Simulation of a full-waveform LiDAR traversing a tree [94]. . . . 69

3.14 Overview of different scanning geometries from the HELIOS++ simulator: a) rotating mirror, b) oscillating mirror, c) fibre-optic line scanner, d) slanted rotating mirror (Palmer scanner), and e) Risley prisms (i.e., Livox scanner). . . . 70
3.15 Example of 3D model for experimentation of P4S. The main scene has another inner room which some gaps which serve to stress the planning for scanning algorithm. It helps to show how algorithms perform in terms of LOD, LOA and LOC. . . . 72
3.16 Overview of the iterative process and data encoding in a Genetic Algorithm (GA). . . . 73
3.17 Path planning of a fixed-wing military UAS with the objective of minimizing flight altitude and fuel consumption. . . . 73
3.18 Wide angle view of a vineyard plot. . . . 74
3.19 Feature transformation of hyperspectral data from a spectrometer using LDA, Uniform Manifold Approximation and Projection (UMAP) [336], PCA and Factor Analysis (FA). . . . 77
3.20 Variation of ground and vegetation temperature over time, thus showing which is the best time of the day for collecting thermal imagery (maximum difference among both plots) [374]. . . . 80

4.1 Boustrophedon path planning. . . . 85
4.2 Quadcopter DJI Matrice 210 equipped with a thermal sensor, DJI Zenmuse XT2, and the Parrot Sequoia multispectral sensor. . . . 85
4.3 Hexacopter DJI Matrice 600 Pro. . . . 85
4.4 Quadcopter DJI Phantom 4. . . . 86
4.5 Parrot Sequoia multispectral device. . . . 86
4.6 RGB and multispectral bands as captured from a Parrot Sequoia device. . . . 87
4.7 Image with fisheye effect and the target image after the removal of such a distortion. . . . 87
4.8 Relation between a pixel from the target image, p', and another from the source image, p. . . . 87
4.9 Undetermined values are obtained when the undistortion process is carried out from the source image to the target one. . . . 88
4.10 DJI Zenmuse XT2 dual-payload sensor. . . . 88
4.11 Result of geometrical corrections regarding visible and thermal imagery. Barrel and pincushion effects are depicted through distorted grids. . . . 89
4.12 Comparison of distorted RGB image and the corrected result. . . . 89
4.13 Comparison of a) free of distortion thermal image with blank values preserving the original size, b) distorted thermal image and c) free of distortion thermal image with reduced size. . . . 90
4.14 Normalized thermographic image encoded with the inferno colour-ramp texture. . . . 93
4.15 Nano-Hyperspec sensor. . . . 93
4.16 Conversion of a) hyperspectral DNs into b) reflectance using a white and dark reference. The three grey levels are sampled from a). . . . 94
4.17 Hyperspectral orthomosaic obtained by adjusting the yaw, roll and pitch angles as well as the UAS's altitude in SpectralView™ software for every swath. . . . 94

5.1 The fusion of multispectral, thermographic and RGB imagery compose a multi-layer model which is later applied to the characterization of individual trees. . . . 95
5.2 Image ghosting effect obtained by overlapping multispectral bands using $\alpha < 1$. . . . 96
5.3 Transformation models that can be estimated using ECC. . . . 96
5.4 Hierarchical registration of multispectral imagery. . . . 96
5.5 Confusion matrices depicting the similarity of multispectral images in three different datasets. . . . 97
5.6 Image-matching of RGB and multispectral datasets. The blur size was increased for a better understanding of the figure. . . . 98

5.7 An example of image-matching between an RGB image and the Green multispectral band, with the initial images being depicted on the right side. . . . 98
5.8 Flow chart of the image-matching procedure between RGB and infrared imagery. . . . 99
5.9 Matching of RGB and thermal images using 1) an RGB area smaller than thermal images and 2) a larger RGB area. . . . 99
5.10 Transformation of multispectral images to distinguish vegetation and soil. . . . 100
5.11 Conversion of polygons into convex and hexagonal hulls. . . . 100
5.12 An overview of the study area presented in the Universal Transverse Mercator (UTM) coordinate system. . . . 101

6.1 Summary of the procedure of this work. SfM-MVS and image registration stages build isolated results and thus can be performed in parallel. Then, thermal data is linked to foreground 3D points using aggregation operators that reduce the dissimilarity from the initial samples to the aggregated value. The visualization of thermal point clouds is improved by highlighting outliers, and finally, the vegetation is identified and characterized with the observed temperature. . . . 110
6.2 An example of image-matching between RGB and thermal images. The latter is overlapped within the RGB image and depicted with transparency ($\alpha = 0.7$). The bottom image presents the final quadrilateral shape. . . . 112
6.3 Rendering of a) an octree whose nodes are spheres, optimized for searches within a radius, and b) a traditional octree whose nodes are boxes. . . . 113
6.4 Top view of a thermal point cloud captured by a *nadir* camera. Absolute temperature values are normalized and rendered with the colour ramp on the top side. The search radius was fixed to 30 m. 114
6.5 Depth buffer of a camera during a visibility test carried out with a radius of size a) 20 m and b) 30 m. 114
6.6 Triangle mesh estimated for a point cloud subset through Advancing Front algorithm. Reconstruction of vegetation is clearly erroneous in comparison with the ground. . . . 115
6.7 Occlusion test with two points represented as spherical volumes. a) Point where the ray impacts first (p_1), occluding the point p_2 (b). . . . 116
6.8 Visualization of the study area using multiple configurations. a) Visualization proposed by Javadnejad et al. [58], while b), c) and d) show our modified procedure. b) $\gamma = 0.8, k = 0.8$, c) $\gamma = 0.8, k = 0.5$, d) $\gamma = 0.5, k = 0.5$. . . . 117
6.9 Representation of a point cloud with nineteen TIR radiation samples acquired by different viewpoints from the same point. The semi-sphere represents a Lambertian radiator. . . . 118
6.10 Schematic representation of normal estimation by detecting the plane that better represents a group of points. . . . 119
6.11 Comparison of normal estimations for a small point cloud. Normal vectors are up-scaled and not normalized to improve the visualization, whereas colours are encoded as the normal vector. The proposed method and PCL obtain similar results, although ours is more rapid for denser point clouds. 120
6.12 Response time, in seconds, measured in the estimation of normals from four different methods. The results were calculated as the arithmetic mean of five different measurements for each algorithm. 120
6.13 Classification of ground points with $k = 1$ and a normal threshold of 0,7. a) First step of our methodology, with candidate points being displayed as green points. a.1) Valid candidate point ($\alpha \leq \beta$), a.2) Point discarded for further processing ($\alpha > \beta$). The angles that both normal and boundary vectors form with respect to the Y-axis are defined as α and β respectively. b) Evaluation of ray-casting for each candidate point. Note that fewer points are rendered as green at this stage. 121

6.14 An overview of the study area. On the left side, the location of the surveyed area in the region of Jaén, whereas the right side shows the polygon of the study area. Coordinates are given in WGS84 (EPSG:4326). The image at the bottom shows three point clouds used in this work, from higher to lower size. . . . 123
6.15 Intensity distribution of image samples collected during the reconstruction of a thermal point cloud, as well as the grayscale distribution obtained after using the naive-no penalty and naive-penalty (P_3) approaches. The density functions concerning other variants were omitted as they returned very similar results. . . . 126
6.16 Similarity between histograms for each configuration. For the sake of clarity, Hellinger distance was expressed as $1 - d$ to provide the same interpretation for the three metrics. Hence, a value of 1 indicates that both histograms are equal ($h_{1_i} = h_{2_i}, \forall i \in [0, n)$), while 0 denotes complete dissimilarity ($h_{1_i} = 1 - h_{2_i}, \forall i \in [0, n)$). . . . 127
6.17 Frequency diagram of thermal radiation for 3D points labelled as ground and vegetation. . . . 128
6.18 a) Original RGB point cloud, reconstructed with the original resolution, b) thermal point cloud computed using the naive approach, as well as the results obtained by c) Agisoft Metashape and 4) Pix4Dmapper. . . . 129
6.19 Close-up view of a thermal point cloud with ∼20M points. Details of thermal images are preserved and no blurring effects are observed. . . . 129

7.1 Overall procedure of the proposed method. First, the visible point cloud is reconstructed. Then, alternative data sources are projected into the previous point cloud. Finally, point clouds are aligned through the estimation of rigid transformations. The reconstruction of alternative point clouds is compared against popular commercial solutions. . . . 132
7.2 Workflow of the rendering-based methodology for a single batch of 3D points. . . . 134
7.3 Overview of the rendering-based algorithm for multiple point cloud subdivisions. . . . 135
7.4 Compute shader steps involved in rendering a point cloud. . . . 135
7.5 Overview of methodology based on compute shaders for a single batch of 3D points. . . . 136
7.6 Colouring of two randomized point clouds in [0, 1] according to Morton encoding with 30 bits. . 137
7.7 ICP registration process: a) RGB point cloud from the first dual device, b) RGB point cloud from the second dual device, and c) both point clouds aligned by minimizing their distance, with G referring to a global system. . . . 138
7.8 Overview of the surveyed region and areas. a) The region of Jaén, b) a forest and c) an olive grove. 140
7.9 Confusion matrix to depict the similarity of multispectral images in the first and second dataset (a) and b), respectively). . . . 142
7.10 Image similarity measured after performing the ECC alignment with a precision equal to 1^x. Lower values seek the most accurate alignment. The similarity is normalized in [0, 1] for improving the visualization. . . . 142
7.11 Accumulated response time for every data source (1. thermographic and 2. multispectral), study area (a. forestry and b. olive orchard) and solution, including commercial software and configurations of our method. . . . 143
7.11 Continuation of Figure 7.11 on the dimensionality of the reconstructed point clouds. . . . 144
7.12 Latency in seconds for OpenGL and CUDA approaches, using a point cloud with 271M points and 1352 viewpoints). . . . 144
7.13 Average response time and point cloud size obtained by commercial software and three versions of the proposed method. . . . 145

7.14 Comparison of performance of the dense reconstruction stage using sorted & shuffled data and globally sorted data. The first image compares the overall latency of the dense reconstruction, whereas the second image only compares the projection procedure. 147
7.15 Comparison of latency measured from the first two stages of our method in the first and subsequent launches (loading calculated data from binary files). 148

8.1 The overall approach proposed for the generation of 3D hyperspectral point clouds. The starting point cloud is first sorted and shuffled to enhance its rendering. Hyperspectral swaths are then corrected and fused to build an orthomosaic. This result is projected into the RGB point cloud, previously voxelized to obtain the visible points. The hypercube is then compressed according to a stack-based representation. 152
8.2 RGB hyperspectral swaths and the reconstructed orthomosaic using penalty-based functions (green wavelength). 154
8.3 Overview of the aggregation procedure in the GPU. 154
8.4 Point cloud rendered according to the selected aggregation operator for red (a) and green (b) wavelengths. The operator used in non-overlapped areas is irrelevant (indeed, it is the first one: max). 154
8.5 2.5D voxelization of an RGB point cloud. 155
8.6 The 70th layer of the Pavia University hypercube. 156
8.7 Stack-based representation for a hypercube with two layers. Only the first column is represented in the compact buffer. 156
8.8 Rendering of a point cloud with the Green wavelength, using the optimization described by Schütz et al. [122]. 157
8.9 Iterative rendering of a hyperspectral point cloud during several frames, using 10k points per frame. 157
8.10 Response time in ms for the GPU-based approach, stacked per aggregation stage, whereas the chart in the right side compares the performance of both CPU and GPU-based methods. 159
8.11 Performance of 2.5D voxelization for three different point clouds. Dotted lines show the results of the CPU-based approach, while solid lines represent the GPU-accelerated one. 159
8.12 Response time (s) for building a hyperspectral point cloud with 270 spectral bands. 160
8.13 Frame rate for rendering hyperspectral point clouds with an increasing target layer. Activations refer to the number of points which were rendered in every frame. 161

9.1 Depth buffer of a 3D scene, as proposed in previous LiDAR simulations. 165
9.2 Overview of a LiDAR simulator operating over 3D virtual scenes, either procedural or static, which are previously indexed to speed up spatial queries. 166
9.3 Hydraulic erosion of 200k particles over a heightfield modelled with 2D Perlin noise. 167
9.4 Workflow of the procedural modelling of a forestry area with water, low-vegetation, high-vegetation and buildings. First, the soil is eroded using hydraulic processes, and then, water and low vegetation are procedurally generated. Finally, trees and buildings are generated so that collisions among them are avoided. 167
9.5 Valleys detected on an eroded terrain. 168
9.6 Configuration of the node editor for constructing an environment such as the one depicted on the right side. 168
9.7 Procedural urban environment based on a) a residential area and b) a metropolis. 169
9.8 Detection of collisions between two triangle meshes. 169
9.9 Rendering of the BVH indexing of the very same scene in 9.11. It was constructed using a buffer radius, r, of size 100. 170

9.10 Example of hierarchical annotation, where the furniture label includes a wide range of items. . . 170
9.11 Different criteria for labelling a static scenario. First, the scene is annotated using custom labels, whereas the second classification is performed with semantic labels in the LAS 1.4 standard. . . . 171
9.12 Collimated rays from the same pulse intersecting the Bunny model. The figures on the right side show the scheme of a) collimated rays and b) diverging rays. 171
9.13 Terrestrial scan with colours encoding the point's relative height. 173
9.14 Multiple airborne scans following a) parallel, b) zigzag and c) elliptical patterns. d) illustrates a custom path defined by the user in the GUI. 175
9.15 Summary of the simulation workflow as implemented in the CPU and GPU. CPU processing is minimized during this pipeline to avoid delays from data transfers. 177
9.16 Scanning in three environments using an increasing maximum range. 179
9.17 'Time-walk' effect simulated on glass surfaces rendered according to their semantic labels. 179
9.18 Terrain-induced errors for a procedural environment. a) depicts an environment scanned without slope-based errors, whereas b) simulates such errors, thus making these points visible in the foreground. 180
9.19 a) A loss function that slightly tolerates surfaces as glossy as water, and b) another loss function that discards every return from glossy surfaces. 181
9.20 Outliers in a TLS simulation, with $h \leftarrow 0.95$. 181
9.21 ALS and TLS simulations over a procedural environment. First, the complete point cloud is rendered ($k_r \geq 0$), whereas the second image renders only the last return of each pulse. 182
9.22 Comparison of response time between the GPU and multi-core CPU approaches using a) TLS and b) ALS with 10M pulses, 10 rays per pulse and multiple returns. 183
9.23 Absolute (left) and relative (right) response time of LiDAR stages. a) The first chart row compares the performance of TLS in a procedural forest, using a different number of allowed returns (r) and rays within a pulse (n_r). Similarly to TLS, the second chart row presents the results for an ALS sensor. 184
9.24 Number of LiDAR points per semantic label in two different urban environments. The first bar reports the results of TLS scans, whereas the second bar shows the results combining both ALS and TLS. a) Distribution of 139M and 145M points for TLS and TLS + ALS scans, respectively, whereas b) shows the profile generated by 165M and 170M points. 185
9.25 Point clouds simulated over urban environments. The input triangle meshes are first depicted, and then, environments are scanned either with TLS or ALS. 186
9.26 Comparison of real and synthetic point clouds. The first two images depict the input CAD model and its corresponding LiDAR point cloud as captured from a Pandar64 sensor [407]. The third image illustrates a synthetic point cloud from our simulator, and finally, the fourth image shows the result from the Blensor plugin with a uniform vertical resolution. 187

10.1 Overview of the proposed LiDAR simulation. First, the input scenarios are attached to semantic annotations and materials. Then, the previously explained LiDAR simulator solves the scan and the resulting point cloud is stored in a standard file format. 192
10.2 Modelling of scene surfaces to estimate the returned intensity. a) Reflectance from a BRDF parameterized with the input and output vector (w_i and w_o) [95] and b) model-agnostic material specifications, including refractivity and roughness exponent. Finally, c) previous material properties are scaled according to the model textures (albedo, metallic and roughness). 193

10.3 BRDFs depicted 1) by shading a 3D model and 2) by distorting a semi-sphere with the estimated intensity. $f_r(\vec{w})$ is represented by the distance from each vertex of the distorted semi-sphere to the origin, [0, 0, 0]. From left to right: a) Lambertian, b) Oren-Nayar with α_m = 0.5, c) Minnaert with a darkening factor (k) of 1.47, whereas ρ_d is increased by a factor of A = 2.6, d) Blinn-Phong with α = 11, e) Cook-Torrance with α = 0.685 and F_0 = 0.40, and f) Ward anisotropic with α_x = 0.15 and α_y = 0.75. ρ_d = 1 for all the illustrations. 194
10.4 Intensity simulated using the analytical BRDFs in the scanning of an urban environment. 196
10.5 LiDAR intensity estimated over an urban scenario using the BRDFs collected with a goniphotometer. 197
10.6 Signatures sampled from the material database published by Dupuy and Jakob [95]. Each cell represents a different material, and the depicted signatures are different samples for randomized input and output vectors. 197
10.7 Point clouds obtained with a TLS simulation over an urban scenario. The first is coloured according to the relative height, and the second is shaded with an unique color linked to the semantic label. 198
10.8 Comparison of the intensity estimated in the aerial scanning of a procedural forestry environment. a) Objects were modelled with a BRDF similar to the expected appearance, whereas b) uses a Lambertian BRDF for every surface. 199
10.9 Comparison of intensity estimated from a) airborne and b) terrestrial scans. 199
10.10 a) Intensity from the simulation of an HDL-64E LiDAR operating at 903 nm over surfaces with realistic materials, b) idem but working at 532 nm and c) the first experiment replicated with Lambertian surfaces. 200

11.1 Two different triangle subdivisions over the same scene. a) Triangles subdivided by iterating through the edge to be split, and b) triangles subdivided with the proposed method. 207
11.2 Binary encoding of active LiDAR solutions for a genetic algorithm. 207
11.3 Average distance (μ) of every sampled point with the rest. From left to right, and from top to bottom: uniform grid distribution, uniform random sampling of u and v, random distribution, QMC random distribution, random distribution only for a part of the triangle and three points uniformly scattered. 209
11.4 F_1 and F_2 results from uniformly sampling a 3D environment. The first image depicts how F_2 behave with the proposed formula (F_2 must be minimized, whereas F_1 is intended to be maximized). The second plot does not remove the optimal average distance of collided polygons. 210
11.5 Grid occupancy in two different configurations. a) Grid with higher resolution and three overlapping circumferences, and b) sparser grid with only two overlapping circumferences. 211
11.6 Comparison of polygons sampled using the Halton sequence and the C++ built-in random uniform distribution. 212
11.7 Overview of gradients for a 2D neighbour search. a) A fixed approach generates eight vectors, whereas b) the subsampling of a circumference can be performed with a variable resolution. . . . 213
11.8 Overview of GAs on the evaluation of the best combination of spatial LiDAR setups. 215
11.9 Showcase of a node editor chaining SA, Greedy and GA nodes. 217
11.10 Three different environments authored by Autodesk Revit ®. a) A basement with 130k triangles ($20 \times 4 \times 21\mathrm{m}$), b) a school with 500K triangles ($44 \times 24 \times 32\mathrm{m}$) and c), an office building with 2.7M triangles ($16 \times 8 \times 49\mathrm{m}$). 218
11.11 Selection of the best thirty LiDAR scans with a greedy approach in the basement scene. 219
11.12 Optimization obtained using a) naive LS, b) SA and c) TS, showing both F_1 and F_2 results in a timeline as well as the final distribution. The bottom images show the locations optimized with SA and TS, organized into a regular grid and filtered by selecting only one solution per subdivision. . 219

11.13 Results obtained by combining local searches and genetic algorithms. The left images depict the final LiDAR distribution, whereas the right images show the improvement of intermediate results with regard to the proposed metric ($Y \leftarrow$ Minimization, $X \leftarrow$ Maximization). a), b) and c) refer to LS + GA, SA + GA and TS + GA combinations. 221
11.14 a) Initial overlap and results obtained after requiring an overlap of 20%, 40% and 60%. b) Rendering of LiDAR scans selected by a Greedy approach as well as those selected by the proposed method to enhance the overlap. 223
11.15 a) The average response time per scan for LS algorithms, implemented by following two approaches: naïve and optimized, and b) summation of response time from previous configurations. 224
11.16 Response time of GAs implemented as CPU (multi-core) and GPU solutions. First, the response time is shown per GA population, and finally, the response time is accumulated for each optimization. 224
11.17 a) An experimental environment designed for testing the benefits of variable height, and b, c) locations selected by a Greedy algorithm using HDL-64E and Pandar64 LiDAR sensors. The scatter plots show the values of F_1 and F_2 for each initial solution. The first one shows the results of a LiDAR with a fixed height. 225

12.1 Artistic ideation of the Castle of Arenas [424]. 227
12.2 Scheme of the structure of the Castle of Arenas, proposed by Enrique Fernández Hervás [424]. . . 228
12.3 Anomalies that could be observed in archaeological remains, according to if these are buried, visible or remains are simply variations in the ground compounds [374]. 229
12.4 Overview of the proposed methodology. The archaeological remains are monitored through land and aerial surveying. The first is simply used to help with the analysis from the latter survey, whereas the UAS results provide a better insight into details that could not be observed from land. 230
12.5 Path from two UAS flights performed with different forward directions to capture features from the fortress walls. 230
12.6 Overview of the research area. The right image shows the surveyed area, with the archaeological remains highlighted as a black polygon. The coordinates are in the UTM reference system (m), zone 30, ETRS89. 230
12.7 Histogram of radiometrically corrected thermal samples as collected for every flight. Lower temperature values may correspond to vegetation, whereas the centre is mainly linked to rock masses, either from archaeological features or not. Higher temperature is linked to anomalies and heated rock masses. 231
12.8 DEM of the fortress along with a point cloud shaded according to the relative height. 232
12.9 a) Thermal point cloud and b) fused rendering of RGB and thermal point cloud to highlight statistical anomalies. 233
12.10 Overview of the anomaly detection procedure. a) Voxelized environment, and b) neighbourhood of a voxel at a distance of r. 233
12.11 Comparison of a) anomalies detected from a 3D voxelized neighbourhood, depicting the underlying structure, and b) the enhanced structure proposed Fernández Hervás [424]. 234
12.12 Aerial view with highlighted archaeological remains, including both visible shapes and those identified with thermal imagery (dotted). 234
12.13 Fortress entrance in RGB and thermal orthomosaics. 235
12.14 Water canalization system highlighted in the RGB point cloud as well as its ground view. 235
12.15 Rendering of thermal anomalies over two possible decimated towers. 236

13.1 At the top, the average spectrometer signature of every vineyard variety. At the bottom, there is a comparison of the normalized spectrometer and UAS data. Variance is upscaled by a factor of ten for enhancing the visualization. . . . 241
13.2 Overview of the proposed network architecture, with the input being concatenated to the output of a spatial-attention layer. Then, spectral and spatial features are extracted in Part II. Both Part III and IV comprise an Inception block with frequent follow-up layers. Finally, the resulting features are transformed to obtain the predicted label. . . . 243
13.3 Overview of dataset preparation. First, a binary mask was generated using the NDVI and rows were organized into different groups to distinguish vineyard classes. Once pixels are processed as described in Section 13.3, both reflectance and labels were split into patches. The signatures on the right side show the original and transformed reflectance, including the variance per feature. . . . 244
13.4 Augmentation of a single hyperspectral patch (23×23). . . . 250
13.5 Overview of the areas surveyed with UAS hyperspectral imaging for the classification task. Two different vineyard crops are depicted according to their main variety: a) red and b) white. . . . 251
13.6 Overall accuracy obtained for patches of different sizes, from 9 to 31. . . . 253
13.7 Training time, in minutes, and number of parameters as the window size increases. . . . 253
13.8 First proposal of Inception block [433]. . . . 253
13.9 Overall accuracy and loss during training. . . . 255
13.10 Response time for training the network over a) red and b) white grapevine samples, as well as the number of parameters for every compared network, including ours. . . . 255
13.11 Clustering of samples according to the feature transformation performed by UMAP over 1) the starting hyperspectral features and 2) features extracted by the CNN prior to transferring it to the final Softmax layer. . . . 256
13.12 Confusion matrices for the classification of red and white varieties. . . . 256
13.13 Errors observed in the classification of red varieties, together with the hyperspectral signature of a few samples concerning different surfaces. . . . 257
13.14 Mean spectral signature of a) red and b) white grapevine varieties. . . . 260
13.15 False RGB colour images from the collected hypercubes depicting two different areas, focused on 1) red and 2) white grapevine varieties. . . . 261
13.16 CNN architecture proposed by Xue et al. [365] as implemented in this study to establish a comparison. 261
13.17 CNN architecture proposed by Liu et al. [358] as implemented in this study to establish a comparison. 262
13.18 CNN architecture proposed by Nezami et al. [359] as implemented in this study to establish a comparison. . . . 263
13.19 CNN architecture proposed by Moraga and Duzgun [361] as implemented in this study to establish a comparison. . . . 264
13.20 CNN architecture proposed by Chakraborty and Trehan [362] as implemented in this study to establish a comparison. . . . 265
13.20 Continuation of Figure 13.20. . . . 266
13.21 CNN architecture proposed by Roy et al. [363] as implemented in this study to establish a comparison. 267

14.1 Overview of an iterative image-matching algorithm, with higher image dimensionality providing slower but more fine-grained transformations. . . . 274
14.2 Overview of a Generative Adversarial Network composed of discriminator and generator modules. The generator is trained to produce realistic data as similar as possible to the input data, while the discriminator is trained to distinguish between real and generated data. . . . 276

A.1 Observación de la erupción del volcán Cumbre Vieja desde el satélite Landsat-8 [4]. . . . 282

A.2 Número de artículos relacionados con diferentes plataformas de teledetección. 282
A.3 Quadcóptero Matrice 300 RTK, acoplado con un dispositivo dual RGB-termográfico. 282
A.4 Procedimiento general de adquisición de datos con plataformas y sensores de teledetección. En primer lugar, los sensores se acoplan en vehículos espaciales, aéreos o terrestres, e incluso pueden ser transportados por humanos. A continuación, se diferencian dos categorías dentro de los productos obtenidos: resultados gráficos y numéricos, aunque la mayoría puede producir ambos tipos de datos. Por último, dichos productos se procesan e interpretan para proporcionar a los usuarios información útil en el control de algún proceso. 283
A.5 Nube con 2.4 millones de puntos, reconstruida utilizando 900 imágenes adquiridas en la Sala Hintze (Modelo de *Thomas Flynn* en *Sketchfab*). 286
A.6 Fotografía de San Francisco después del terremoto de 1906, capturada desde una cámara soportada por siete cometas. 287
A.7 Tres bandas en el infrarrojo cercano capturadas por la misión Sentinel-2 (banda 9, 935-955 μm, banda 11, 1567-1658 μm, y banda 12, 2114-2889 μm). 288

B.1 Registro de imagen iterativo, donde un mayor tamaño de imagen implica mayor tiempo de respuesta y mayor precisión en la estimación de la matriz de transformación. 296
B.2 Esquema de una red generativa adversaria, compuesta a su vez por dos modelos diferentes: generador y discriminador. El primero se entrena para producir datos realistas, mientras el segundo debe discernir, con la mayor eficacia posible, si los datos de entrada son reales o no. 298

List of Tables

2.1 Consumer-grade multispectral devices found in the literature and their specifications, according to the manufacturer's source. . . . 24
2.2 Consumer-grade thermal devices found in the literature and their specifications, according to the manufacturer's datasheets. . . . 29
2.3 Hyperspectral sensors found in the literature and their specifications, according to the manufacturer's datasheets. . . . 31
2.4 Most common LiDAR systems in the literature and their attributes. . . . 34

4.1 Specifications of the Parrot Sequoia multispectral device. . . . 86
4.2 Specifications of images acquired by a DJI Zenmuse XT2 device. . . . 88

5.1 Normalized CC between original multispectral images. . . . 102
5.2 Normalized correlation coefficient and response time on the matching of a random subset of multispectral images over four different tests. . . . 103
5.3 Continuation of Table 5.2. . . . 103
5.4 Normalized CC and response time on the matching of a random subset of multispectral and RGB images from the fourth dataset. . . . 104
5.5 Normalized CC and response time on the matching of a random subset of RGB and thermal images over three different tests. . . . 105

6.1 Parameters obtained from the calibration of RGB and thermal imagery, as well as from their metadata. 111
6.2 a) Percentage of pairs of RGB-thermal images that were successfully matched, and b), percentage of points which were not visible from any thermal image for two different point clouds. Edge points refer to points that would not be visible even if the complete image dataset was successfully registered. A large radius ($r \leftarrow 80\mathrm{m}$) was utilized during the search of candidate points to avoid omitting visible values. . . . 113
6.3 Preference ratio of each aggregation operator over three different penalty functions. Frequency is normalized in $[0, 1]$, and therefore the sum of each row is one. The first and second preferred choices for each configuration are highlighted in bold. . . . 119
6.4 Performance comparison of the proposed methods. The reported values are averaged over five different tests. r_1 and r_2 correspond to a search radius of 20 m and 30 m, respectively. The best results are highlighted in bold, whereas the number of mapped images shows how many TIR images were successfully projected into the point cloud. Hence, it shows the number of aligned pairs of RGB-thermal images. Third-party software also discarded some images which could not be calibrated. 124
6.5 Dissimilarity of aggregated thermal data and initial thermal samples in a scale of $[0, 1]$, where 0 indicates perfect similarity and 1 indicates maximum dissimilarity. . . . 126

7.1 Numeric results obtained from commercial solutions and different configurations of our approach in two different thermal datasets. The global response time is split into reading point cloud (Stage 1), pre-processing (Stage 2) and densification (Stage 3), i.e., generation of a dense point cloud. . . . 145
7.2 Continuation of Table 7.1 for multispectral datasets. . . . 146
7.3 Response time of methodologies using OpenGL and CUDA, including data transfers between CPU and GPU. . . . 146

7.4 Latency of the first two stages if 1) it is the first load or 2) binary data has already been stored as a result of a previous load. . . . 148
7.5 Response time of the first two stages if 1) it is the first load or 2) binary data has already been stored as a result of a previous load. Stage 1 refers to point cloud reading, whereas Stage 2 involves the image processing prior to the dense reconstruction. . . . 149
7.6 Graphic results of the dense reconstruction computed by 1) Agisoft Metashape, 2) Pix4Dmapper and 3) our method. Multispectral point clouds are rendered with the Green layer. . . . 150

9.1 Baseline configuration parameters of LiDAR simulations. These parameters are in common to ALS and TLS. . . . 172
9.2 Parameters which can be configured on the propagation of laser beams from TLS. Position-related factors are expressed in m, whereas angular parameters are expressed in radians. . . . 172
9.3 Parameters which can be configured for generating rays from an ALS. Again, angles are expressed in radians. . . . 174
9.4 Comparison of CPU and GPU response time for TLS and ALS scans. The results were obtained using static and procedural scenes with a significant number of triangles. . . . 187
9.5 Overview of outdoor LiDAR datasets with semantic annotations, regarding their average size (*Points*/*Scan*) and number of labels. . . . 188
9.6 Specifications of sensors simulated during the scanning of urban environments. . . . 188

11.1 Configuration of parameters concerning greedy, SA and TS algorithms. . . . 218
11.2 Specifications of LiDAR sensor during optimization, following the commercial device HDL-64E. . . . 220
11.3 Configuration of the evaluated genetic algorithm. . . . 222

12.1 UTM coordinates (m) of GCPs, as well as their East, North and height error in centimetres (δ_E, δ_N, δ_H). . . . 237

13.1 Layer specifications of the proposed network. Inception is simply named, but its layers were not expanded here to make this table more readable. . . . 247
13.2 Results in terms of Overall Accuracy (OA), Average Accuracy (AA) and Kappa coefficient with different methods. . . . 254
13.3 Overall results in terms of Overall Accuracy (OA), Average Accuracy (AA) and Kappa coefficient with different methods. . . . 255
13.4 Classification of hyperspectral imagery from satellite platforms in terms of Overall Accuracy (OA) and Kappa coefficient (κ). . . . 258
13.5 Summary of information collected from different grapevine varieties. For each variety, the number of field rows is shown, along with the number of laboratory samples and the number of collected leaves. Regarding UAS acquisition, samples are obtained by labelling aerial imagery. . . . 260

List of Listings

2.1 Function for finding the most similar wavelength among those available for a specific sensor. . . . 25
2.2 Minnaert BRDF implementation for a 1) reflectance visualization tool using a hemisphere and 2) a rendering application. . . . 38

4.1 Process of converting radiance as a DN to temperature in Celsius units. . . . 91

13.1 The proposed CNN network aimed at classifying different vineyard varieties. . . . 248

Acronyms

Numbers

2DCS Two-Dimensional Correlation Spectroscopy. 71

A

AA Average Accuracy. xxxvi, 244, 248–250, 253

AABB Axis-Aligned Bounding Box. 66, 107, 108, 147, 164, 170, 171, 201, 206, 207

AI Artificial Intelligence. 6, 11, 23, 43, 60, 70

ALS Airborne Laser Scanning. xxix, xxxvi, 63, 65, 159, 166, 168–171, 176–181, 183, 185, 192, 265, 267

ANN Artificial Neural Network. 72

AVHRR-3 Advanced Very High-Resolution Radiometer. 8, 281

B

BA Bundle Adjustment. 41

BIM Building Information Modelling. 65, 199–201, 230

BMMLS Backpack-mounted Mobile LiDAR Scanning. 65, 159

BPMRAE Back-Projection MRAE. 69

BRDF Bidirectional Reflectance Distribution Function. xxiii, xxix, xxx, xxxvii, 25, 30–33, 63, 145, 157, 185–195, 267, 289

BSDF Bidirectional Scatter Distribution Function. 30

BSSDF Bidirectional Scattering Surface Distribution Function. 30

BVH BBoundary Volume Hierarchy. xxiv, xxviii, 45–47, 109, 110, 116, 119, 163, 164, 171, 177, 178, 201, 207, 210

C

CAD Computer-aided Design. xxix, 40, 64, 179, 181, 200

CARS Competitive Adaptive Reweighted Sampling. 71

CBERS China–Brazil Earth Resources Satellite. 8, 281

CC Correlation Coefficient. xxxv, 91, 95, 96, 98, 99, 120

CCD Charge-coupled Devices. 14–16

CCSM Cross-Correlogram Spectral Matching. 69

CEM Constrained Energy Minimization. 70

CG Computer Graphics. 26, 31, 32, 44

CGAL Computational Geometry Algorithms Library. 43, 113

CMC Classic Monte Carlo. 206

CMOS Complementary Metal-Oxide. 14, 15, 80, 82

CNN Convolutional Neural Network. xxxii, xxxvii, 70, 72, 73, 233, 234, 236, 239, 242–245, 249–253, 255–259, 261

CNN-AE Convolutional Autoencoder. 36

CP Control Point. 147, 152

CPU Central Processing Unit. xxviii, xxix, xxxi, xxxv, xxxvi, 7, 41, 46, 113, 125, 128–134, 140, 152–155, 164, 168, 170, 171, 176–178, 181, 183, 199, 202, 211, 212, 217–219, 279

CUDA Compute Unified Device Architecture. xviii, xxvii, xxxv, 39, 41, 44, 118, 125, 128, 130, 134, 135, 137, 138, 140, 143

D

D&C Divide and Conquer. 66

DEM Digital Elevation Model. xxxi, 24, 72, 73, 87, 176, 226, 229, 235, 253, 267

DL Deep Learning. 6, 43, 58, 61–64, 68, 70, 72, 73, 160, 175, 183, 185, 187, 195, 235, 238, 279

DN Digital Number. xxv, 7, 24, 33, 34, 86, 88, 280

DSM Digital Surface Model. 40, 58, 66, 103, 201, 226, 269, 291

DSSC Dice Spectral Similarity Coefficient. 69

DTM Digital Terrain Model. 40

E

ECC Enhanced Correlation Coefficient. xxv, xxvii, 89, 90, 92, 95, 98, 99, 106, 124, 132, 135–137, 223, 268, 287, 291

ELM Empirical Line Method. 24

EM Electromagnetic. 13, 14, 21, 60

EOP External Orientation Parameters. 55

F

FA Factor Analysis. xxv, 71, 234, 236, 240, 244, 245, 252, 253

FGR Fast Global Registration. 57

FOV Field of View. 4, 63, 106, 107, 116, 166, 168, 170, 182, 192, 193, 218, 276

FPA Focal Plane Array. 51

FPS Frames Per Second. 80, 162

G

GA Genetic Algorithm. xxv, xxx, xxxi, 66, 67, 199–202, 207, 209–212, 214–219

GAN Generative Adversarial Network. 70, 269

GCP Ground Control Points. xxiv, xxxvi, 24, 54, 55, 57, 104, 105, 116, 117, 143, 152, 225, 231

GD Gradient Descent. 239

GIS Geographic Information System. 34, 230

GLSL OpenGL Shading Language. 46, 116, 143, 152, 168, 176, 183, 202

GNSS Global Navigation Satellite System. 55, 57, 152, 225

GPC Graphic Processing Cluster. 37

GPGPU General-Purpose Computing on Graphics Processing Units. 39, 44, 51, 129, 131, 143, 145, 168, 265, 266

GPS Global Positioning System. 9, 24, 27, 55, 57, 59, 61, 63, 79, 87, 235, 282, 283

GPU Graphics Processing Unit. xix, xxviii, xxix, xxxi, xxxv, xxxvi, 7, 11, 36, 37, 39, 41, 42, 44, 46, 64, 66, 68, 102, 103, 110, 113, 114, 116, 118, 123–126, 128–137, 139, 140, 143, 145, 148–155, 159, 161, 164, 165, 167, 168, 170, 171, 176–178, 181, 183, 188, 191, 194, 199, 202, 207, 209–212, 217–219, 227, 238, 243, 245, 266–268, 279, 280, 287–289, 291

GSD Ground Sampling Distance. 3, 9, 20, 110, 114, 117, 149, 154, 224, 266, 276, 288

GUI Graphical User Interface. xxix, 161, 169, 170

H

HDRF Hemispherical-Directional Reflectance Function. 25, 31
HQS High-Quality Shading. 39
HSI Hyperspectral Imaging. 23, 53, 59, 60, 70–73, 236, 237, 240, 243–246, 249, 252

I
ICA Independent Component Analysis. 70, 234
ICP Iterative Closest Point. xxvii, 57, 58, 66, 75, 104, 132
IMU Inertial Measurement Unit. 9, 24, 27, 59, 61, 63, 79, 87, 235, 282
InSAR Interferometric Synthetic Aperture Radar. 10, 283
IR Infrared Thermal Imaging. xxiii, xxiv, 20–22, 51–58, 75, 82, 98

J
JPEG Joint Photographic Experts Group. 34, 82, 142

K
KJSSC Kumar–Johnson Spectral Similarity Coefficient. 69
KNN k-Nearest Neighbours. 40, 55, 57, 58, 71, 113

L
LAI Leaf Area Index. 7, 280
LAS LASer. xxix, 37, 164, 165
LDA Least Discriminant Analysis. xxv, 70–72, 234, 236
LiDAR Light Detection and Ranging. ix, xvii, xix, xxiii, xxiv, xxviii–xxxi, xxxv, xxxvi, 4, 6, 10, 11, 13, 26–28, 37, 44, 52, 54, 57–67, 72, 74, 103, 114, 143, 145, 146, 157, 159, 160, 162, 164–168, 171–183, 185–189, 191–195, 197, 199–201, 204, 205, 207, 209, 211–219, 231, 265–269, 277, 279, 284, 285, 287, 289, 291
LMM Linear Mixture Models. 70
LOA Level of Accuracy. xxv, 66, 199, 202, 204, 214, 216, 219
LOC Level of Coverage. xxv, 66, 199, 202, 214, 216, 219
LOD Level of Detail. xxv, 3, 38, 39, 45, 47, 61, 66, 94, 99, 160, 164, 166, 199, 201, 202, 207, 216, 226, 231, 268, 276
LOO Level of Overlap. 66, 199, 202, 216
LPC Layered Point Clouds. 38
LS Local Search. xxx, xxxi, 199, 208, 211–218
LSM Least-Square Method. 24
LSTM Long Short-Term Memory. 73
LW Long-wave. 22
LWIR Long-wave Infrared. 22, 82

M
MAE Mean Absolute Error. 119, 120
MESMA Multiple End-member Spectral Mixture Analysis. 70
ML Machine Learning. 72
MLP Multilayer Perceptron. 72, 233
MLS Mobile Laser Scanner. 65
MLSAC Maximum Likelihood Sample Consensus. 54
MMS Mobile Mapping Systems. 159
MRAE Mean Relative Absolute Error. 69

MSE Mean Square Error. 69
MTMF Mixture-Tuned Matching Filtering. 70
MVS Multi-View Stereo. xxvi, 41, 103, 104, 108, 109, 116, 123
MWIR Mid-wave Infrared. 22

N
NBV Next Best View. 65, 201
NDVI Normalized difference vegetation index. xxxii, 17, 19, 93, 237, 238, 243
NIR Near-Infrared. 17, 19, 29, 58, 69, 73, 80, 89, 91, 94, 96, 97, 135
NNMF Nonnegative Matrix Factorization. 70, 234, 236, 245

O
OA Overall Accuracy. xxxvi, 73, 244, 246, 248–253
OLI-2 Operational Land Imager. 8, 281
OOBB Object-Oriented Bounding Box. 66, 109, 200
OpenCV Open Source Computer Vision Library. 95
OpenGL Open Graphics Library. xviii, xxvii, xxxv, 31, 36–39, 44, 64, 116, 124, 125, 128, 129, 134, 137, 138, 140, 151, 152, 162, 168, 176, 177, 188
OpenMP Open Multi-Processing. 134, 152, 176, 211
ORB Oriented FAST and rotated BRIEF. xxiv, 52–54, 90, 147
OS Operative System. 44, 95, 116, 134, 152, 211, 245

P
P4S Planning for Scanning. xxv, 65–67, 199–201, 210, 211, 219
PA Precision Agriculture. 7, 17, 68
PBR Physically-based Rendering. 31, 188
PCA Principal Component Analysis. xxv, 43, 69–73, 113, 226, 234, 236, 245
PCC Point Cloud Compression. 36
PCL Point Cloud Library. xxvi, 43, 113, 114
PLS Partially Least Square. 71, 72
PLS-DA Partial Least-Square Discriminant Analysis. 71
PLY Polygon File Format. 142
PnP Perspective n Point. 56
PSNR Peak Signal-to-Noise Ratio. 69
PSO Particle Swarm Optimization. 201
PV Precision Viticulture. 68

Q
QMC Quasi-Monte Carlo. xxx, 171, 202, 203, 206

R
RANSAC Random Sample Consensus. 52–55, 57
RBF Radial Basis Function. 72
REG Red-Edge. 17, 19, 80, 89, 91, 96, 97, 135
ReLU Rectified Linear Unit. 239–241
RF Random Forest. 72
RGB Red, Green, Blue. xviii, xxiv–xxviii, xxxi, xxxii, xxxv, 11, 14, 15, 34, 50, 52, 54–60, 72–74, 80–83, 89, 91–94, 96, 98, 99, 102–111, 116–118, 122–124, 126, 127, 130–132, 134, 138, 141–143, 145–149, 152, 155, 180, 186, 195, 223, 226, 227, 229, 231, 251–253, 255, 265, 266, 287, 288, 291

RGB-D RGB-Depth. 58, 59
RIFT Radiance Invariant Feature Transform. 52
RJPEG Radiometric Joint Photographic Experts Group. 82, 85
RMSE Root Mean Square Error. 69, 119, 120, 225
RS Remote Sensing. xxiii, 3, 4, 6–8, 11, 13, 28, 33, 34, 37, 44, 125, 126
RTK Real-Time Kinematic. xxxiii, 9, 225, 276

S
SA Simulated Annealing. xxx, xxxi, xxxvi, 66, 67, 201, 208, 211–216, 247
SAH Surface Area Heuristic. 45
SAM Spectral Angle Mapper. 54, 69, 70
SBR Stack-based Representation. 37
SCA Spectral Correlation Angle. 69
SCM Spectral Correlation Matching. 69
SFC Space Filling Curve. 39
SfM Structure from Motion. xxiv, xxvi, 41, 54, 55, 59, 74, 75, 83, 103, 104, 108, 109, 116, 117, 123, 127, 149, 226, 227
SGA Spectral Gradient Angle. 69
SID Spectral Information Divergence. 69, 70
SIFT Scale-Invariant Feature Transform. xxiv, 51, 53, 54, 56, 90
SNR Signal-to-Noise Ratio. 14, 16
SNV Standard Normal Variate. 71, 234
SPA Successive Projection Algorithm. 71
SSBO Shader Storage Buffer Object. 129, 137, 148, 150, 162, 168, 177
SURF Speeded-Up Robust Features. 51, 53, 54, 90
SVD Singular Value Decomposition. 43, 236, 245
SVM Support Vector Machine. 72, 233
SWIR Short-wave Infrared. 22, 60

T
TIFF Tag Image File Format. 33, 82, 86, 142
TIR Thermal Infrared. xxvi, xxxv, 105–107, 109, 110, 112, 118, 123, 124
TIROS Television and Infrared Observation Satellite. 8, 281
TIRS-2 Thermal Infrared Sensor 2. 8, 281
TLS Terrestrial Laser Scanning. xxix, xxx, xxxvi, 63–66, 159, 166–168, 170, 171, 175–181, 185, 192, 193, 195, 199–202, 205, 220, 230, 265, 268
ToF Time of Flight. 26, 171, 177, 183, 186, 266
TS Tabu Search. xxx, xxxi, xxxvi, 208, 211–216

U
UAS Unmanned Aerial System. xxiv, xxv, xxxi, xxxii, xxxvi, 4, 6, 9–11, 20, 22–25, 31, 40, 49–51, 54, 58–60, 67, 68, 73, 74, 79, 80, 88, 90, 99, 103, 125, 145, 146, 152, 224, 226, 230, 233–235, 237, 245, 246, 251–254, 267, 269, 276, 277, 282, 283
UHD Ultra High-Definition. 80
UMAP Uniform Manifold Approximation and Projection. xxv, xxxii, 71, 234, 250
UTM Universal Transverse Mercator. xxvi, xxxi, xxxvi, 95, 105, 147, 149, 224, 225, 231
UV Ultraviolet. 26

V

VR Virtual Reality. 7, 280

VRAM Video Random Access Memory. 129–132, 152

VSLAM Visual Simultaneous Localization and Mapping. 41

Part I

Introduction and overview

This section introduces a shallow background of Remote Sensing as well as the aim and objectives of this doctoral dissertation. Then, the state-of-the-art on the fusion and simulation of Remote Sensing data is presented.

Introduction 1

1.1 Historical background and technology details . 13
1.2 Aims and objectives . . . 16
1.3 Organization 17

Sensing real-world processes help to monitor, predict and optimize the activities that depend on them and therefore act according to the analyzed data. Numerous devices are aimed at this task, which can be at least classified according to the sensing distance, the wavelength interval captured by the device detectors and the data acquisition mechanism, either passive or active. The latter two classifications are based on how a device works as provided by the manufacturer. However, choosing a sensing distance is up to researchers and consumers, depending on their objectives, area of expertise and budget, as well as on the dimensions and environmental conditions of the study area, among other factors. Despite contact-based sensors being implemented nowadays to measure variables such as humidity, temperature or rainfall amount, these require a high degree of maintenance and are more prone to deteriorate under adverse environmental conditions [1, 2]. Instead of covering large areas, each one of these sensors communicates measurements of very specific locations within and outside a network. The arranged layout, logically and physically, must be carefully designed for intra- and extra-communication involving nodes in different topological levels. Therefore, these technologies have a worse spatial awareness and are more appropriate to cope with real-time, rapid and constant monitoring.

Given the cited drawbacks, **Remote sensing** (RS) techniques help to mitigate them by acquiring data from remote platforms. RS refers to the science and art of acquiring information about objects and phenomena without being in contact with them [3]. Similarly to the human eye, data is acquired from the environment in the form of impulses from light stimuli corresponding to some wavelength intervals. At least two components are involved in the reading process: the sensors and the platforms where they are mounted. The first will be the matter of discussion throughout this dissertation, whereas platforms were initially narrowed to satellites.

Remote sensing techniques overcome the spatial limitations of devices that measure data in contact with the target surface and phenomena.

Satellite imaging is better suited for the monitoring of changes over large areas using time series spanned over months and years. These data help to understand the dynamics of human and nature interaction as well as the impact of natural phenomena (Figure 1.1). Some applications of the large collections of available satellite data are the analysis of land use, deforestation, land changes and urban settlements [5]. Nevertheless, the spatial resolution and revisit period of non-commercial satellite programmes harden their applicability to fine-grained monitoring tasks. The level of detail (LOD) of these tasks can refer to **spatial resolution**, **temporal resolution** or both, which are the main **limitations** of satellite imaging besides their cost. In spite of the described drawbacks, the use of satellite imagery is on the rise due to the steady reduction of Ground Sampling Distance (GSD) and low periods for revisiting the same points.

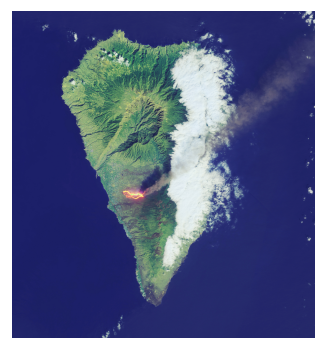


Figure 1.1: Cumbre Vieja volcano eruption observed from Landsat-8 [4].

Figure 1.2 depicts the amount of research devoted to each remote sensing

platform. The Scopus searches were the following: $(p_1 \vee p_2 ... \vee p_n) \wedge (remote \wedge sensing)$, with p_i being one of the platforms depicted in the legend.

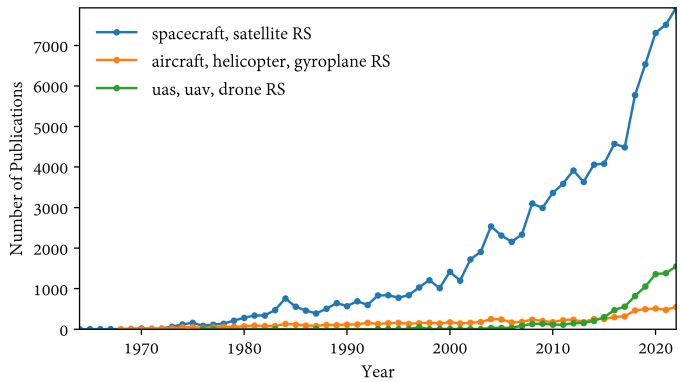


Figure 1.2: Number of manuscripts related to different RS platforms.



Figure 1.3: Quadcopter Matrice 300 RTK coupled with a dual RGB-thermal sensor (Zenmuse H20T).

The digitization of assets, processes and systems is greatly helped by reconstructions from sensor data. These allow connecting virtual and physical replicas through a data stream, thus shaping the fundamentals of **Digital twins**.

However, it was until recently that satellite resolution was restricted by governments. These restrictions as well as the described limitations led to the use of alternative platforms. Besides satellites, **airborne** (fixed-wing and helicopters), **Unmanned Aerial Systems** (UAS) (Figure 1.3) and other **land platforms** (mobile and static proximal-sensing) are the most frequent [3]. The choice of any of them always involves trade-offs concerning manoeuvrability, land coverage, repeat coverage, spatial resolution, spatial accuracy, cost or field of view (FOV) [6]. However, **UAS** have been gaining interest in the last decade as they transitioned from military tools in the early 2000s to easy-to-deploy, small and low-cost systems, thus widening their applicability to civilian activities and research. These platforms range from hand-sized to large aircraft that can be either controlled by human intervention or be partially, even fully, autonomous. Figure 1.3 shows a standard UAS designed by DJI which can carry up to 2.7 kg, with a size of $810 \times 660 \times 430$ mm.

Therefore, UASs represent a cost-efficient tool for acquiring high-resolution data. The most common sensors mounted on UAS are cameras and LiDAR (Light Detection and Ranging) sensors, with the first comprising a large number of imaging devices. Similarly, these groups also lead to the distinction between passive and active sensors. While the second has both transmitter and receiver components, the first is only aimed at capturing incoming radiance from the scene surfaces. Also, the observations of imaging devices generate 2D images, whereas the immediate results of active sensing are 3D point clouds. Accordingly, the technology involved in **active sensing** is much more **prohibitive** than imagers, which have been widespread for the last decades with consumer-grade devices. Still, both kinds of sensors can be applied to the collection of complementary resources. However, the acquisition of multiple high-resolution datasets poses several advantages and drawbacks.

Firstly, observations from multiple sensors can be interpreted as different features that complete a **knowledge-based system**, with each one

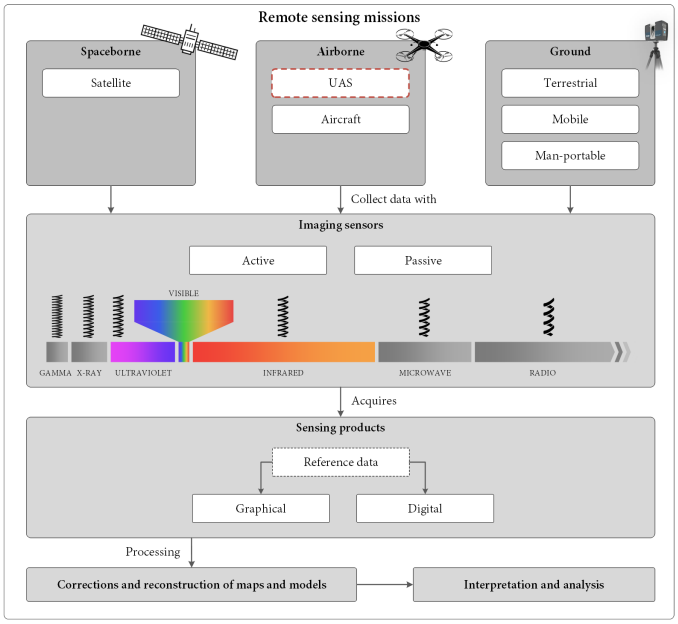


Figure 1.4: Overall procedure to acquire data with remote sensing platforms and sensors. Firstly, sensors are coupled on spaceborne, airborne and terrestrial vehicles or operated by humans, either by carrying them as a backpack or in hand. Then, sensor products can be grouped into graphical and numerical results, though most of them produce both kinds of data. Finally, the sensing products are processed and interpreted to provide users with valuable analyses.

providing information within a wavelength range. Furthermore, most of the algorithms aimed at analyzing and drawing conclusions from sensor data benefit from the use of complementary features. The contribution of each one in the extracted conclusions can be adjusted through weights that determine how important are these for the analysis. Unless a large number of features are used, which can lead to the curse of dimensionality problem, it is safe to say that the greater number of features, the better. On the other hand, **high-resolution data** help to generate **precise models that are easier to analyze and visualize by human operators**. Denser data also involves denser geometry and reconstructions, thus omitting fewer details of target surfaces and easing the construction of digital models that emulate the Earth's scenarios and processes.

Different layers of information are ideally noise-free and represented under the same coordinate system. In reality, navigation sensors present small positioning errors that harden the fusion of several layers of information. Despite being small, these are reported as shifts, rotation and scale variations among different sensor data. Furthermore, environmental conditions such as atmospheric composition, wind, temperature or solar radiation can vary from one flight to another and so do the acquired data, even under similar flight plans. These changes not only affect observations performed in a short period of time with different sensors but also time series acquired with previously used tools. Although tedious, variations concerning positioning and orientation errors can be diminished by including reference points measured with sensors of higher precision in easily recognizable image features. Other shortcomings are the noise induced by faulty device detectors, especially for active sensors, unwanted atmospheric particles and surfaces whose bouncing behaviour leads to unreal geometry. Therefore, a

system capable of accurately fusing the outcome of every sensor coupled on a UAS is necessary to facilitate the later processing stages and provide reliable conclusions.

Besides the information that can be directly extracted from remotely sensed data, other kinds of information may be required for **computer vision** tasks. For instance, it would be of great help to distinguish materials and instances in pixels and 3D points. However, these data cannot be inferred from nature without previous knowledge or trained models. Accordingly, one of the main challenges is the **labelling of large datasets**, which must be performed by human operators [7, 8] or unsupervised algorithms aimed at transforming and extracting relevant features. None of these methods is perfect and thus can result in spurious labelling data that can mislead the learning algorithms. Hence, the task of building large datasets for training computer vision algorithms is very time-consuming. It comprises acquiring, processing and extracting features with **considerable computational and human resources**. All these time-consuming stages can be avoided with the existence of publicly available datasets for RS, motivated by Artificial Intelligence (AI) algorithms. These cover a wide variety of sensors and spectral wavelengths, but the attached data present the same drawbacks as previously explained. Moreover, these datasets may not comprise areas of interest for case studies that fall out of state-of-the-art trends.

Given the drawbacks of using real datasets, from which cost and time consumption stand out, **synthetic information** generated by **virtual sensors** could be used as a partial or complete replacement. The main challenge lies in emulating the sensor mechanism. However, the process is much less time-consuming and enables adding any kind of knowledge to the sensing results. These are obtained from digital models with no uncertainty, in contrast to what occurs in nature, and they can be enriched with features such as semantic annotations which are later transferred to synthetic datasets. It neither requires acquiring prohibitive sensors such as LiDAR, and therefore is also much more cost-efficient. The most frequently simulated sensors are Radar and LiDAR systems, but recent advances in Deep Learning (DL) have made it possible to generate synthetic images as well.

Sensing products are typically **transformed** into other **data representations** that are not provided by the sensor. Accordingly, individual images are not sufficient to interpret the scenario in some contexts. A complete vision of it can be provided by joining images, thus resulting in 2D maps and 3D points calculated by estimating the camera pose and finding features in common among several images. During these transformations, estimated radiometric and geometric properties may be affected by precision loss in the interpolation process. Therefore, it is important to compare the results with other reliable data sources. For instance, LiDAR is known to produce highly accurate point clouds, which can be used as a reference to evaluate the quality of point clouds reconstructed from imagery. Similarly, images can serve as reference data for measuring radiometric changes.

On the other hand, **large volumes of data are also harder to operate in terms of computing, storage and visualization**. Thousands of images or millions of points can be hardly operated in commodity hardware due to the required storage and computing capabilities. However, current trends in informatics have favoured the proliferation of personal and professional computers with large storage capacity, more efficient access to data and great multi-threading capabilities. The latter can be performed over the Central Processing Unit (CPU), whose trend is to increase the number of cores, and the Graphic Processing Unit (GPU), composed of millions of units that can massively solve small tasks and cooperate among small thread groups. Chapter 2 delves into the most common data formats in RS; at this moment, it is enough to think of data as large volumes of digital numbers (DN). Despite large storage being inexpensive nowadays, challenges such as the partial retrieval of information remain, especially for interactive applications with high data throughput [9, 10].

Another drawback in real-time applications is the **visualization of these large volumes of data**. Traditional rendering pipelines are built as a set of static stages, where the model geometry and topology are iteratively transformed into pixels. The colours of these come from the shading of one or multiple light sources over the textures of a surface, using a discretization of the formulae that describe light interaction. Unlike synthetic models designed by human operators, the shading of sensor data depends on the recorded wavelengths and is not calculated. Thus, alternative pipelines are needed to visualize these data more efficiently. According to the GPU architecture, the data can be sorted and organized in data structures in pursuit of workload balancing and geometry simplifications that help to trick user perception. These requirements are even stricter for Virtual Reality (VR) devices that render every scenario at least once for each eye, even more whether other rendering techniques are included, e.g., shadow mapping.



Figure 1.5: Point cloud with 2.4M points, reconstructed using 900 images from the Hintze Hall (Model uploaded by *Thomas Flynn* in *Sketchfab*).

Previous methods are simply a procedure that allows extracting text-based or visual clues that help in the recognition of weaknesses and failures in processes. Accordingly, the **last step** is to **classify, segment and recognize features** from data to optimize future operations. The main contributions of RS techniques in Precision Agriculture (PA) are yield estimation, crop-type inventory, measuring of water content and Leaf Area Index (LAI), control of diseases and insects as well as moisture, changes, growth, stress and drought monitoring, among other risk factors such as snow or fire [11]. These monitoring tasks are intended to maximize profitability and minimize waste and pollution.

This dissertation has Precision Agriculture (PA) as its main research field and therefore, the data analysis has been narrowed to 1) segment crops into ground and vegetation, and 2) phenotyping of a large number of plant varieties in a non-destructive way.

1.1 Historical background and technology details

The term Remote sensing was first coined in the 1960s by Evelyn Pruitt during her work at the US Office of Naval Research to refer to satellite and aircraft instrumentation that measure reflected and emitted radiation.

However, topographic mapping was first suggested in 1849 and attempted in 1858 by F. Tournachon from a captive balloon over France. From here, the camera size has been reduced without the need of carrying a darkroom. Other airborne platforms were explored, either as a result of war needs or simply innovation, ranging from kite-camera systems (1906; see Figure 1.6) to pigeons (1909) and aircraft (1908) [12].



Figure 1.6: Photograph of San Francisco after the 1906 earthquake, taken from a camera held by seven kites.

Satellite imaging, as known today, was first possible by the ideation of Konstantín Tsiolkovsky about rocketry to explore space, published under the title *Exploring Space using jet propulsion devices*. This idea led to the first successful launch of the Sputnik satellite in 1957 and the development of Earth-orbiting satellites aimed at atmospherical monitoring. The Television and Infrared Observation Satellite (TIROS) was made operational in 1960 and integrated a small infrared system and a narrow-angle camera capturing data in the visible wavelengths. In 1978, a breakthrough change in satellites and coupled sensors came with TIROS-N (N for new). This spacecraft integrated a very high-resolution radiometer with 1 km footprint and four channels (visible, near-infrared, midrange-infrared and thermal infrared). Nowadays, an advanced version of these satellites is still operative under the name of AVHRR-3 (Advanced Very High-Resolution Radiometer) with a similar footprint, though it captures six channels instead of four [13].

An extensive repository of recent satellite missions can be found in the Earth Observation Portal, maintained by the European Space Agency [14].

From currently operative satellite programs, Landsat provides the largest collection of continuously acquired RS data. It was first tested as part of NASA's NIMBUS program and there are currently two active satellites, Landsat-8 and Landsat-9, while the other seven have been terminated or planned to be decommissioned (Landsat-7). The payload of Landsat-9 consists of two instruments: Operational Land Imager (OLI-2) and Thermal Infrared Sensor (TIRS-2) [15–17]. A total of 11 bands and 740 scenes are collected every day, including red, blue, green, near-infrared, shortwave infrared, thermal, panchromatic, coastal and cirrus bands, ranging from a resolution of 15 m to 100 m. Other notable satellite programs are the China–Brazil Earth Resources Satellite (CBERS) and the Copernicus programme financed by the European Commission. The CBERS-4A satellite is equipped with three imaging tools capturing five different spectrum intervals with a spatial resolution of 2-55 m [18]. On the other hand, the Sentinel-2 mission acquires 13 bands in the visible, shortwave and near-infrared spectrum (see Figure 1.7) with a spatial resolution ranging from 10 m to 60 m [19]. Its cycle to revisit the same Earth's location is ten days, in comparison with the 16 days needed by Landsat-9 or 31 days from CBERS-4A.

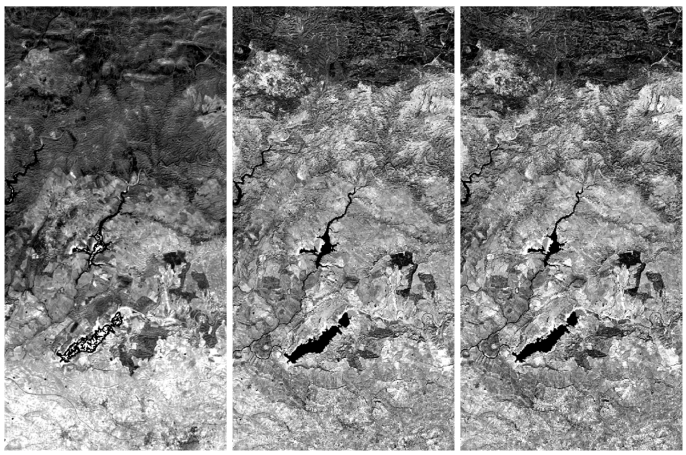


Figure 1.7: Three shortwave infrared bands acquired by Sentinel-2 (band 9, 935-955 µm, band 11, 1567-1658 µm, and band 12, 2114-2889 µm).

As observed, the main limitations of satellite datasets are their temporal and spatial resolution. However, these drawbacks are being mitigated nowadays, especially by commercial missions. For instance, the commercial satellite Pléiades Neo (VHR-2020) from Airbus Defense & Space [20] is able to daily acquire seven bands with a Ground Sampling Distance (GSD) of up to 30 cm for panchromatic images. Even for governmental missions, the repeat cycle is typically reduced as several twin missions could be active at the same moment with similar instruments and orbits. Accordingly, the offset in Landsat-8 and Landsat-9 trajectories allows acquiring data from the same point every 8 days [15]. On the other hand, spatial resolution is steadily improving as most governmental restrictions have been lifted. Yet, spaceborne platforms are the most expensive Remote sensing technology by a huge lead.

In addition to satellite imaging, the use of UAS has considerably increased over the last decade and has attracted the interest of both researchers and commercial applications. The most frequent components of remote sensing UAS are imaging sensors as well as navigation and communication systems. The latter allows the operator to manoeuvre the platform within a communication range and transfer data in a bidirectional stream. Regardless of the navigating mission, either manual or waypoint-based, Global Positioning System (GPS) and Inertial Measurement Unit (IMU) sensors are used to calculate and record the craft's position, orientation and movement. These components are especially relevant to perform accurate flight missions, and their precision is generally described by means of vertical and horizontal error in meters (m). In this regard, the most recent UAS series from DJI achieve vertical and horizontal errors of up to 10 cm using Real-time Kinematic (RTK) positioning rather than GPS. As one would expect, UAS with lightweight and high-performance navigation sensors are more prohibitive.

In terms of safety regulations, the term UAS does not only refer to the vehicle but also to the coupled sensors, which are following introduced. Sensors typically coupled on aircraft vary according to the platform's flight

altitude and cruising speed as well as application requirements. Fixed and rotating wing UAS solutions are considerably more limited in flying height and speed than other airborne vehicles, including helicopters and gyroplanes. Apart from the platform limitations, the flight altitude may be limited by UAS regulations to avoid entering the domain of other airspace vehicles. Therefore, sensors and missions that require lower speed, lower flight altitude, and thus higher precision, are especially convenient for UAS. A widespread example of this is the monitoring of transmission lines, which happens to have a very thin structure and therefore require slower mapping. In comparison, other technologies such as Interferometric synthetic aperture radar (InSAR) have been mainly applied to spaceborne missions to track changes on the Earth's surface.

The operability of unmanned aircraft over the Single European Sky airspace is controlled in the European Union by regulation 2019/947. Among other rules, the maximum flight altitude is established as 120 m over the Earth's surface unless an obstacle is overflown.

1.2 Aims and objectives

According to previously presented challenges, the overall aim of this dissertation is to contribute to some of the stages presented in Figure 1.4. A significant number of challenges have already been briefly reviewed, including the collection of datasets from multiple sensors, difficulties in the correction and fusion of these, the generation of products with larger dimensionality (e.g., 2D → 3D) or the analysis of final products. Accordingly, the objectives of this work are the following:

- The correction and processing of data as collected by airborne surveys. Rather than a single problem, this objective involves studying how can be every data source corrected, including geometrical and radiometric distortions. The first must be solved in most cases, since it enables the later fusion of data, whereas the latter depends on whether the radiance is further analyzed or not.
- The matching of images collected by different sensors, thus overcoming differences concerning 1) triggering timestamps, 2) optical aberrations, 3) optical systems and 4) wavelength intervals. The image matching ought to work over images with notable intensity dissimilarity to obtain reliable transformations that allow projecting one data source into another.
- The efficient generation of large and dense 3D point clouds from multiple data sources without geometric inaccuracies. The proposed pipeline ought to tackle most of the drawbacks of traditional photogrammetry. In addition, this is a time-consuming task that is further stressed by including multiple data sources; hence, it must be addressed using accelerated computing.
- To emulate LiDAR sensors over 3D synthetic scenarios. Nonetheless, this is also a time-consuming task that is especially tough to model in terms of returned intensity. The efficient generation of large labelled LiDAR datasets must be addressed, again, using accelerating computing, by modelling the surfaces' properties.
- To demonstrate the applicability of previous results. Though not exhaustive, this objective involves detecting anomalies, segmenting some items apart from others or phenotyping a large number

Despite visible, multispectral, thermal infrared and hyperspectral datasets covering a wide part of the spectrum, these will be projected to a 3D baseline point cloud generated with photogrammetry. Image-based reconstructions are less reliable than other optical sensing products, such as LiDAR. However, it is also more cost prohibitive than most of the revised optical imaging sensors.

of vegetation varieties. It can be either performed with traditional geometry-based techniques as well as using AI algorithms.

1.3 Organization

This dissertation comprises six different parts which are following detailed:

PART I This part includes the current chapter. A shallow introduction to RS is presented together with the main challenges concerning data fusion. Then, the fundamentals of this dissertation are presented in Chapter 2, and later, a review of the state-of-the-art is given in Chapter 3 to introduce previous work on the fusion and simulation of UAS-based data.

PART II Here, the fundamentals of our methods are presented, including which kind of sensors and data are available. The correction and the image fusion algorithms are isolated in Chapter 5. In summary, this part comprises the materials and methods that are shared among most of this dissertation's chapters.

PART III This part explains the generation of 3D point clouds consisting of multiple layers: visible, infrared, multispectral and hyperspectral data. The visible colour is presented as the baseline 3D point cloud that is enhanced with the rest of the datasets. First, thermographic imagery is fused with RGB data to propose an efficient projection in the Graphics Processing Unit, the so-called GPU. This procedure is following applied to multispectral data, despite this being significantly more time-consuming and intricate since the latter comprises four spectral bands. Then, a hyperspectral point cloud is generated following a different pipeline, yet accelerated, that fits the conditions on which these data were acquired.

PART IV Instead of correcting and fusing real-world datasets, others are simulated using detailed synthetic scenarios that are rapidly labelled with semantic tags. Simulations are aimed at modelling products similar to those collected by real sensors, including their geometrical properties. The fundamentals of LiDAR simulations are reviewed in Chapter 9, whereas radiance properties require further considerations which are detailed in Chapter 10. Then, these simulations are used to assist in the scanning of indoor scenarios such as buildings in Chapter 11.

PART V This part shows multiple applications of the collected data and the derived results, including the simulated LiDAR point clouds, reconstructed thermal point clouds and hyperspectral swaths. The first are Thermal infrared data are applied to the detection of thermal anomalies that may be linked to buried remains at an archaeological site. Hyperspectral swaths are, on the other hand, corrected and processed for the classification of red and white grapevine varieties using Deep Learning.

PART VI Chapter 14 concludes this dissertation by highlighting the main contributions and pointing out future work.

Fundamentals of imaging sensors

2

2.1 Optical imaging systems 19
2.1.1 Scan geometry 21
2.1.2 Multispectral imaging . . . 22
2.1.3 Thermography 26
2.1.4 Hyperspectral imaging . . . 29
2.1.5 Active sensing 31
2.1.6 Principles of physical measurement 34
2.2 Data types, processing and indexing 39
2.2.1 Results from Remote Sensing 39
2.2.2 Data processing 42
2.2.3 Data indexing 50

The main goal of imaging sensors is to obtain measurements from objects and processes to extract quantitative, visual and text-based results. The most frequent form of RS results are images, and the science and technology that transforms images into valuable products is known as photogrammetry. Images are derived from passive sensors that capture the incoming radiance from the surrounding environment. The technology involved in this will be briefly discussed in Section 2.1.1. However, other kinds of sensors operate as active technologies that emit and receive backscattered light from the Earth's surfaces. The latter technology directly captures the geometry of the impacted surfaces, though images can also be derived from these products. The use of active and passive sensors depends on the budget, the required products and how will the conclusions be extracted from these products. In this regard, passive imaging sensors are typically cheaper and 3D reconstructions derived from these may lack accuracy despite being supported by reference data. On the other hand, the results from active sensors record the geometry of items instead of reconstructing them and they are also less likely to be altered by atmospheric effects. As opposed to passive results, deriving imagery from geometry may have some disadvantages due to non-uniform densities and lack of spatial resolution over the covered area. Hence, both technologies have their downsides and thus must be used as complementary data sources.

This section is structured as follows. First, optical imaging systems are presented to cover visible, thermographic and multispectral sensing tools. Then, the fundamentals of LiDAR technology are introduced as another optical imaging technique. Finally, some of the most frequent operations performed over the data collected by the revised sensors are then presented to introduce the reader to the basics.

2.1 Optical imaging systems

The underlying concept in RS is to measure the emitted energy from the Earth's surfaces either from excitation of internal processes or interaction with incoming energy. We refer to energy as electromagnetic (EM) radiation that travels through time and space with a wave-like shape. If these are unpolarized, they oscillate in all the directions that are perpendicular to the travelling vector. The number of oscillations is known as frequency (ν), whereas the spacing between two successive crests is the wavelength, λ. In a vacuum, EM radiation travels at the speed of light, $c = \lambda * \nu$; in nature, it depends on the traversed medium. The fundamental unit in EM radiation is the photon, the physical form of a quantum, and these units are the target of the following described optical imaging tools [12]. These are designed to capture EM radiation passing through a polarized filter, thus focusing

on a specific spectral interval.

The early cameras were simply light-tight boxes with a pinhole from where the light was allowed to pass through. The exposure of the film was controlled by letting light pass or not through it. Then, they were replaced by lenses aimed at enlarging the hole, thus capturing more light in less time. Besides lenses, these systems were also composed of a diaphragm adjusting the diameter of the lens opening, and a shutter that controlled the exposure time. Nowadays cameras are not that different from these early devices, though they have now better control over the focus and exposure. The main concern of this dissertation are sensors measuring EM radiation as functions of location (x, y), time and wavelength (λ). These sensors are classified into imagers, which produce two and three-dimensional images, and nonimagers, such as hand-held devices that output single response values.

Imagers are also known as radiometers, and these instruments require decomposing the radiation contributions and quantifying them with matched detectors. The first dispersion method is a prism, which was first utilized to separate out wavelengths by Isaac Newton. This dispersion results from refractivity being dependent on wavelength, causing entering light rays to exit at different angles. Other dispersion mechanisms are filter-wheel radiometers, grating spectrometers and interferometers, where the former is the most widespread. Filter-wheel radiometers separate wavelengths using a wheel, composed of radiative filters, that cycles through the radiation path. Filters can be classified as absorption- or interference-based, though only the first is integrated into a filter-wheel. They can be designed to let through long, short or very narrow wavelength bands, defined by a central wavelength and cutoff limits. Coloured absorption filters are made of coloured glass or dyed gelatin that absorbs energy from a wavelength interval and let others pass. Furthermore, filters also attenuate the passing radiation, and their radiative efficiency is given by the signal-to-noise ratio (SNR) [12].

On the other hand, the transmitted energy must be captured into images. The transition from analogical to digital photography was led by utilizing solid-state sensors instead of silver halide crystals, with the first being more sensitive to brightness changes. Digital cameras incorporate two-dimensional arrays of semiconductors, either charge-coupled devices (CCD) or complementary metal-oxide (CMOS). Each one of these detectors is dedicated to sensing the energy for an image's pixel. The scene brightness captured in that pixel is proportional to the small electrical charge triggered by the striking energy. Charges are transformed to voltage values using the photodetector's area and responsivity, the incoming radiance and the spectral width, followed by analogical to digital (A-to-D) conversion. Although semiconductors are monochromatic, RGB-colour data can be obtained by placing filters over these photosites in the sensor array with a Bayer pattern. Red, green and blue photosites are arranged so that missing colour data can be interpolated from surrounding photosites with the same colour. With this pattern, half the filters are green, while the remainder is equally split into blue and red. Other detectors are able to record red,

green and blue radiance at every pixel, such as Foveon CMOS [3]. Besides RGB, hyperspectral imaging constitutes a greater challenge as it does not require capturing three simultaneous wavelengths, but hundreds of narrow bands. Hence, photo-detectors are arranged as arrays along the spatial and spectral dimensions. Depending on the wavelength range, the material in which semiconductors are made varies; from silicon (visible) to PbS (lead sulfide, near-IR) or InSb (indium-antimony, mid-IR) [12].

The overall objective of imaging spectrometers is to observe large portions of the Earth and therefore, to progressively build data cubes recording spectral and spatial information. However, it is possible to scan the area of interest with hundreds and thousands of images or simply with a few of them that are built as the scanning platform translates, thus yielding images with higher dimensionality.

2.1.1 Scan geometry

At least three scanning geometries are distinguished, involving different mechanical and optical principles: single-snapshot, whiskbroom and push broom. Further insight into push broom and snapshot scanning is provided in Figure 2.1.

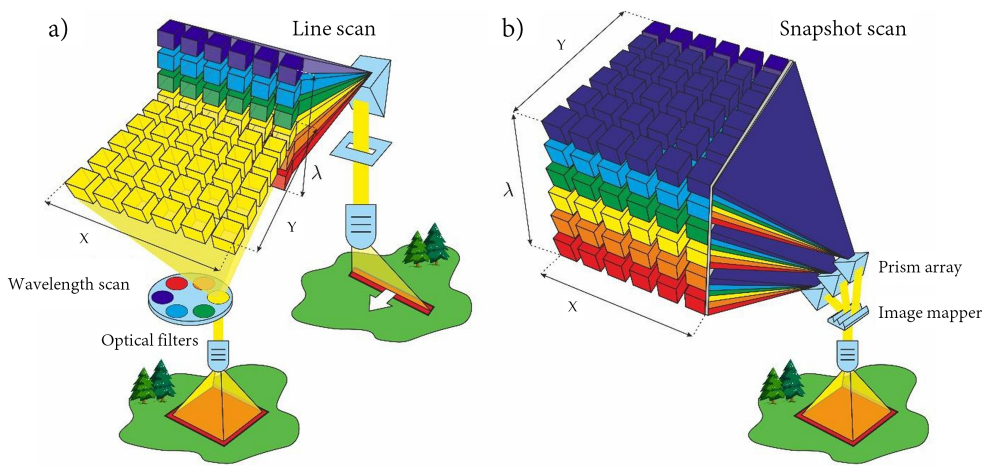


Figure 2.1: Mechanism involved in a) push broom and b) single-snapshot scanning geometries.

Single-snapshot is the most widespread scanning geometry. Matrices of semiconductors simultaneously acquire spatial and spectral information from the area of interest. Therefore, there is no scanning involved; it captures a single snapshot with limited spatial resolution. The main drawbacks are given by the dimensionality of CCD arrays, as they bound the number of pixels. Although it is possible to increase spatial or spectral resolution by enlarging CCD arrays, it may come at the expense of reducing the dimensionality of one axis. This is especially relevant for imaging devices producing large data cubes, e.g., hyperspectral. Despite the advent of large detectors measuring millions of voxels at the same time, the already huge

size of hyperspectral CCDs forbids enlarging axes without penalizing others [21, 22].

On the other hand, push-broom or line-scanning is possible with a 2D matrix of semiconductors ($\lambda \times x$) recording data in the spectral, λ, and spatial, x, dimensions. Since the CCD array is 2D, horizontal sweeps are acquired simultaneously (lines), and these are accumulated into images known as swaths. Thus, this approach requires a mobile platform or imaging device that moves forward and helps to enlarge the observed area in y. Rather than a complex mechanical system, this scanning requires an optical design that collects radiation with pointing mirrors and collimates it into dispersing elements [21, 22].

Whiskbroom geometry is halfway into single-snapshot and push-broom. The core of this approach is a rotating mirror that dispersers radiation through a grating plane or prism into a linear CCD array with a dimensionality of λ. Hence, it requires a lesser number of photo-detectors than previous scanning geometries and consequently, these devices are easier to calibrate while providing a high spectral resolution and radiative sensitivity. Due to the acquisition mechanism, the dwell time on every pixel is also smaller, which leads to a lower Signal-to-Noise Ratio (SNR). However, the misalignment of bands in single-snapshot and lines in push-broom is here found at the pixel level, thus hardening the generation of orthorectified products [22].

Besides the scanning geometry, aerial photographs are taken along flight lines and thus their acquisition setup also matters for the final results. First, these are recorded along a viewing direction, with *nadir* being the most trivial, i.e., perpendicular to the soil represented as a horizontal plane. Furthermore, successive images are taken with a degree of overlapping, either frontal or side overlap. It guarantees that features are visible in more than one image, thus leading to stereoscopic pairs and stereoscopic coverage. Frontal overlap refers to the overlapping between successive images in the same flight strip, whereas side overlap is the degree of overlapping among consecutive strips. The geometric elements of a vertical photograph are straightforward, considering that o is the principal point, i.e., the intersection of the optical axis with the film plane, whereas the optical axis controls the view direction. In Figure 2.2, the photograph is taken with *nadir* view direction.



Figure 2.2: Image captured during a flight planned with *nadir* view direction.

The following sections present the most widespread imaging sensing tools and those that will be covered in the methodology of this work.

2.1.2 Multispectral imaging

Multispectral images consist of multiple bands corresponding to different wavelengths. However, these are typically discrete readings of spectral intervals with varying bandwidths. Most of the research and academic work describe multispectral as a special case of super spectral and hyperspectral imagery, and so we will, though some remarks are mentioned in this section for our later work. First, Table 2.1 presents some of the multispectral sensors

that are frequently revised in the literature, along with the works where they were mentioned. The discrete wavelengths and the resolution of every reading are shown in the fifth column, *Spectral bands*. It can be observed that only a few bands with a coarse resolution, ranging from 14 μm to 57 μm, are recorded.

The most frequently recorded bands are Red, Green, Blue, Red-Edge (REG) and Near-Infrared (NIR). REG band is halfway between Red and NIR, whereas NIR precedes the short-wave infrared interval. Despite other sensors being able to capture most of these individual bands, multispectral imaging opens a wide number of opportunities by simultaneously recording bands sparsed throughout the visible and infrared spectrum. Most of the multispectral applications benefit from the use of multiple bands through the so-called indices, i.e, linear combinations of them aimed at highlighting relevant features in a group of target objects. In this regard, a total of 82 spectral vegetation indices were collected from the literature by Pu, Ruiliang [22]. A recurrent vegetation index in PA work is the NDVI (Normalized difference vegetation index), calculated as a combination of Red and NIR bands. Besides identifying vegetation, other indices are focused on estimating pigments and chlorophyll content in plants [22]. Most of them work by aggregating bands where the target compound has the most and less significant presence, thus allowing us to measure normalized amounts.

For tasks such as vegetation classification, visible and near-infrared bands from multispectral imagery are sufficient in most cases. Bare soils tend to show a more constant spectral signature, whereas vegetation presents a steeper slope with higher values in the near-infrared wavelengths [29].

Table 2.1: Consumer-grade multispectral devices found in the literature and their specifications, according to the manufacturer's source.

Multispectral tool	Focal length	HFOV	Image resolution	Spectral bands (μm)	Ref. work
			Parrot		
Sequoia	3.98 mm	63.9°	1280 × 960 px	Green: 0.55 ± 0.02 Red: 0.66 ± 0.02 Red-Edge: 0.735 ± 0.005 Near-IR: 0.79 ± 0.02	[23]
			Micasense		
RedEdge-MX	5.4 mm	47.2°	1280 × 960 px	Blue: 0.475 ± 0.016 Green: 0.56 ± 0.013 Red: 0.668 ± 0.07 Red-Edge: 0.717 ± 0.06 Near-IR: 0.842 ± 0.028	[24, 25]
Dual-Camera	5.4 mm	47.2°	1280 × 960 px	Coast blue: 0.444 ± 0.014 Blue: 0.475 ± 0.016 Green 1: 0.531 ± 0.007 Green 2: 0.56 ± 0.013 Red 1: 0.65 ± 0.008 Red 2: 0.668 ± 0.007 Red-Edge 1: 0.705 ± 0.005 Red-Edge 2: 0.717 ± 0.006 Red-Edge 3: 0.74 ± 0.009 Near-IR: 0.842 ± 0.028	[26]
Altum	8 mm	50°	2064 × 1544 px	Blue: 0.575 ± 0.016 Green: 0.56 ± 0.013 Red: 0.668 ± 0.007 Red-Edge: 0.717 ± 0.006 Near-IR: 0.842 ± 0.028	[27]
	1.77 mm	57°	160 × 120 px	Long-Wave IR: 11 ± 3	
Altum-PT	8 mm	50°	2064 × 1544 px	Blue: 0.575 ± 0.016 Green: 0.56 ± 0.013 Red: 0.668 ± 0.007 Red-Edge: 0.717 ± 0.006 Near-IR: 0.842 ± 0.028	[27]
	1.77 mm	48°	320 × 256 px	Long-Wave IR: 11 ± 3	
			DJI		
P4 Multispectral	5.74 mm	62.7°	1600 × 1300 px	Blue: 0.45 ± 0.008 Green: 0.56 ± 0.008 Red: 0.65 ± 0.008 Red-Edge: 0.73 ± 0.008 Near-IR: 0.84 ± 0.013	[28]

Activation of relevant spectral bands

As explained, the aim of spectral indices is to measure or simply indicate the presence of a compound: vegetation, water, etc. A rapid way to determine which bands have a higher weight to highlight a certain compound is to train a trivial neural network with five trainable parameters (Figure 2.3). These correspond to four weights ($W^{4\times1}$) belonging to Green, NIR, Red and REG spectral bands, plus one bias, b^1. It can be trained over perfectly overlapped multispectral datasets, where the ground truth annotates vegetation and ground. Neither it is a great classifier nor do we plan it to be as so. It was randomly initialized, and the first fitting obtained an overall accuracy close to 98% and yielded $W = [-3.6329935, 33.97777, -67.50758, 2.1926873]^\intercal$. Hence, NIR and Red bands had a higher contribution to the computed probability, which also happens to correspond to $w_2 \cdot NIR$ - $w_3 \cdot Red$. NDVI is, on the other hand, defined in Equation 2.1:

$$NDVI = \frac{NIR - Red}{NIR + Red} \tag{2.1}$$

where the aim of $NIR + Red$ is to normalize the contribution of both spectral bands so that it is minimized when $\lim_{NIR\to Red} NDVI = 0$ and maximized if $\lim_{NIR\to 1-Red} NDVI = 1$, with NIR, Red $\in [0, 1]$. Therefore, the numerator was directly explained by a neural network with a single dense layer, whereas the normalization was achieved with a simple sigmoid activation, $x \leftarrow w_2 \cdot NIR$ - $w_3 \cdot Red$; $s(x) = \frac{1}{1+e^{-x}}$, to output a probability $s(x) \in [0, 1]$. It is intended to approximately infer and emulate the *NDVI* formula; nevertheless, neither $s(x)$ is linear as defined nor the weights were accurately estimated (note that $w_3 > -2 \cdot w_2$).

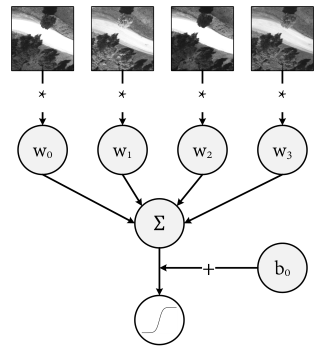


Figure 2.3: Architecture of a simple neural network with a single layer, aimed at learning the weights of importance of every spectral band.

However, the revision of vegetation indices converges into the previously mentioned shortcomings of multispectral imagery. Most of these indices operate very specific wavelengths, e.g., 670, 800, 1050, etc., while only a few bands are captured in multispectral imaging. Although it is possible to approximate missing wavelengths with the most similar bands, as shown in Code 2.1, the results are far from optimal. Therefore, multispectral imaging is better suited for trivial applications such as vegetation classification, while it is not sufficient for other applications that require higher spectral resolution, such as mineral mapping or classifying vegetation species [22, 29]. Indeed, some metrics proposed in the literature for discerning spectral signatures are based on their derivative. On the other hand, multispectral bands represent a step function with some intervals being undefined, and yet, the covered wavelengths have a single value.

```
1 def __search_nearest_layer(bands, wl):
2     b, e, idx = 0, len(bands) - 1, 0    # Begin, end
3
4     while True:
5         m = floor((b + e) / 2.0)  # Middle
```

Listing 2.1: Function for finding the most similar wavelength among those available for a specific sensor.

```
        if b == e or bands[m] <= wl < bands[m + 1]:
            idx = m
            break
        elif wl < bands[m]:
            e = m - 1
        else:
            b = m + 1

    if idx == (len(bands) - 1):
        return idx
    else:
        if abs(wl - bands[idx]) < abs(wl - bands[idx + 1]):
            return idx
        else:
            return idx + 1
```

Recently, novel aerial platforms and multispectral sensors are emerging in order to increase the accuracy of spectral measurements, reduce the noise coming from different layers of the atmosphere, and increase the model coverage by multiple view-points with a higher spatial resolution [30]. The UAS's capabilities to carry lightweight imaging systems have positively impacted recent research. In contrast to satellite images, the GSD of multispectral imagery collected from UASs can be significantly reduced to a few centimetres. The increase in spatial resolution and the possibility to capture many overlapped images enable the development of image-based techniques for 3D model reconstruction.

Finally, remark that the radiometric processing of multispectral imagery from commercial devices is performed with proprietary software and formulae defined by manufacturers. The acquired irradiance at discrete wavelengths is not transformed in a manner that involves the spectral signature. Instead, irradiance is typically turned into reflectance using the shutter speed, the observation angle, the gain factor and other calibration coefficients, either provided by the manufacturer or obtained before and after flights [31, 32].

2.1.3 Thermography

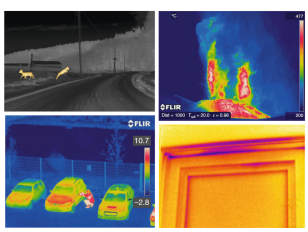


Figure 2.4: Some applications of infrared imaging [33]. From left to right, and from top to bottom: night vision, volcano monitoring, surveillance and detection of incorrect thermal insulation.

Infrared thermal imaging (IR) or thermography is a noninvasive technology for acquiring the temperature of objects' surfaces and phenomena through their emitted radiation. In recent years, rapid growth has been witnessed due to the progress made in the technology of infrared detectors. Furthermore, the competition among camera manufacturers has led to releasing low-cost IR products, thus widening the applicability scope and diversity of customers. It is considered an excellent noninvasive technology, and therefore, it is applied over very different scopes. Some of the most revised applications in the literature are quality control and monitoring in industrial environments [33, 34], medical analysis [35], building inspection [36, 37], agriculture and animal applications [38, 39], geological monitoring [40], fire and gas leaks detection [41], surveillance, etc. (see Figure 2.4). On crop

and forest management, it has been mainly applied to tree characterization by measuring their energy flux, evapotranspiration and photosynthesis [42], as well as for water stress and disease detection [43–45]. Heating monitoring also prevents frost damage from crops [46].

Discovery of infrared radiation

The heat was initially believed to only transfer through matter until the experiments of William Herschel led to the discovery of 'calorific' or 'invisible' rays. This term referred to wavelengths that fall beyond the red spectral cutoff limits and thus are invisible to the naked eye. Circa 1737, Châtelet suggested that different colours could carry different temperatures, though it would not be until 1800 that this theory was proved. W. Herschel passed the Sun's radiation through a prism and placed mercury thermometers over the split spectrum. The measured temperatures not only differed from atmospheric temperature but also the readings beyond the red end were higher. A more advanced set-up of this experiment is shown in Figure 2.5, with a reflective mirror and a platform that can vary its slope [47–49]. Later in 1804, John Leslie checked the discrepancies in the temperature readings of different materials, using the so-called Leslie cube. Another curious experiment showed that there exist atmospheric windows with a higher transmission factor and less absorption. W. Herschel's son, John Herschel, used the technique called evaporography to create a thermal image, also known as a thermogram, over a wet piece of paper. The result is shown in Figure 2.6, with white dots representing beyond infrared wavelengths.

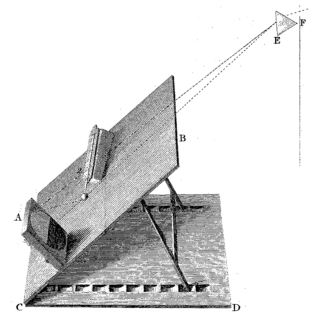


Figure 2.5: Measurement stand for reading temperature from radiation with a single thermometer [47].



Figure 2.6: Example of thermogram originated from a wet piece of paper and targetted radiation.

IR radiation is physically represented as electromagnetic (EM) waves that propagate through space as a sinusoidal function ($f(x) = k \cdot \sin x$) with a spatial periodicity, known as wavelength (m), λ, and a period of oscillation (s), $\mathcal{T}$. Waves are affected by disturbance, and according to this can be organized into longitudinal and transverse waves, with both being perpendicular. Transverse waves can oscillate in many different directions, which yields the concept of polarization and plane of polarization. The latter is defined through the direction of propagation and the disturbance. If a wave is categorized as unpolarized, then the plane of polarization can adopt all possible orientations. Regarding IR radiation, it is perturbed by electric and magnetic fields, and its polarization is expressed through the electric field and the direction of propagation ($x - z$ plane). Therefore, the recording of waves and IR radiation can be performed by polarizers, with a simple microscopically small conducting grid being sufficient for filtering out waves that are non-perpendicular to such a grid [33]. In thermography, the captured EM radiation is referred to **thermal radiation**, emitted by bodies at a temperature over 0 K (-273.15°C) [12, 33].

The radiation measured from a surface is not solely affected by such a surface, but other factors intervene in the acquisition. Among these, the most notable factors are the absorption and scattering phenomena from atmospheric particles and sensor components, as well as the radiation emitted

by surrounding objects and the atmosphere. The degree of transmissivity of the atmosphere, which may vary throughout time, is known as the transmission factor. Other minor influencing factors are the distance to the sensor, the object's emissivity, dimensions and shape, the atmospheric temperature and humidity as well as the temperature of optics [33].

Although the IR spectrum covers a wide range of wavelengths (see Figure 1.4), IR imaging is focused on a small part of these. Three spectral ranges are the most frequent in thermography: short-wave infrared (SWIR: 900-1,700 nm), mid-wave infrared (MWIR: 3,000-5,000 nm) and long-wave infrared (LWIR: 8,000-14,000 nm) [33, 41]. Commercial sensors are available on the three spectral ranges; whether one of them ought to be used over another must attend to criteria such as atmospheric transmission, expected radiation and the physics of detectors. A simple Scopus search shows that MWIR imaging is the most widespread, though it is significantly downgraded over SWIR and LWIR for UAS platforms. Figure 2.7 shows the percentage of work devoted to each IR interval. The Scopus searches were the following: $(thermographic \vee thermography \vee thermal \vee infrared \vee ir) \wedge (drone \vee uav \vee uas) \wedge ((x - wave \vee y\,WIR))$, with x being short ($y \leftarrow$S), mid ($y \leftarrow$M) or long ($y \leftarrow$L). In this dissertation, the integrated thermographic camera operates over the LWIR range.

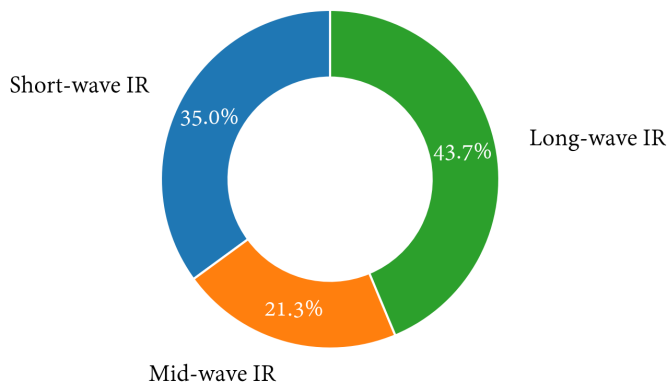


Figure 2.7: Number of manuscripts related to different IR intervals.

Preferences on the LW range over SWIR and MWIR are not a casualty, but it is rather influenced by atmospheric transmissivity. Most of Earth's surface has its emission peak in this interval, and it is also known to be an atmospheric window where absorption is considerably smaller [50, 51].

An exhaustive, though not complete, list of the most frequent commercial IR cameras used for research is shown in Table 2.2. Every single of them is designed for measuring LWIR, while their focal length is typically over 13 mm. Instead of widening the camera view, these devices acquire images focused on very specific areas with a very small image resolution. Therefore, these drawbacks ought to be considered to make an appropriate flight plan. For instance, the side overlapping among different sweeps from a trajectory must be large enough to guarantee there exists overlapping between thermal imagery.

Table 2.2: Consumer-grade thermal devices found in the literature and their specifications, according to the manufacturer's datasheets.

Thermographic tool	Focal length	FOV	Image res.	Spectral range	Ref. work
		DJI			
Zenmuse XT2	13 mm	$32° \times 26°$	640×512 px	LWIR (7.5 - 13.5 µm)	[46, 52, 53]
	19 mm	$25° \times 19°$	336×256 px		
Zenmuse H20T	13.5 mm	$31.7° \times 25.36°$	640×512 px	LWIR (8 - 14 µm)	[54, 55]
		FLIR			
FLIR A320	18 mm	$25° \times 18.8°$	320×240 px	LWIR (7.5 - 13.5 µm)	[56]
FLIR A35	9 mm	$48° \times 39°$	320×256 px	LWIR (7.5 - 13.5 µm)	[57]
FLIR One	15 cm	$50° \times 38°$	80×60 px	LWIR (8 - 14 µm)	[58]
FLIR A65	25 mm	$25° \times 20°$	640×512 px	LWIR (7.5 - 13.5 µm)	[36, 59–61]
FLIR Tau2 640	13 mm	$45° \times 37°$	640×512 px	LWIR (7.5 - 13.5 µm)	[62, 63]
		Sensefly			
Sensefly thermoMAP	N.A.	N.A.	640×512 px	LWIR (7.5 - 13.5 µm)	[64]

2.1.4 Hyperspectral imaging

Hyperspectral imaging (HSI) differs from multispectral imaging in the number of bands, wavelengths and bandwidth of these. Hyperspectral images are 3D data cubes known as **hypercubes**, where each pixel is defined within a continuous spectrum range (Figure 2.8). The spectral resolution is typically given in nm or as a number of bands, though the second is preferred as the resolution can vary throughout the covered interval. Spectral signatures are complete and differentiable over the observed wavelength range, and thus, hypercubes outperform discrete multispectral data on the classification, segmentation and unmixing of materials. For instance, different varieties of plants might have a radiative response too similar among them, even using HSI. Therefore, AI algorithms could be applied to the automatic detection of the most discriminative wavelengths and be accordingly weighted to provide a meaningful result from input data.

Unlike other imaging tools, HSI has undergone a rising path from major to minor; from satellite scanning to UAS and laboratory applications. Hence, it is applied in a wide range of applications, from phenotyping, cultural heritage, medical research and food processing to biochemistry and the monitoring of Earth's processes and areas of interest, among others [65]. Also in contrast to multispectral imaging, preprocessing on HSI is still very frequently investigated.

Remotely sensed data is affected by several factors, including acquisition conditions and random errors caused by sensors as well as atmospheric and surface effects. Radiometric corrections over HSI have been frequently performed to obtain representative data from the Earth's surfaces. The bibliography on this field has covered many topics, though it is mainly focused on satellite imaging. While some of the studied techniques can be applied to UAS imaging, other chapters are irrelevant to our target platform, UAS. For instance, atmospheric effects such as absorption are of no interest in close-range work. However, the low flight altitude, the UAS instability or its viewing angle harden the pre-processing operations

[66]. Geometric and radiometric corrections as well as spectral calibrations are also required to obtain accurate data [67]. Geometric distortions refer to those defects caused primarily by the UAS instability as well as due to the acquisition technique. For instance, push-broom sensors present higher geometric distortions that can be reduced by means of stabilizers. Geometric correction can be achieved with an IMU, a GPS as well as a Digital Elevation Model (DEM). Although there exists commercial software based on this approach, it requires high-precision systems for obtaining accurate corrections. Instead, Ground Control Points (GCP) have been extensively used to ensure correct positioning [68]. Also, the dual acquisition of visible and hyperspectral imagery allows matching both data sources [68–70], with the visible data being geometrically correct. Alternatively, feature matching among overlapping images has also been shown to help with geometric correction [71].

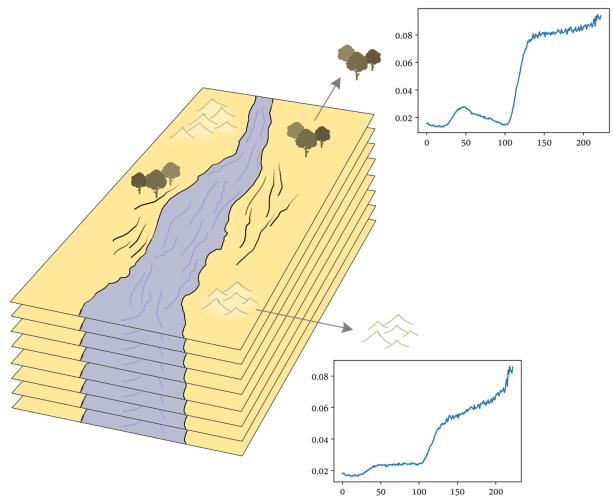


Figure 2.8: Scheme of hyperspectral images, where vegetation and ground surfaces have a different spectral signature.

Similarly to geometric distortions, radiometric defects can be corrected with software tools from the hyperspectral manufacturer. The objective is to transform the sensor's DN to radiance, influenced by acquisition conditions, and reflectance, regardless of previous conditions. To this end, the coefficients needed for this correction are typically calibrated in the laboratory. However, they are more likely to vary through time [67], thus leading to worsening the radiometric correction. This process can be supported by the use of grayscale tarps whose reflectance is known, thereby allowing to perform linear interpolations to calibrate the acquired DN [72] with the Empirical Line Method (ELM) [21, 73]. Dark and grey/white references are required to perform this linear interpolation [66, 74, 75], typically from isotropic materials (near to Lambertian) that present a grayscale palette. Alternatively, the acquisition of radiance samples can help to transform DNs with fitting methods such as the Least-Square Method (LSM). Sensor-based factors, including the vignetting effect that causes a drop in the intensity, lead to a different radiometric response in pixels of the same bands [76].

Atmospheric corrections have also been documented for UAS flights. Ac-

cordingly, the at-sensor radiance can be correlated to the surface Hemispherical-Directional Reflectance Function (HDRF) measured in the laboratory beforehand. The previous grayscale palette with Lambertian materials can be used to achieve this [72]. Physically-based methods have also been investigated by measuring the atmospheric conditions that influence hyperspectral imaging, though they are more time-consuming. Finally, some public repositories offer atmospherically corrected datasets, e.g., the Aviris data portal [77], whereas widespread software for hyperspectral processing also includes semi-automatic correction algorithms (ENVI, Headwall SpectralView, etc). Other studies calculate the Bidirectional Reflectance Distribution Function (BRDF) of visible materials. Previous corrections are applied, and BRDF kernels are estimated, such as Ross and Li, from the varying response of materials according to solar and view angles [74, 78, 79].

The list of hyperspectral sensors on the market that can be coupled with UAS has significantly increased in recent years. Therefore, Table 2.3 solely illustrates those that provide higher spectral resolution and have been explored in the hyperspectral state-of-the-art literature. As previously regarded, the spectral range can be calculated from the spectral resolution, in nm, and the number of bands. However, as observed in Table 2.3, this is not exactly right for every sensor as the spectral resolution may not be constant. In this dissertation, the research focused on hyperspectral imaging operates over data cubes from Headwall Photonics Inc.®Nano HyperSpec, with sampling intervals ranging from 2.2 nm to 6 nm in the central wavelengths.

Table 2.3: Hyperspectral sensors found in the literature and their specifications, according to the manufacturer's datasheets.

Tool	Spectral range	#Bands	Spectral res.	Acquisition	Spatial res.	Ref. work
			Headwall Photonics			
Nano HyperSpec	400-1000 nm	272-775	6 nm	Push-broom	640 px	[21, 80]
Micro Hyperspec	400-1000 nm	837-923	2.5 nm	Push-broom	1004 px	[21]
			SENOP			
VIS-VNIR Snapshot	400-900 nm	380	10 nm	Snapshot	1010×1010 px	[21]
HSC-2	450-900 nm	Up to 1000	6-18 nm	Snapshot	1024×1024 px	[21]
			XIMEA			
XIMEA SNm4x4	460-600 nm	16	15 nm	Snapshot	512×272 px	[81]
			Specim			
AISA Angle	400-970 nm	488	2.9 nm	Pushbroom	1024 px	[82]
FX17	900-1700 nm	224	8 nm	Pushbroom	640 px	[21]
			Cubert GmbH			
UHD 185 Firefly	450-950 nm	125	8 nm	Snapshot	1000×1000 px	[83]

2.1.5 Active sensing

Active systems, unlike previous passive sensors, have their own energy source and therefore do not rely on external light sources, radiation emit-

ted and reflected by surfaces and weather conditions [84]. Among active sensors, the focus of this dissertation is on LiDAR. It is an optical remote sensing technique that measures the properties of scattered light and can be used on spaceborne, airborne and ground-based platforms. The emitted energy is bounced back after colliding on the Earth's surfaces, thus being capable of determining distances by measuring the time delay between emission and reception, the so-called Time of Flight (ToF). It is also aimed at measuring speed (wind LiDAR) and attenuation of volume targets such as the atmosphere or seawater [12]. The emitted electromagnetic energy operates at a wavelength, with LiDAR working at much shorter wavelengths than radar, typically in atmospheric transmission windows. Some of these were previously reviewed (infrared), although LiDAR systems also operate at the visible and ultraviolet (UV) spectral range. This parameter determines which is the minimum size of objects to be detected, with shorter wavelengths being more sensitive to smaller particles. Hence, LiDAR systems are typically used for atmospheric and meteorology research. Backscattering is dependent on the dielectric discontinuity produced by collided surfaces, and therefore, nonmetallic objects have significantly lower responses to higher wavelengths [84].

A simplification of LiDAR emissions is the concept of ray: $r(t) = o + \vec{d}t$. This is a geometric data type defined by the starting position, o, and a direction vector, $\vec{d}$, that infinitely spread in the space. Unlike rays in Computer Graphics (CG), LiDAR emit volumetric beams whose targets are surface areas rather than points. Therefore, impacts on surfaces may not entirely block these beams, thus leading to penetrating surfaces that harden the visualization of objects below it but are not completely enclosed. For example, LiDAR has been applied to archaeology with notable results since it allows removing points from high-density forests to uncover remains at the Earth's surface. Other widespread applications of LiDAR technology are found in geology, atmospheric physics or seismology [12]. More specific ones concerning our work are building inspections [85], monitoring of natural environments (landform dynamics [86], ecological resilience [87], etc.), autonomous driving [88], preservation of cultural heritage [89], land mapping or urban planning [90].

Similar to previous optical sensors, LiDAR sensors are also influenced by factors such as the local incidence angle between the surface normal and beams, γ, and the dielectric properties of materials, expressed as $\varepsilon = \varepsilon' + j\varepsilon''$, with the dielectric constant depending on the permittivity (ε') and the loss factor (the imaginary part, ε''). Other major factors are polarization, i.e., the orientation of the transmitted electromagnetic vectors, and the surface roughness.

Most common LiDAR operates on the 600-1,000 nm spectral range for non-scientific applications; ground-based are frequently in 500-600 nm, whereas airborne systems work over 1,000-1,600 nm. The second is of special interest for this dissertation and thus we will stick to it. Despite being more expensive, these are also eye-safe due to working at larger wavelengths. Nevertheless, most commercial sensors operating below 1,000 nm are eye-safe nowadays. Airborne LiDAR can record several discrete mea-

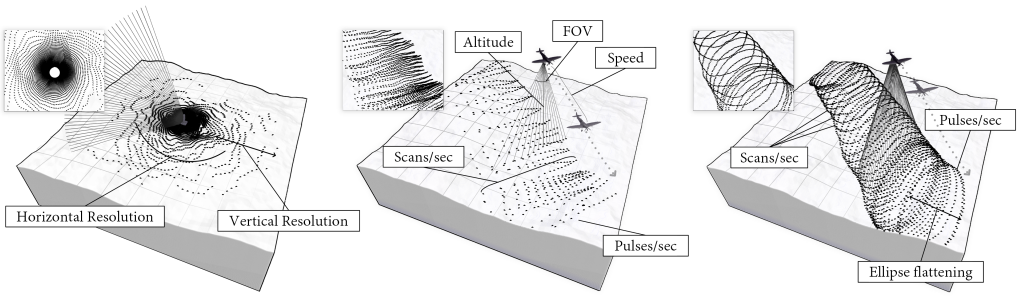


Figure 2.9: Different types of LiDAR and scanning geometries: a) terrestrial LiDAR, b) airborne LiDAR with a zig-zag scan pattern, and c) the same mission than b) using an ellipsoidal scan pattern.

surements per emitted pulse, or record a signal sampled with a frequency determined by time interval or distance (for example, sampled every 1 ns or 15 cm). The latter is known as full-waveform LiDAR and is mainly used in forestry applications since it provides a signature that helps on estimating the density at different height levels. On the other hand, discrete-return LiDAR has found a wider number of applications. A minimal operative LiDAR is composed of laser and receiver units, GPS and IMU systems and control, monitoring and recording units. Hence, one of the main shortcomings of imagery is overcome by LiDAR, as recorded 3D points are georeferenced.

$$R = \frac{1}{2} c t_s \tag{2.2}$$

Equations 2.2 and 2.3 may shed some light on the functioning of LiDAR systems. First, the distance of the surface that returned the backscattered energy, R, is calculated according to the time delay, t_s, and the speed of light, c. Also, the acquired energy, involving both emitter and receiver, is expressed as shown in Equation 2.3. There are multiple formulae in the literature to represent how is LiDAR intensity retrieved, though most are analogous [84, 91, 92].

$$\begin{aligned} \sigma &= \frac{4\pi f_r(\vec{w}) A_t}{\Omega} \\ A_t &= \frac{\pi R^2 \beta_t^2}{4} \\ P_r &= \frac{\bar{I} D_r^2 \eta_{atm} \eta_{sys} \sigma}{4\pi R^4} = \frac{4\pi^2 \bar{I} D_r^2 \eta_{atm} \eta_{sys} f_r(\vec{w})}{4\Omega R^2 \beta_t^2} \end{aligned} \tag{2.3}$$

where $\bar{I}$ is the emitted energy, D_r^2 is the receiver diameter (m), η_{atm}, η_{sys} are atmospheric and system transmission factors, $f_r(\vec{w})$ is the target reflectance, A_t is the target area, R is the distance to receiver in m, β is the transmit beam width (rad) and Ω is the scattering solid angle (sr). π controls the amount of energy spreading a specific point since it spreads in every possible direction within a hemisphere.

Lasers are the main unit in LiDAR systems, and energy emitted by these can be oriented through a scanning deflector to output different scanning

Table 2.4: Most common LiDAR systems in the literature and their attributes.

LiDAR system	FOV	Echoes	Maximum range	Wavelength	Channels	Points/s
			Velodyne			
HDL-32E	$360° \times 41.3°$	2	80-100 m	903 nm	32	1.39M
Puck/Puck LITE	$360° \times 30°$	2	100 m	903 nm	16	600K
Puck Hi-RES	$360° \times 20°$	2	100 m	903 nm	16	600K
ULTRA Puck	$360° \times 40°$	2	100 m	903 nm	32	1.2M
Alpha Prime	$360° \times 40°$	2	250 m	903 nm	128	4.8M
			Valeo			
SCALA Gen 2	$133° \times 10°$	3	200 m	905 nm	16	25Hz
			SICK			
MRS1104C-111011	$275° \times 7.5°$	3	64 m	850 nm	32	165K
			Phoenix Systems			
SCOUT-M2X	$360° \times 40.3°$	3	300 m	905 nm	32	640K
SCOUT-32	$360° \times 40°$	2	220 m	905 nm	32	600K
HydroRANGER	40° VFOV	15	75 m	532 nm	-	200K
RANGER-LR LITE	360° HFOV	5	1000 m	1550 nm	-	1.5M

patterns, as depicted in Figure 2.9. If the orientation of lasers varies, so does the scanning pattern. The fundamental deflections are based on polygonal mirroring, which generates parallel scan lines and swinging mirrors with zig-zag patterns, thus achieving more point density at the line boundaries. Other deflections are fibre optics, which produce a similar output to polygonal mirroring despite changing beam transmission technology, and rotating slanted mirrors that generate elliptical shapes. Finally, Risley prism scanners such as DJI Livox have a different pattern for sampling surfaces with more density and longer integration times [93, 94].

Table 2.4 shows a brief list of some widespread LiDAR sensors, both terrestrial and airborne. It provides further insight into which are the most frequent wavelengths. Among these, LiDAR sensors operating at 532 nm are also known as bathymetric and enable capturing shallow, underwater surfaces. Thus, it is especially relevant for hydrology and coastal monitoring.

2.1.6 Principles of physical measurement

RS covers a wide range of spectral wavelengths depicted in images that show the emitted and reflected radiation from the Earth's surfaces, including matter such as atmospheric particles and bare soil. A huge number of conditions and parameters are involved in the observed radiation, whose explanation would cover dozens of pages, if not a book. Hence, only a few remarks will be done here.

A key ingredient in the understanding of the scene is the detailed specification of the appearance of materials regardless of light and the influence of other matter. Energy, as captured in nature, does not depict the underlying appearance of materials but the appearance when some specific circum-

stances are met. Therefore, measurements ought to be processed to make them independent from environmental and light conditions as well as surrounding surfaces. Not only it helps in the classification of materials present in the scene, given some reference spectral signatures, but also enables simulating lighting as occurs in real-world scenarios. For the latter application, surfaces are assigned a spectral appearance that will not be rendered as so and will undergo changes as light and other matter interfere.

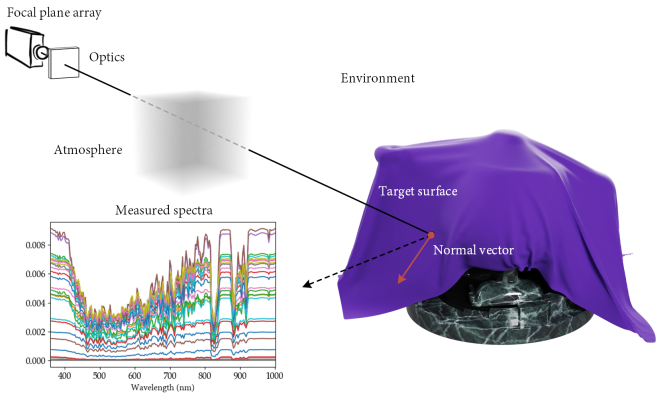


Figure 2.10: Parameters that intervene in the measurement of radiance with a passive sensing tool.

Vollmer and Mollmann [33] provided an exhaustive list of parameters conditioning the material emissivity, regardless of surrounding objects. Materials are the most differentiating feature, naïvely split into metal and nonmetal, where the latter are frequently returning higher emissivity. Also, the roughness level affects the outcome considerably; polished surfaces have lower emissivity and the contrary happens for rough surfaces. Further insight into energy dispersion helps to understand that it does not propagate over every direction of a hemisphere with the same strength. Unlike blackbodies, which behave as perfect isotropic emitters, materials vary the emitted radiation conditioned to the outgoing direction. A key factor in this regard is the normal vector of the surface: the smaller the angle between the camera view direction and the normal vector, the higher the observed emissivity. However, this does not hold up for every material since conductors show the opposite behaviour. Similarly, the physical behaviour of light transport, including reflections, leads to increasing the emissivity for grooved surfaces as a consequence of rebounds. As will be observed throughout this dissertation, the spectral signature of materials is not constant, and therefore, some intervals are preferred over others in the seek of capturing a high emissivity from the target surfaces. For example, the NIR interval comprises the peak emissivity of vegetation. Finally, surface temperature also leads to slight emissivity changes.



Figure 2.11: Subsurface scattering simulated over a triangle mesh.

Radiometric measurements are layered data that can be carved to find the surfaces' spectral signatures. However, the carving process relies on corrections that are simple approximations considering a few of the mentioned factors rather than the overwhelming amount of uncontrolled attributes that can be found in nature. Otherwise, measurements can be performed with controlled surface features and lighting to provide databases of a wide

range of materials [95]. Furthermore, databases on surface appearance comprise a number of discrete spectral signatures for each material, typically parameterized by the input vector, ω_i, and the output direction where radiance spreads, ω_o (Figure 2.12). Otherwise, these vectors can be notated as local cartesian coordinates, half-angle difference parametrization and spherical coordinates, which are also very frequent [96]. The latter are expressed through four angles, $\varphi_i = \text{atan}(y_i/x_i)$, $\varphi_o = \text{atan}(y_o/x_o)$, $\theta_i = \text{acos}(z_i)$ and $\theta_o = \text{acos}(z_o)$. These functions, provisionally expressed as $f(\omega_i, \omega_i)$, parameterized with ω_i, ω_o, or their equivalent spherical coordinates, represent what was coined as Bidirectional Reflectance Distribution Functions (BRDF), or yet more intricate, Bidirectional Scatter Distribution Functions (BSDF) and Bidirectional Scattering Surface Distribution Functions (BSSDF). Both BSDF and BSSDF are aimed at simulating the indirect light transport from surrounding surfaces as well as their own surfaces. BSSDF are particularly hard to calculate as the entering and exit points vary, i.e., surfaces are not opaque (see Figure 2.11); instead, light travels through matter to provide a transparent-like, yet opaque, appearance. From now on, we will only refer to BRDF as the two last functions are beyond the scope of this dissertation.

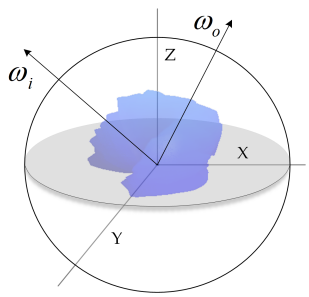


Figure 2.12: Parameterization of the BRDF response.

In the last years, hyperspectral data acquisition has significantly increased, benefiting from the use of more cost-effective sensors and versatile platforms. On the one hand, traditional systems, which are deployed in the laboratory, are currently more efficient and capable of getting accurate spectral measurements of isotropic and anisotropic materials. These are gaining in accuracy, efficiency, and ease of use. At the laboratory level, the most accurate and well-known technique for collecting the optical features and BRDF of any real-world material is the use of a gonio-photometer [97]. Different variants of this configuration have been developed either to obtain better results in anisotropic materials, to take into account refraction phenomena or even to obtain normal maps, and any other characteristic that helps in synthetic modelling [98, 99].

The first databases only included anisotropic materials that captured red, green and blue spectral response as a dense set of measurements [100], instead of analytical functions. Thus, discretized representations are more likely to be parameterized to build larger, yet realistic, BRDF datasets [101]. More recently, the provided wavelengths have been extended to the whole spectra available for a motorized gonio-photometer, thus allowing to extend render engines beyond the visible spectra [95]. As a result, BRDF are not solely parameterized through two spatial vectors, but also with a radiometric value. Hence, f is now expressed as $f(\omega_i, \omega_o, \lambda)$, with λ being the target wavelength. Accordingly, coarse radiometric resolution can be simulated as $\int_{\lambda_i}^{\lambda_j} f(\omega_i, \omega_o, \lambda)\ d\lambda$.

These advances lead us toward the generation of real-world material databases, which are notably used for rendering photorealistic scenes, virtual interior design in architecture, the game industry, etc. These applications can be enhanced by the increasing availability of real-world scenarios, which can be efficiently reconstructed using versatile platforms. However, building material databases is time-consuming, prohibitive when it comes

to acquisition devices and requires considering a list of laboratory constraints.

Due to the large measurement time, UAS flights carrying a hyperspectral sensor have been proposed for the collection of material BRDFs. These come at the expense of noisier data that must be corrected afterwards. In addition, a single or a few flights yield very incomplete spatial response due to the low variety of measurements for ω_i and ω_o [102]. There have been studies on atmospheric corrections for UAS flights, which involve correlating the at-sensor radiance to the HDRF measured in the laboratory beforehand. For instance, grayscale palettes in Lambertian materials can be used for this purpose [72]. Physically-based methods have also been explored, but they tend to be more time-consuming. In addition, public repositories such as the Aviris data portal (California Institute of Technology) [77] offer atmospherically corrected datasets. Other widespread hyperspectral processing software, such as ENVI and Headwall SpectralView, include semi-automatic correction algorithms [74, 78, 79].

The lack of computational power and real-world material samples also led to approximate realistic shading with analytical functions. These are easier to solve and integrate into script-based routines such as vertex and fragment shaders. They intend to emulate the appearance of real-world materials; however, continuous functions are smoothed-out versions that include errors as a consequence of fitting real data. Furthermore, fitting functions in non-linear optimizations is expensive to compute, although analytical functions have a low memory footprint and enable simulating a wide range of materials (with more or less precision). Hence, this BRDF representation was widely adopted until recently.

Analytical BRDFs have a large taxonomy if we attend to the kind of material and the criteria under which these were modelled. Although most of them aim to simulate a specific appearance (metal, wood, mats, polished surfaces, the Moon, etc.), materials are at least organized into isotropic and anisotropic materials. The firsts are invariant to rotations of the surface around n, whereas the second changes its reflection around n, emulating materials such as hair or brushed metal [103]. On the other hand, BRDFs are categorized as phenomenological or empirical, physically-based or theoretical and data-driven. Phenomenological functions are computationally convenient methods for estimating the reflectance appearance of some materials, whereas theoretical make use of some attributes that intervene in physically-based shading. For instance, Phong was one of the first phenomenological models that were widely adopted with the advent of programmable pipelines. It is simply parameterized by the surface normal, n, the lighting vector from a single source, l, the view direction, v, and the reflection, r, computed according to the Fresnel law. On the other hand, Cook-Torrance is one, if not the most, representative theoretical BRDF as it pays attention to micro-facets, distinguished metal and dielectrics and uses the halfway vector, h, to compute the specular lobe. For novel CG programmers, this is frequently introduced as the foundational core of Physically-Based Rendering (PBR) for shader-based frameworks such as OpenGL (Open Graphics Library). For further insight into analytical



Figure 2.13: Insight into micro facets of a surface. The striking energy reflects into the hemisphere formed on the basis of the normal vector. Below, the same object is rendered as a polished and rough surface. Therefore, the second present microfacets that lead to a higher energy scattering.

BRDFs, we refer the reader to the explanations of these from a theoretical point of view [103, 104]. Note that radiance spreads over a hemisphere, and therefore, most of these functions include π in the denominator, whereas the numerator is scaled by the cos function parameterized by ω_i, ω_o and n. However, traditional CG applications avoid downscaling the computed intensity by π since it darkens the rendered scene and they do not intend to behave as physically-based engines. A simple demonstration of Minnaert BRDF is shown in Code 2.2, firstly implemented to render intensity in a hemisphere, whereas the second block is rather aimed at shading a polygonal mesh. A comparison of frequent analytical BRDFs is also established in Figure 2.14.

Listing 2.2: Minnaert BRDF implementation for a 1) reflectance visualization tool using a hemisphere and 2) a rendering application.

```
::begin parameters
float A 1.0 10.0 2.0
float Darkening 0.0 20.0 1.5
bool Normalized 0
::end parameters

::begin reflectanceShader

vec3 BRDF(vec3 L, vec3 N, vec3 V, vec3 X, vec3 Y)
{
    float dotLN = dot(L, N);
    float dotNV = dot(N, V);

    return N * A * pow(dotLN * dotNV, Darkening - 1.0f) / (
    ↪ Normalized ? PI : 1.0f);
}

::end reflectanceShader

::begin renderingShader

vec3 BRDF( vec3 kad, vec3 ks, float shadowDiffuse, float
    ↪ shadowSpecular, vec3 L, vec3 N, vec3 V, vec3 X, vec3
    ↪  Y )
{
    float dotLN = dot(L, N);
    float dotNV = dot(N, V);
    float diffuseFactor = A * pow(dotLN * dotNV, Darkening
    ↪ - 1.0f) / (Normalized ? PI : 1.0f);

    return shadowDiffuse * Id * kad * max((dotLN +
    ↪ materialScattering) / (1 + materialScattering), .0f)
    ↪  * diffuseFactor;
}

::end renderingShader
```



a) Cook-Torrance b) Phong c) Oren-Nayar

Figure 2.14: Comparison of analytical BRDFs: a) Cook-Torrance ($f_0 \leftarrow 1$, $\alpha \leftarrow 0.077$), b) Phong ($ns \leftarrow 32$) and c) Oren-Nayar ($\alpha \leftarrow 0.75$).

2.2 Data types, processing and indexing

Once the imaging devices and the interaction of light and matter have been revised, the results provided by most RS tools are explained, including the main challenges that we must address if these will be the focus of our work. Then, other traditional processing algorithms which are later used are explained. These algorithms are either based on the image space or operate with 3D point clouds.

2.2.1 Results from Remote Sensing

Image: vector and raster

The concept of images seems very straightforward, though the focus of this section is on the adjective preceding the image term. Besides human impression of someone or something, images are defined as "any picture, especially one formed by a mirror or a lens" by Cambridge Dictionary [105] and "a picture of somebody/something seen in a mirror, through a camera, or on a television, computer, phone, etc." in Oxford Dictionary [106]. As a mathematical concept, an image is "a point or set formed by mapping from another point or set", thus requiring **at least** two point sets to form an image. The subdivision of images over $\mathcal{X}$ and $\mathcal{Y}$ axes yields what is known as a pixel: the images' primitive. Despite being unique pieces of information, these are highly correlated among them and thus influenced by surrounding pixels. The significance of this conclusion is later exposed in this document to discriminate the plant variety captured in a pixel.

At least two point sets are required to form an image, though the most frequent images in the wild have three channels (visible imaging) and thus require more than two sets. For RS, images with up to hundreds of channels are required for precise monitoring and data mining.

Hence, images are discrete representations where each pixel shows the average captured radiance. However, these values are not the originally measured radiance. Instead, an Analog-to-Digital (A-to-D) conversion transforms the electrical signal into values known as Digital Numbers (DN). These are typically integer values that fall in the interval 2^x, with $x \leftarrow 8$ being the most frequent radiometric depth [29], though float-point values can also be found for file formats storing uncompressed representations, e.g., TIFF. The radiometric resolution is especially relevant for scientific

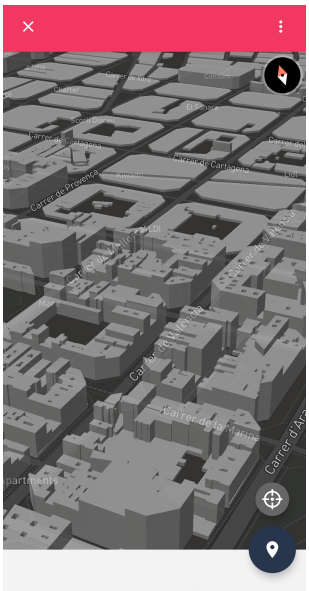


Figure 2.15: Vector data in a geographic information system within an Android application.

purposes where A-to-D leads to a loss of precision. Other not-that-frequent file formats store embedded DNs in the image metadata, such as radiometric JPEG, thus allowing us to obtain the temperature afterwards. Despite images being here referred to results acquired by sensors, the term "image" covers any pictorial representation, even generated on computers. Unlike sensing tools, computers have the advantage of working with digital data whose geometrical shapes can be defined by means of step-wise and continuous functions. As a result, the internal encoding of images does not always fits into the previous mathematical definition. The images that are not defined as a set of pixels, but rather as a set of geometrical shapes, are known as vector images instead of raster images. While vector data have no limitations on the resolution and present a lower memory footprint, raster data is easier to analyze and include in Geographic Information Systems (GIS) together with other layers acquired with similar spatial resolution [3]. Still, vector data is especially relevant for visualization, including GIS aimed at mapping Earth's cities and infrastructure (see Figure 2.15). A comparison of raster and vector data is depicted in Figure 2.16.

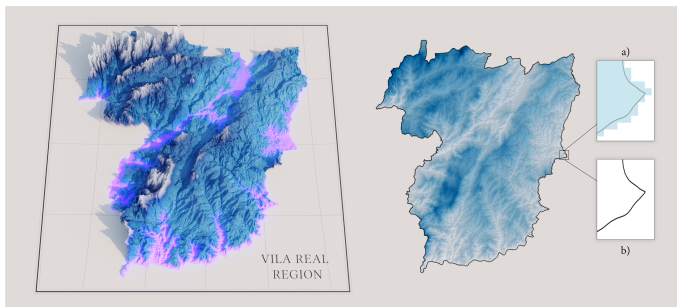


Figure 2.16: Comparison of a) raster and b) vector representation of Vila Real district in Portugal.

The spatial resolution of images refers to the number of pixels represented in $\mathcal{X}$ and $\mathcal{Y}$ axes. However, it can be further stressed to introduce the amount of space in m^2 represented in a pixel. Indeed, the latter concept is more frequent in RS. It is typically derived from the image dimensions and the distance covered in $\mathcal{X}$ and $\mathcal{Y}$.

The most widespread impression of images is colour-space examples. However, there exists a considerable imagery taxonomy organized from the number of bands ($\mathfrak{L}$) and the information that is encoded in each of them. The most revised kinds of imagery are visible (colour-space), thermographic, multispectral and hyperspectral. The sensors that capture this information are coined with the same name. In this regard, Figure 2.17 shows the percentage of research devoted to each one of the mentioned tools in the last few years. Barely addressed tools such as RGB-D, with D standing for depth, are omitted in this figure. Previous sections have delved into each one of these sensors by providing a better insight into their results. The Scopus searches were the following: $(\epsilon \cdot (rgb \vee visible) \vee \epsilon \cdot (thermal \vee thermography) \vee \epsilon \cdot multispectral \vee \epsilon \cdot hyperspectral) \wedge ((remote \wedge sensing) \vee uas \vee uav)$, $\epsilon \in \{0, 1\}$, where ϵ refers to the selection of one of the sensor keywords.

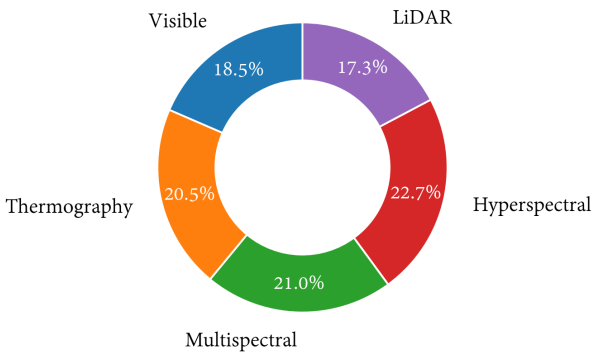


Figure 2.17: Number of manuscripts related to different sensors.

Point clouds

Point clouds are collections of points with no connectivity, which are typically extracted from laser scanners and photogrammetry. There exists an enormous number of hardware and software solutions measuring real-world geometry at a constant pace, including the so-called digital twins. Although images carry relevant features and lead to meaningful conclusions, these are not linked to geometry. Three-dimensional representation of real-world scenarios has enjoyed a great interest to improve visualization and interpretation of the surveyed areas and phenomena. Therefore, 3D point clouds derived from imagery can be generated, if not directly acquired with passive sensing, to facilitate the view of buildings, surfaces and any other area of interest. In contrast to 3D meshes, they enable a simpler, denser and close-to-reality representation [107]. Besides the positioning, points can carry further information, including colour, reflectance, intensity, normal vector and semantic labels, among others. Additionally, all these values can be replicated at different timestamps throughout the time scale.

The staggering pace of information acquisition and requirements concerning the level of detail lead to massive 4D data, thus hardening the efficient storage, transmission and visualization. Yet, point clouds are widely used in fields as disparate as autonomous driving systems, virtual and augmented reality, biomedical applications and precision agriculture. Geometric features provide further data that helps with the recognition and classification of objects, even for human operators. For instance, previous work [108–110] have shown that elevation data significantly improves the classification accuracy on the segmentation of plots, with elevation being a derived product of photogrammetry and passive sensing. However, huge point clouds consisting of billions, even trillions of points, require up-to-date techniques that benefit from nowadays hardware, capable of massively working in parallel. The current state-of-the-art studies have extensively worked on the compression of point clouds, to save storage and reduce bit rate transmission, as well as on the efficient visualization of massive point clouds [111, 112].

Other operators such as the estimation of normal vectors will be following explored as it concerns a later chapter. However, this is simply

a minimal part of research on point clouds, since the uncertainty from photogrammetry and laser point clouds must be individually addressed for each application. Part of this enormous amount of work comprises the classification, segmentation and identification of tree skeletons [113] on point clouds.

2.2.2 Data processing

Compression

The huge amount of point cloud data represents a challenge both for storage and transmission. Furthermore, some of the following operations will be massively parallelized on GPU hardware with limited memory capacity. Although this drawback has been mitigated in the most recent GPUs (NVIDIA GeForce 40 series at the moment this dissertation is being written), these operations ought to work over commodity hardware. The most recent releases have a memory size ranging from 8 GB (GeForce RTX 3050) to 24 GB (GeForce RTX 3090, GeForce RTX 3090 Ti [114] and GeForce RTX 4090 [115]), whereas laptop models have typically their memory size reduced to half of these. Other high-performance GPUs focused on research, for example, NVIDIA A6000 [116], scale up to 48 GB and enable higher capacity using a bridge called NVIDIA NVLink. However, these high-performance products are not widespread, and most computers work using commodity hardware. Accordingly, GeForce Series 10 and 20 are less prohibitive, easier to acquire under stock shortage and more frequent to find in laptops not designed for high-performance tasks. The memory size of these devices is below 12 GB for Series 20 and 8 GB for Series 10.

Furthermore, there exist frameworks for GPU visualization and computing that severely clamps the maximum buffer size. For instance, OpenGL has been the standard platform for visualization for decades. Version 4.3 introduced what is known as computer shaders, i.e., scripts solving small GPU tasks that can be massively parallelized. One would expect to be able to reserve buffers with a maximum size bounded by the GPU's memory capacity; however, this limit can be queried in OpenGL, with the `GL_MAX_SHADER_STORAGE_BLOCK_SIZE` keyword, returning 2,048 MB both for an NVIDIA GTX 1070 (8 GB VRAM) and an NVIDIA RTX A6000 (48 GB VRAM).

Hence, massive point clouds require at some point to be compressed, mainly in transmission and visualization applications. This research field has been coined as PCC, Point Cloud Compression, distinguishing lossy and lossless methods. Similar to the computer science field, PCC has transitioned from traditional techniques [117], based on encoding using data structures (regular grids, octrees, kd-trees, etc. [107]) to Deep Learning, where the encoding is automatically learned by convolutional autoencoders (CNN-AE). The latter are more likely to lose data to some degree unless the compressed data is discrete [118], such as octree nodes.

Very few compression methods offer low-computational complexity for

real-time transmission [107] and reconstruction (decoding), while others do not support the compression of colour information. However, RS point clouds can carry large amounts of information for every point; from LiDAR points whose attributes follow the LAS file standard to hyperspectral points storing spectral signatures with hundreds of measurements. Rather than exploiting the point cloud's entropy in the spatial and temporal dimensions, Graciano et al. [119] proposed to compress layered datasets taking advantage of data redundancy. Although it was proposed to volumetric data from geology, biology or medicine fields, redundancy of sensor measurements can also be exploited for rendering. The algorithm is composed of three main steps: 1) stack redundant values on the z dimension to form the stack-based representation (SBR) from the initial heightfields, 2) group stacks with the same stack content (G-stacks) and 3) organize these into a quad-tree. Spatial redundancy is not as relevant for sensor data since it captures a wide variety of materials, whereas spectral compression is feasible at expense of downscaling the radiometric resolution.

Visualization

Unlike triangle meshes intending to emulate continuous surfaces with underlying discrete polygons, point clouds are discrete representations. Therefore, visualizing the latter is harder for scarce products with low point density (see Figure 2.18). This hardens the detection of features and extraction of valuable information by human operators. Computationally, it implies increasing the neighbourhood distance from which features are extracted or simply obtaining imprecise results due to the lack of data. The traditional rendering pipeline neither helps to accelerate the visualization of massive point clouds. It is designed for polygonal meshes that require 1) processing vertices, 2) including additional geometry, 3) discarding non-visible geometry, 4) rasterizing polygons, and 5) colouring polygon fragments [120].

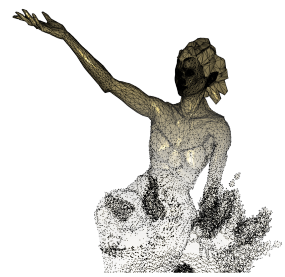


Figure 2.18: Transition from triangle mesh to point cloud representation, from top to bottom, with the second solely showing the triangles' vertices.

The current state-of-the-art of massive point cloud rendering has evolved along with the impressive work of Schütz and Wimmer [121, 122]. The aim of their work was not only to speed up the visualization of billions of points but also to improve the rendered image. OpenGL's compute shaders were used to generate the images in the GPU and recently introduced NVIDIA extensions helped to further optimize it. An alternative rendering procedure is built over OpenGL to omit the stages 2), 3) and 4) of the traditional pipeline. In this work, vertices were massively projected into an image that is later rendered as a texture. However, the order in which points were projected showed to have a big impact on the performance. Points were transformed into Morton codes of 30 bits, 10 for every dimension, and sorted according to them. Then, globally sorted points were shuffled into small batch groups, whose size is determined by Graphic Processing Clusters (GPCs). Rather than distributing the work linearly, the GPU organizes it into 2D patches assigned to GPCs. Thus, global sorting helps in grouping framebuffer updates, but it is suboptimal due to workload unbalancing. Spatially close GPCs in charge of visible points at a timestamp t are assigned

a heavy workload, while others remain idle. Hence, the random shuffling of globally sorted points using small batches has been demonstrated to slightly enhance the rendering performance.

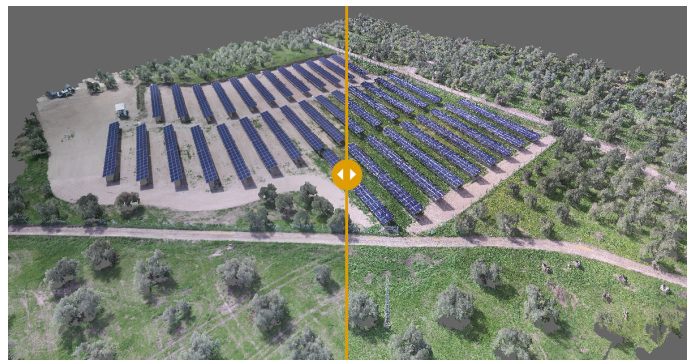


Figure 2.19: On the left part, colouring using a mixed representation of surrounding points and on the right side, using a sole point colour for each pixel. This image was obtained using the procedure described by Schütz and Wimmer [122].

Compute shader logic proposed by Schütz and Wimmer

The projection approach from Schütz and Wimmer [122] uses modern OpenGL extensions following this pipeline:

- The depth buffer is firstly computed, with the depth being encoded along with the point index (64 bits). The atomic minimum operators help to select the point with the minimum depth. However, this approach leads to a noisy-like rendering for sparse point clouds. Instead, the depth of a pixel takes into account several points; data is exchanged within thread groups whose points fall in the same pixel using `shuffleNV` and `ballotThreadNV` operators. Then, `shuffleXorNV` computes the minimum depth within a pixel neighbourhood.
- Colours are iteratively accumulated: the limited capacity of SSBOs may require that multiple point cloud subdivisions aggregate data. Accumulations can be implemented using atomic operators, whereas (red, green) and (blue, alpha (α)) channels are encoded in two different buffers with values of 64 bits. Hence, the alpha channel is also used for accounting for the number of visible points at each pixel.
- Aggregated colours are finally transformed by averaging the mean value for each channel. Pixels with $\alpha \longleftarrow 0$ are shaded with the background colour.

Lately, optimizations of point cloud rendering have focused on the LOD as well as on shaping data into Layered Point Clouds (LPC). A key in the construction of these data structures is that they ought to be view-dependent to enable adjusting the LOD based on the camera position and direction. Schütz et al. [111] proposed to render lower LODs with voxels from an octree, whereas the highest LOD goes back to render points. Ogayar et al. [10] managed to create an adaptive hierarchical structure that coupled different data structures at different levels. It was shown to retrieve larger

amounts of points with higher average and peak throughput. Previously, Schütz et al. [123] sorted points encoded as Morton codes, split them into batches and built a hierarchical data structure using these batches as leaf nodes. Visualization of LODs varies according to the camera, and therefore, patches of different point densities could be rendered for distant objects. If the underlying discrete levels of a LOD-based data structure are visible, they are called rendering artefacts. Van Oosterom et al. [112] proposed to organize point clouds into a Space Filling Curve (SFC) and compute the continuous LOD of every point, thus enabling gradual changes in the point density.

Noise in point cloud visualization was also mitigated by Schütz et al. [121, 122] using modern OpenGL extensions. Individual GPU threads are grouped in subgroups, also known as warps, of size 32 in NVIDIA microarchitectures, that can communicate efficiently and exchange results. For a globally sorted point cloud, warps work over spatially close points that can aggregate colour data. Hence, this approach leads to a High-Quality Shading (HQS) variant [122] that provides a more continuous shading (see Figure 2.19). This also opens the possibility of filling holes derived from lower point density rather than reconstruction failures.

Occlusion detection

The projection of additional sources of information over point clouds is trivial once camera poses are estimated. For each viewpoint, the camera matrix enables the projection of 3D points into the image plane. However, projections do not separate background and foreground surfaces; although a point may be theoretically visible from a camera, it could be covered by another surface in the real scenario. The chances are that points are visible in more than a single viewpoint, and therefore, erroneous assignments of information may be partially masked by other correct measurements. However, this approach is not precise for colouring point clouds and occlusion must be detected to tackle this.



Figure 2.20: Shading of a 3D model and depth obtained from the z-buffer.

The most straightforward way to detect occlusion is to use what is called z-buffers in rendering: images that store for each pixel the minimum depth at which geometry was projected, as depicted in Figure 2.20. It has been traditionally used for shadowing and discarding non-visible geometry for optimizing the rendering [120, 124]. Similarly, it enables recording the nearest visible point at every image's pixel. It does not require indexing points into a data structure, though they are of great help to narrow the subset of points to be projected into a single viewpoint. Determining the nearest depth is trivial in the GPU, as most GPGPU frameworks provide atomic operations. Despite these being supposedly slower operations, they compete well against other approaches in terms of memory consumption and efficiency. Jurado et al. [125] proposed to build out-of-core occlusion-aware multispectral point clouds in Compute Unified Device Architecture (CUDA). Asynchronous data loading and computation were applied to complete z-buffers for hundreds of images and point clouds of up to 1,084M points. Instead of using z-buffers, Jo et al. [53] computed the point and

camera distance to solely project information from the closest viewpoint. This approach leads to errors if the point remains occluded from the closest one, though it can work over very dense and uniform image datasets. Not to mention this colour assignment is more prone to noise, while aggregations mitigate it.

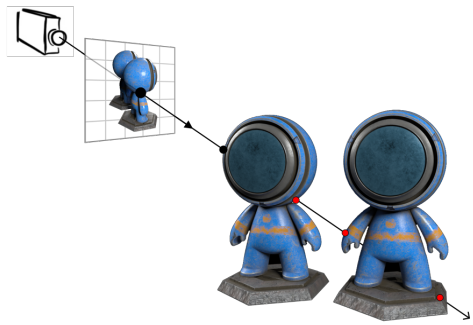


Figure 2.21: Overview of occlusion, with the left model occluding the right one from the perspective of a camera placed on the left side.

An alternative to z-buffers is to estimate the underlying polygon mesh of point clouds. In this regard, occlusion can be detected by constructing a small triangle mesh using the k-nearest neighbours (KNN) of a point [126]. Although this method can work over planar surfaces, polygonal meshes are harder to reconstruct in other scenarios, for example, forestry with dense and incomplete vegetation. Other studies benefit from additional sources of information to detect occlusion, such as the semantic segmentation of point clouds [127].

Photogrammetry

Photogrammetry refers to the science and technology of generating geometrically correct products derived from photographs. 3D reconstructions are fundamental in this dissertation, as they provide an automatic procedure for modelling scenes, in contrast to Computer-Aided Design (CAD) tools, arm-mounted probes and active methods. However, mapping is also derived from what was coined as photogrammetry: the fusion of image datasets to compound results that helps in the understanding and visualization of target areas. Some of the most common maps are Digital Surface Models (DSM), Digital Terrain Models (DTM) and orthophotos. These products are estimated from datasets ranging from terrestrial (Figure 2.22) to UAS-based imagery, which is known to be acquired sequentially in most cases, and large datasets from publicly available data. The latter images are harder to manipulate since one of their major stems is crowd-sourcing from the Internet: data published by users with very different devices, lighting conditions, visible objects in the scene, etc.

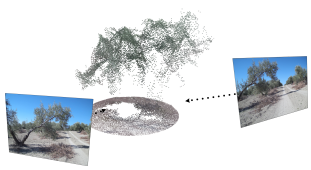


Figure 2.22: Different points of view from terrestrial imagery, which can be later used to estimate a 3D point cloud with photogrammetry.

The transition from laboratory to outdoor was possible due to the advent of digital images with higher resolution, better computational capabilities and mature techniques. The overall procedure in photogrammetry is 1) to estimate parameters from cameras, 2) to reconstruct 3D points and 3) to reconstruct materials if necessary. Two slightly different techniques have

had great success in recent years: Structure from Motion (SfM) and Visual Simultaneous Localization and Mapping (VSLAM), though this section is focused on the first.

The SfM method comprises a vast literature, but it has been robustly applied for over 20 years. The basis consists of 1) detecting key points in images, 2) matching these key points among multiple images, 3) estimating camera parameters from previous links and 4) building SfM models. The final step, despite not being part of SfM, is to reduce errors from estimations, also known as bundle adjustment (BA). The error to be minimized is formally defined as follows (Equation 2.4) [128]:

$$E(P, M) = \sum_{j} \sum_{i \in V(j)} \mid P_i(M^j) - m_i^j \mid^2 \tag{2.4}$$

where P is a set of camera parameters, P_i, M^j represents a set of 3D coordinates of a track, m_i^j are the projections of M^j in the i-th camera, $V(j)$ are the list of cameras where a point M^j is visible and $P_i(M^j)$ is the projection obtained by using the camera parameters P_i.

SfM is also found in the literature as SfM-MVS, with MVS referring to Multi-View Stereo. MVS originated as an improvement to two-view stereo algorithms. Instead, thousands of images are applied to the reconstruction of large metropolitan and natural scenes. However, the main concerns in this regard are the photo consistency of features that are identified in several images. As we previously stated, conditions may vary from one viewpoint to another; hence, it is required to calculate how likely a candidate in V_i is the correct match of a pixel in V_j, with $i \neq j$. Finding candidates for every feature over huge datasets is neither trivial nor efficient. Therefore, most of nowadays photogrammetric software is programmed in the GPU to accelerate the pipeline.

Yet, applying photogrammetry is not prohibitive for small datasets, and it is even possible to find software for mobile devices that can work over rapidly collected imagery. Computer-based solutions, aimed at large datasets, include open-source and commercial software. The first group includes ColMap, AliceVision, Zephyr and VisualSfM, whereas notable commercial software includes Pix4DMapper [129], Agisoft Metashape [40], RealityCapture and Autodesk Recap 360 [130]. The first two commercial solutions are sped-up using multi-core CPU and GPU acceleration for image matching, whereas BA is performed in the CPU. [131] revised the efficiency of SfM implementations by applying it over four datasets. According to their evaluation, performed using commodity hardware, Metashape showed the worst performance both for BA and feature matching, whereas RealityCapture showed similar performance during feature matching and significantly better during BA. However, Agisoft Metashape generated the densest point clouds for every dataset. Also, reported results in the literature depend on the computer used during the experiments and yet, they may vary over a short period of time. For example, Agisoft Metashape is currently known to support accelerated processing in the GPU for image-matching, depth-map reconstruction and other mesh-related operations. It works over CUDA-

enabled GPUs, and therefore, it was not expected to provide results as discouraging as reported by Jiang et al. [131].

Colour aggregation

Points, if back-projected to imagery, are mapped to multiple values that may vary one from another since these are captured under different lighting and viewing conditions. The observed radiance varies according to the viewing angle, expressed through azimuth, $\phi \in [0, 360]$, and zenith, $\delta \in [0, 180]$, angles. Also, surfaces behave so that the emitted radiation is not spread uniformly unless it is a perfect Lambertian generator [33]. Surfaces emit more radiation if $\delta = 0$, i.e., the camera's view vector is $-\hat{n}$, the normal vector of the surface. The naïve and widespread way to solve aggregations is to average them [58, 132], while consensual approaches minimize the distance from the aggregated value to the originals. The latter method has been mainly applied to expert-guided decision-making to aggregate opinions following those operators that minimize the conflict among the experts' points of view.

Instead of averaging samples, multiple aggregation functions can be used together with penalty functions to select the aggregation that differs the less from the experts' opinions. In this context, experts are different camera viewing angles which collect distinct results. In this context, penalty functions compute the similarity of each aggregation and the starting samples [133, 134], which must be maximized (whereas the dissimilarity is minimized). Although penalty functions have been gaining interest in scientific research, they have barely been explored in some contexts such as image processing [135, 136]. Nevertheless, they have obtained notable results on image downscaling and even upscaling.

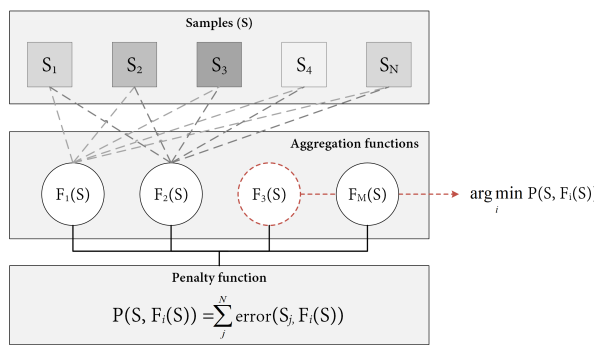


Figure 2.23: Overall procedure to compute the penalty value of aggregation functions.

Briefly, n-ary aggregation functions, $f : [0, 1]^n \rightarrow [0, 1]$ are monotone increasing and satisfy the boundary conditions (Equation 2.5).

$$\begin{aligned} &\forall\ i \in \{1, \ldots, n\},\ x_i \leq y \wedge x_i, y \in \mathbb{R} \\ &\Rightarrow f(x_1, \ldots, x_n) \leq f(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n) \\ &f(x_{min}, \ldots, x_{min}) = x_{min} \\ &f(x_{max}, \ldots, x_{max}) = x_{max} \end{aligned} \tag{2.5}$$

On the other hand, averaging aggregation functions are bounded by the minimum and maximum input (Equation 2.6).

$$\min\{x_1, ..., x_n\} \leq f(x_1, ..., x_n) \leq \max\{x_1, ..., x_n\} \tag{2.6}$$

Finally, penalty functions, $P : \mathbb{R}^2 \rightarrow \mathbb{R}$, satisfy the conditions in Equation 2.7, where ϕ is an aggregated value. Consequently, penalty functions can be expressed as $\sum_{i=1}^{n} P(x_i, \phi)$ with the goal of retrieving the aggregation that minimizes the value of P, $g(x) \leftarrow \text{argmin}_{\phi} \sum_{i=1}^{n} P(x, \phi)$.

$$\begin{aligned} &P(x_i, \phi) = 0 \quad \forall x_i = \phi \\ &P(x_i, \phi) > 0 \quad \forall x_i \neq \phi \\ &P(x_i, \phi) \geq P(x_j, \phi), \ \mid x_i - \phi \mid > \mid x_j - \phi \mid \end{aligned} \tag{2.7}$$

Normal estimation

Point clouds, either derived from photogrammetry or passive sensing, lack certainty in most of the attributes that provide further value to this product, ranging from normal vectors to semantic labels of different levels of detail. Many studies work on solving the uncertainty of these attributes, where normal estimation is a recurrent topic. This operation is provided as a ready-to-use tool in some widespread libraries for point cloud processing, for example, PCL (Point Cloud Library) and CGAL (Computational Geometry Algorithms Library). Since these are very time-consuming tasks, implementations are even facilitated as multi-core solutions that significantly reduce the latency.

Most methods addressing normal estimation operate over neighbourhoods of varying dimensionality that correlate points to form minimal triangle meshes, from which normal vectors are estimated. Thus, the main challenges in this procedure are 1) selecting an appropriate neighbourhood size and 2) being resilient to noise, as these factors greatly contribute to wrong estimations. From the neighbourhood, plane fitting is approached using unweighted Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). Another challenge is the efficiency of the method: seeking the nearest n points is not trivial nor efficient unless data structures with a large memory footprint are constructed from point clouds. Hence, some works have proposed to perform a global sort on the basis of Morton codes calculated from *xyz* coordinates [137]. This way, the neighbourhood is narrowed to spatially close indices on a linear buffer.

With current trends in AI, normal estimation has also been performed with DL techniques. It can be achieved by representing point clouds as graphs [138] and images [139, 140] since the estimation is conducted using points rather than voxels. To this end, additional sources of information such as depth and Hough transform [140] help with the normal estimation along with *xyz* coordinates.

Parallel processing

This section is focused on the revision of approaches for parallelizing processes concerning RS, rather than on specific applications. The key term in this brief section is GPGPU, General-Purpose computing on Graphics Processing Units, which is a programming field that takes advantage of GPUs to rapidly solve parallelizable tasks. This is possible due to the large number of small cores this hardware comprises.

In this regard, OpenGL is the most widespread Computer Graphics framework since it allows both rendering and solving computing tasks using massive geometry. From a rendering approach, point clouds have their own primitive in the draw calls, `GL_POINTS`. It follows the standard rasterization pipeline while processing point clouds for rendering, despite some of these are not required for such a primitive. This drawback can be sorted out with other already explained approaches implementing a custom pipeline [122, 123]. For problems involving parallel computing and rendering, OpenGL is an easy-to-use platform as it allows developers to resolve both tasks at once. However, one of the main shortcomings of GPGPU in OpenGL is the handling of asynchronous tasks that could overlap to get the most out of the GPU's capabilities, which is very limited in OpenGL and constrained to the use of barriers. More recently, OpenGL has been discontinued at the expense of more efficient and adaptive graphics frameworks such as Vulkan, which are expected to turn into the standard frameworks in this field. Since these are still ongoing and remain niche, very few works use these platforms [141].

Another standard framework for GPGPU is CUDA. As opposed to OpenGL, it is not a cross-platform tool; instead, it is only able to operate over NVIDIA GPUs, but it also provides a more flexible GPGPU framework. With OpenGL being discontinued, most GPGPU studies that used OpenGL have transitioned to CUDA over time [111]. Nevertheless, recent NVIDIA extensions have been introduced on both platforms. Other GPU-based rendering and computing platforms are DirectX [142] (Windows OS), OpenCL (cross-platform) and Metal for Apple platforms.

2.2.3 Data indexing

This subsection introduces the reader to a shallow notion of efficient data structures for indexing point clouds and triangle meshes. This is not part of the state-of-the-art of this dissertation; instead, the aim is to understand the data structures that will be following mentioned.

This work is intended to provide scalable yet efficient products to fuse data sources and emulate them. Therefore, data structures used in ray tracing, one of nowadays most challenging CG fields, are a good fit for our later implementations. Ray-tracing is mainly intended for physically-based simulations of energy spreading, with applications ranging from realistic rendering (Figure 2.24) to LiDAR simulations and non-line-of-sight rendering aimed at finding hidden geometry [143]. In spite of the wide

range of applications, geometry indexing in ray tracing has been mainly narrowed to Boundary Volume Hierarchies (BVH) [144] (Figure 2.25). It is a binary tree that helps to rapidly discard parts of the scene during the traversal (half of the remaining scene with each new traversal step). Building a BVH is immediate, though doing it efficiently and making later queries efficient is not as trivial. The way primitives are merged, whether a bottom-up approach is considered, has a significant impact on the traversal. The quality of a BVH, once built, is measured through a cost function that estimates the minimum number of operations to find the nearest primitive. The probability of interior nodes is typically estimated with the Surface Area Heuristic (SAH) shown in Equation 2.8, which correlates the geometry of an interior node and its parent node.



Figure 2.24: Ray-tracing rendering of a 3D modelled room.



Figure 2.25: Rendering of the BVH data structure (left side) for indexing the conference room [145] on the right side.

$$P(N_c|N)^{SAH} = \frac{SA(P(N_c))}{SA(P(N))} \tag{2.8}$$

BVHs can either be constructed with bottom-up, top-bottom and incremental approaches. However, the quality of a BVH cannot be estimated until completion; therefore, the merging/splitting operations are guided by heuristics. Note that, from now on, this section will be driven by the algorithm of Meister and Bittner [146], which is the bottom-up construction followed in this work. The aforementioned heuristics, for instance, can be guided by the area of the bounding box enclosing two nodes for a bottom-up approach. The tighter the bounding box, the better is for later traversals. Note that larger boxes intersect a larger number of rays, thus increasing significantly the number of tests. Sometimes, however, larger leaves are intentionally preferred when the primitive intersection tests are much less time-consuming than the traversal step.

A significant challenge is to efficiently determine which are the surroundings of a primitive. It has been typically approached using what is known as Morton codes (or Z-curve), a data representation that encodes x, y and z coordinates with $\lfloor \frac{2^r}{3} \rfloor$ bits. The most frequent configuration is $r \leftarrow 5$ with 10-bit encoding for every coordinate, though it can be increased to $r \leftarrow 6$ for a higher LOD. This data encoding happens to sort xyz points in 3D following a curve that goes through X, Y and Z axes. Therefore, a 1D buffer of Morton codes can be sorted to rapidly determine which are the neighbours of a primitive using a radius R. It may not return the optimal neighbour primitive for creating a new node, though it is much more efficient. Yet, R can be increased at the expense of higher response time whether a better BVH quality is required (if the heuristic is good enough).

Another shortcoming is to incorporate massive parallelism in the GPU. The BVH construction of Meister and Bittner [146] is guided by heuristics and shortcuts in spatial searches, and yet, it is time-consuming for highly detailed scenes composed of millions of triangles (e.g., a power plant of ~13M triangles has been traditionally used for comparisons [145]). However, binary-tree inspired algorithms are not a *rara avis* in the GPU; indeed, they are very frequently used to cope with simultaneous readings and writings in non-overlapping indices with a complexity of $\mathcal{O}(n\,log(n))$. For instance, the partial summation up to each index was coined as *prefix-sum* or *prefix-scan*, which can be resolved in the GPU with up-sweep (reduce) and down-sweep phases [147]. Similarly, BVH constructions can follow a merge and compact approach where the nearest neighbour is sought, these nodes are merged and then they are compacted in a global buffer. The *prefix scan* algorithm enables the later compaction since it counts how many nodes are prior to any new node, thus determining its location in the global buffer. The response time measured for constructing the BVH in GLSL with models ranging from 345k to 12.7M triangles is depicted in Figure 2.26.

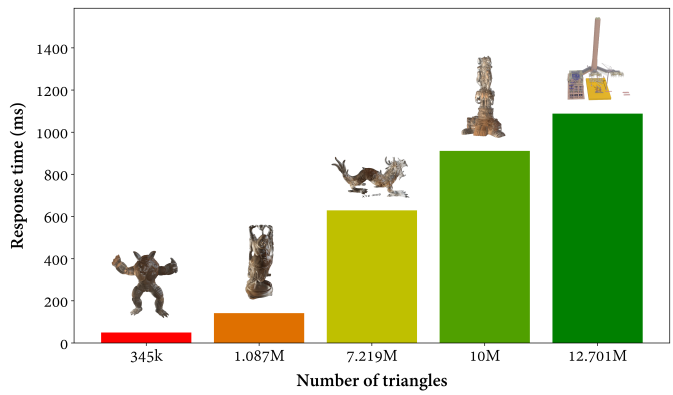


Figure 2.26: Response time in ms required for constructing a BVH for triangle meshes with a wide range of polygons.

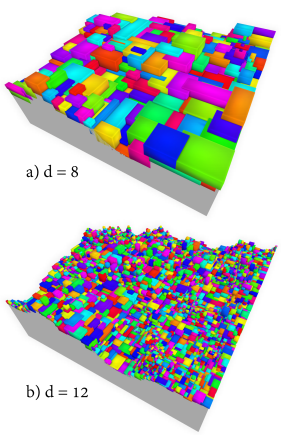


Figure 2.27: Node clustering at different BVH levels.

One would expect the later traversals to be easier than the initial construction. Nonetheless, the first shortcoming is the looping and stack-based traversal; the first makes threads stall until the traversal is terminated, whereas the latter requires storing local access data for every thread. These shortcomings are even harder when the ray-tracing engine does not solely requires the first hit, but also any hit. Stackless traversals have been proposed, but most of them still have a small memory footprint [144]. It is caused by traversals requiring at least some kind of binary encoding of the path. Once footprint and traversal drawbacks are overcome, or at least mitigated to some degree, ray traversals can be further optimized by grouping rays which may have a similar outcome [148]. In this work, ray traversals were approached using a stack with local data. For millions of rays, the GPU will stall for seconds as this procedure is neither efficient nor have a low memory footprint, though it is solved in much less time than would require in the CPU.

Finally, there are other data structures beyond the BVH which are not as fast for ray traversals. Among them, regular grids, kd-trees and octrees are

the most widespread. The first is very fast to construct ($\mathcal{O}(1)$ insertions), although it is not adaptive and queries may lead to performing a large number of ray-polygon intersection tests. It also has a great memory consumption whether a higher LOD is required to lower the number of intersection tests. On the other hand, octrees and kd-trees are adaptive and thus have a lower memory footprint. Subdivisions in octrees are equal splits of a bounding box, whereas kd-trees partitioning relies on the sorting of primitives along X, Y and Z axes. Hence, the former is slightly more efficient on the construction ($\mathcal{O}(n \log n)$), in contrast to the kd-tree ($\mathcal{O}(kn \log n)$). Ray traversals have been long studied for octrees [149], kd-trees [150] and regular grids [151] and do not represent a challenge; however, BVH's bounding boxes are tighter, i.e., the bounding box of the parent node is not equally split among its internal nodes (see Figure 2.27 and 2.28). The latter approach favours that fewer ray-box intersection tests are positive. Nevertheless, note that all these data structures can be constructed so that parallelism is maximized in multi-core environments, as regarded by Karras [152].

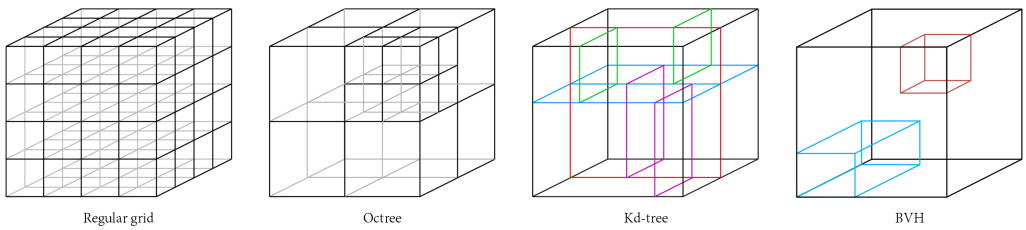


Figure 2.28: From left to right: regular grid, octree, kd-tree and BVH. Part of the image was obtained from Ogayar et al. [10].

Simulation, fusion and analysis of sensor data

3

The previous chapter presented the fundamentals of Remote Sensing which are required for this dissertation, including the notion of images, point clouds and widespread active and passive sensing tools. Different products are the result of these sensors, whereas data ought to be kept in a compact way for subsequent analyses. Hence, pixels from 2D maps, or points from 3D point clouds, are required to have a stack-based representation from each one of the available images. Figure 3.1 summarizes which parts of the spectra are available and which sensors provide this kind of information. Based on previous fundamental concepts, the state-of-the-art concerning the fusion of heterogeneous data, simulation and analysis are following presented.

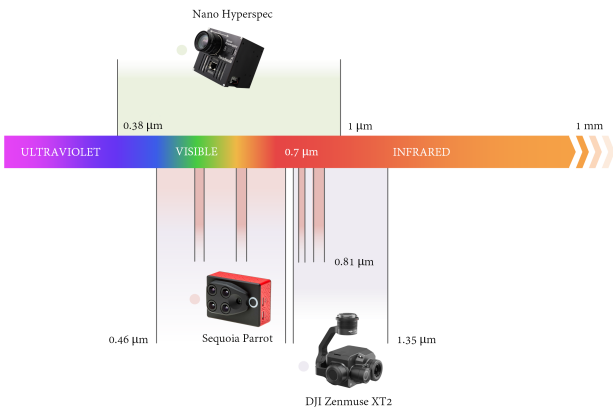


Figure 3.1: Available spectra from three different devices utilized for collecting UAS-based imagery.

This section is structured as follows. First, the fusion of images and point clouds is explored to match pairs of images, point clouds or images and point clouds. Rather than fusing visible imagery, which is the foundational core of photogrammetry, the focus here is on the fusion of visible imagery and other data sources. Then, the augmentation of remotely collected datasets is explored to generate point clouds from metropolitan and rural environments. Finally, the previously fused and generated data is applied to several case studies: the planning of scans in indoor scenarios, the phenotyping of grapevines and the inspection of archaeological sites in seek of buried remains.

3.1 Fusion of information

3.1.1 Motivation

The content of this section is principally obtained from the manuscript **"Remote sensing image fusion on 3D scenarios: A review of applications for agriculture and forestry"**. Although the aim of this publication

was to revise work on the fusion of data for agriculture and forestry, only a few works related to this topic achieved 3D reconstruction with their own pipeline. Instead, most studies coped with 2D and 3D reconstruction using third-party software. Therefore, the domain of application was enlarged and a discussion was established for every revised methodology about the degree of applicability to agriculture and forestry areas.

The volume of remote sensing data is rapidly increasing and many real-world scenarios are currently being replicated by the generation of virtual models in 3D. The multi-source data integration and the 3D representation of surveyed areas is a hot research topic in the field of geoscience and remote sensing and has attracted the attention of both industry and academia. Focusing on natural and environmental areas, it has many prospective applications in smart agriculture [64, 126, 153], forestry and nature preservation [154–157] and monitoring [158]. Initially, the acquisition of remote sensing data required costly sensors mounted on complex platforms. Likewise, the surveying procedure and data processing were labour-intensive and time-consuming. Observed scenarios were mainly represented in 2D maps and orthophotos, after applying manual processes to correct the image distortion [159]. In this field, significant advances have been achieved by the development of efficient methodologies for data acquisition and processing, as well as the production of new sensor capabilities and aerial platforms. Accordingly, extensive research is currently being carried out focusing on multi-source data fusion and image mapping on 3D models.

In this context, the proliferation of UAS as versatile and cost-efficient platforms to acquire multi-source data has enabled the detailed characterization of large scenarios, even with difficult accessibility. Undoubtedly, UAS technology plays an important role in multidisciplinary research benefitting from unprecedented temporal, spatial and spectral resolution, acquired from multiple perspectives in a non-intrusive way.

Although data fusion can be performed over pairs of images, most applications use this as a first step towards approaching 3D reconstructions. From here, primitives from 3D representations, obtained with a single data source, are easily back-projected to another data source. Thus, stack-based representations from image matching also lead to point clouds with multiple layers. Besides the benefits of 3D models in visual inspections, these are significantly more valuable when combined with RGB, multispectral, hyperspectral and thermal data. The fusion of these data and 3D geometry depicting natural and artificial surfaces brings a deep knowledge of our environment. Figure 3.2 offers a vision of how 3D modelling has been gaining interest in data sources beyond the visible spectral range. The Scopus searches were the following: $(p_1 \vee p_2 \ldots \vee p_n) \wedge ((point \wedge cloud) \vee (3D \wedge modelling))$, with p_i being different names that identify a data source (e.g., visible and RGB).

The integration of multiple data sources, in combination with 3D reconstructions, is still challenging since each dataset is obtained with a different mechanism (global or rolling shutter, push broom sensors, etc.), intrinsic and extrinsic parameters, distortion and resolution. Therefore,

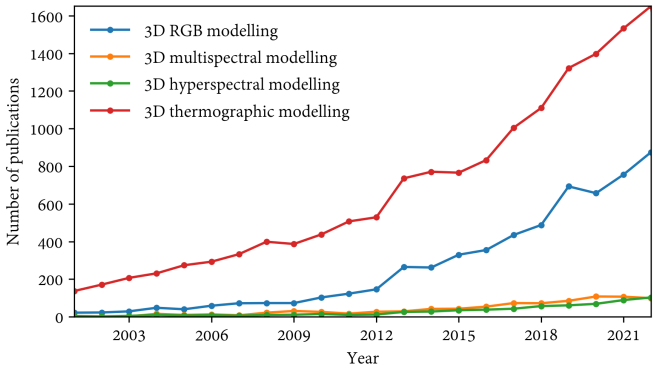


Figure 3.2: Number of publications related to visible, multispectral, hyperspectral and thermographic 3D modelling, from 2000 to the current year.

fusing multiple datasets is not trivial and demands high computational efforts that can be alleviated with GPGPU. In the following subsections, the main contributions to the fusion of visible, thermographic, multispectral and hyperspectral data are summarized. First, the fusion of 2D data is reviewed and then, the combination of imagery and point clouds is revised. Both sections are split into works related to thermal, multispectral and hyperspectral data.

3.1.2 Image matching

Thermal infrared imaging

Consumer-grade IR cameras are not prohibitive, at the expense of having lower resolution and a higher number of defects. Sledz et al. [63] reported several sources of noise causing IR images to have a blurred and smoothed-out appearance. Among these, environmental conditions and non-uniformity of focal plane arrays (FPA) were highlighted [58]. Furthermore, the radiation of a surface is captured by several detector elements, as energy spreads to the surroundings [33]. UAS platforms are less affected by the attenuation from energy dispersion and atmospheric absorption, though this is a well-known problem in thermography [33, 50, 51].

Amongst works addressing matching visible and thermal imagery, only a few consider that both data sources are perfectly overlapped as collected [160, 161]. However, this scenario is not frequent; instead, visible and thermal imagery record slightly different areas of interest, even for images co-acquired by dual devices. This is caused by minor differences in triggering timestamps, the platform's movement and the device calibration, which may degrade over time.

The most frequent approaches for IR image registration are depicted in Figure 3.7. One of the main concerns is the selection of the most appropriate feature descriptor. The reviewed literature shows a higher preference for edge detection algorithms, such as Canny and Sobel operators [132, 162], over conventional feature descriptors, including the Scale-invariant feature transform (SIFT), Speeded-Up Robust Features (SURF) and Oriented FAST

and rotated BRIEF (ORB) (Figure 3.5). In this regard, Hoegner et al. [162] enhanced the edge detection by solely conserving prominent edges with the Hough transform [162]. However, edge detection algorithms are not as appropriate for forestry and rural scenarios, as depicted in Figure 3.3. Previous work has also coped with image matching on the frequency domain, with the aim of handling nonlinear colourimetric differences between visible and IR imagery. Lin et al. [60] implemented the Radiance Invariant Feature Transform (RIFT) [60], followed by a suppression of mismatches with Random Sample Consensus (RANSAC).

From here, projecting IR imagery in 3D reconstructions is trivial; 3D points are efficiently projected into RGB images, which have already been matched with thermography. Figure 3.7 illustrates the image-matching algorithms and how these help in the later 3D reconstructions.

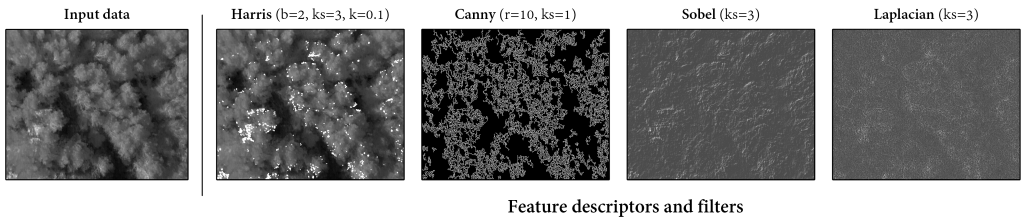


Figure 3.3: Comparison of edge-based feature descriptors frequently used in image registration algorithms. These are less likely to work well over vegetation without relevant features. *ks* refers to kernel size, the ratio is presented as *r*, and *k* is a free parameter on the Harris detector. The brightness of the last two results is increased to improve the visualization.

Instead of using colourimetric data, other studies have proposed a calibrated system with multiple devices. Thermographic imagers are frequently combined with RGB cameras [58, 59, 163] and LiDAR [59, 164]. The calibration of dual-sensor systems is performed by estimating the translation (lever-arm) and rotation (boresight) matrices that represent how these sensors correlate (see Figure 3.4). Calibration is frequently performed by identifying features visible in several images, either following a supervised or semi-supervised approach. However, the calibration of systems with multiple sensors was proven to perform worse in inexpensive, consumer-grade thermal cameras [58]. Constructing a checkerboard calibration panel and marking features on it is particularly tough using IR data [58] since it lacks sharp edges. However, these systems with multiple devices have the advantage of rapidly fusing 2D and 3D data simultaneously once calibrated.

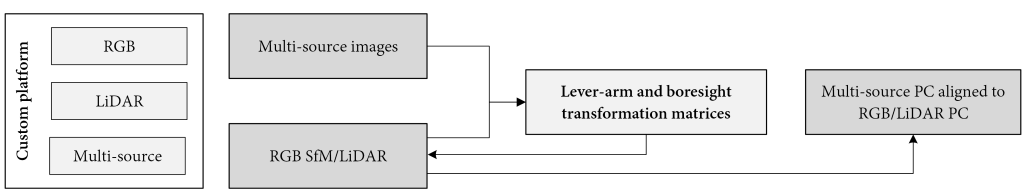


Figure 3.4: Systems with multiple sensors are calibrated with the aim of projecting multiple data sources into a single coordinate system.

Multispectral imaging

Unlike IR, multispectral datasets are composed of several bands in the visible and infrared wavelengths. These bands are more likely to not record the exact same area of interest, influenced by different positioning of lenses, the platform's movement and the calibration of each individual imager, which deteriorates over time. Shen et al. [165] introduced an image-matching solution based on a descriptor measuring the degree of matching for every pixel. Other works used the orientation embedded in the image metadata to correct the misalignment [166], and the error of these parameters was reduced with RANSAC [167]. Similarly to IR image-matching, the SURF [168] and SIFT [169] feature descriptors have also been applied to multispectral datasets. According to Tsai and Lin [170], the SIFT method outperformed other algorithms in terms of quality of results and efficiency.

a) SIFT

b) ORB

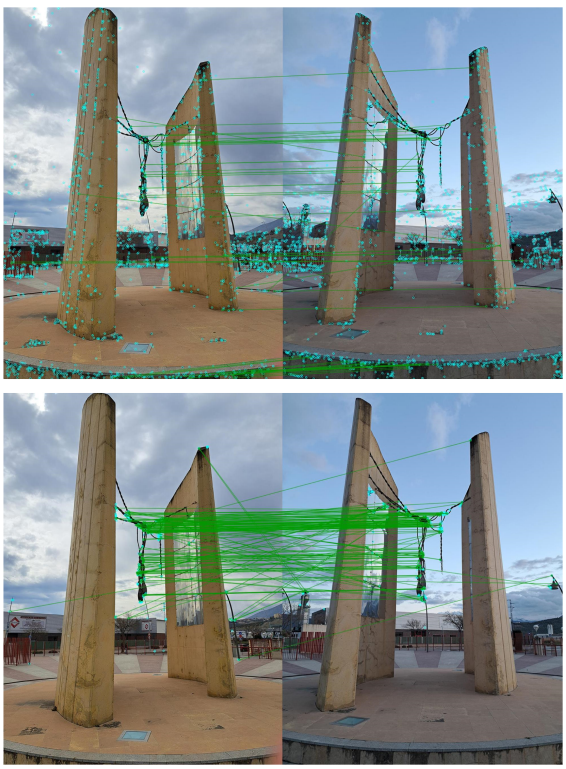


Figure 3.5: Comparison of two traditional feature descriptors and matching algorithms: SIFT and ORB. In this example, both are used to match multi-view sequences of the same scene.

Hyperspectral imaging

Hyperspectral sensors collect the light intensity in a large number of contiguous spectral bands, typically ranging from a few tens to several hundred. Every pixel stores the spectral response of one or multiple materials, thus helping to understand images and monitor scenarios. HSI comprises several scan geometries, already explained in Section 2.1.1, although the vast majority of works are based on the push broom technology (see Figure 2.1). Push broom sensors include a set of 2D detectors that simultaneously

sample all the points along a line. The area of interest is surveyed by translating the platform in the direction orthogonal to such a line. This scanning technology enables the simultaneous acquisition of every wavelength in a pixel, thus facilitating and reducing the post-processing of hypercubes. It is also worth noting that the type of sensor greatly affects field operations, processing performance and the quality of the final product [67].

With this in mind, the main objective of fusing hyperspectral data with other data sources is to correct the geometric distortions, and therefore, the other data source is the reference without geometric errors. In this regard, Jurado et al. [69] proposed a method for matching hyperspectral and high-resolution RGB imagery. Hyperspectral swaths were split into smaller fragments, and the ORB feature descriptor was used to find common features. From here, it was estimated whether the resulting transformation matrix was feasible or not by measuring the distance of control points in hyperspectral (after projection) and RGB imagery. Otherwise, a smaller partition of the swath was used until a stable alignment was achieved or the current fragment could be further split. Previously, Angel et al. [171] aligned hyperspectral and RGB images by applying SURF and discarding mismatches with a variant of RANSAC, Maximum Likelihood Sample Consensus (MLSAC). Also recently, Akhoundi Khezrabad et al. [71] avoided the use of GCPs by simultaneously recording a video together with hyperspectral data. SIFT, RANSAC and Spectral Angle Mapper (SAM) were applied to the geometric correlation. Note that SAM helps to discard mismatches since it measures how similar are two spectral signatures regardless of their scale.

3.1.3 3D reconstructions

Thermographic imaging

The majority of outdoor IR datasets come from UAS platforms, despite terrestrial and indoor collections also being frequent. Lin et al. [60] and Stojcsics et al. [161] collected IR data from a hand-held camera, whereas Zhu et al. [172] recorded both IR and LiDAR from a vehicle coupled with multiple sensors. Adán et al. [173] used a robotic platform to autonomously navigate and scan indoor environments. Previous work also exploited the use of custom systems that are geometrically calibrated by describing the lever-arm between multiple sensors [58, 164]. This calibration is achieved by collecting multiple images depicting a known pattern, typically a checkerboard [58] or landmarks with peculiar reflectivity [59].

In spite of the low resolution of IR imagery, previous work has extensively applied photogrammetry to reconstruct IR point clouds [40, 42, 50, 56, 63, 129, 174–177]. This paradigm is illustrated in Figure 3.6. Among other factors, this approach is very frequent since SfM is part of notable commercial and open-source software and thus is particularly easy to apply. Some of these third-party solutions are Pix4Dmapper [129], Agisoft Metashape [40, 56, 60, 178], Autodesk Recap 360 [130] and Zephyr [179, 180]. Although photogrammetry is trivial to apply from third-party software, IR imagery

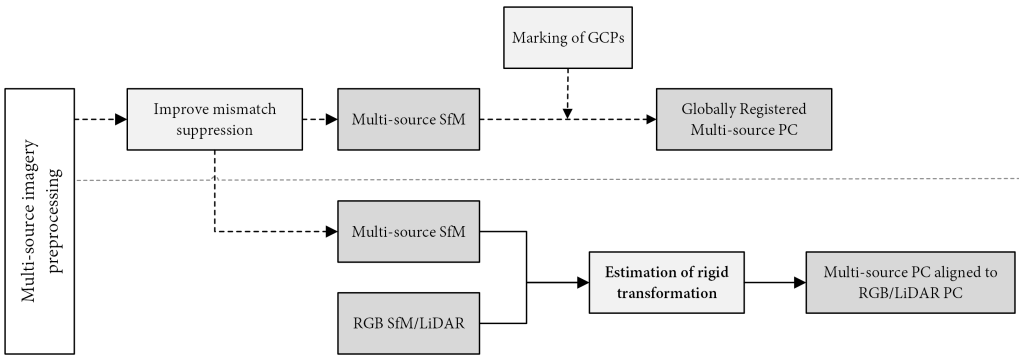


Figure 3.6: Two different approaches for reconstructing 3D IR models. First, SfM is used over visible and IR datasets, whereas the second procedure estimates the rigid transformation that aligns two point clouds.

has defects and limitations that harden this process, for instance, on the retrieval of tie points during feature detection [60]. Consequently, resulting point clouds are more likely to be scarce, less precise and more geometrically inaccurate due to noise, gaps and incorrectly estimated elevation [181]. Photogrammetry is similarly problematic for environments showing repetitive patterns and uniform textures, e.g., buildings and vegetation [36, 60], even for other data sources. On the other hand, geometrical inaccuracies can be mitigated using GCPs. However, the marking of GCPs is more challenging in thermography due to the low resolution and colour contrast [63]. Still, previous work has explored the placement of GCPs without further considerations to reconstruct IR point clouds [50, 63, 129, 174]. Boesch et al. [62] and Nishar et al. [182] overcome the limited visibility of GCPs by using metal-coated materials with low emissivity, in contrast with the surrounding vegetation. In this regard, Figure 3.8 shows the poor visibility of non-metal-coated GCPs when they are surrounded by vegetation (with similar radiation) instead of bare soil.

Another revised approach is to improve some of the stages of SfM with the help of other data sources. For instance, the External Orientation Parameters (EOP) could be estimated from visible datasets that have larger dimensionality and higher accuracy [53]. Maes et al. [183] estimated the IR camera parameters using the Brown-Conrady distortion model. The temperature was corrected by decoupling the influence of air temperature in multiple flights. Also, the image geopositioning was improved with the onboard GPS and Global Navigation Satellite System (GNSS). To overcome the limitations imposed by the SfM's keypoint detection, previous work has explored other algorithms aimed at extracting features and suppressing mismatches. In this regard, Kong et al. [181] applied KNN to remove evident mismatches, followed by a modification of RANSAC where the temperature was used to tell mismatches apart.

Instead of only using IR data, the fusion of several data sources, with one of them being IR, has also been extensively investigated. RGB images are typically the baseline since they help to reconstruct more dense and accurate 3D point clouds. Therefore, the vast majority of previous work

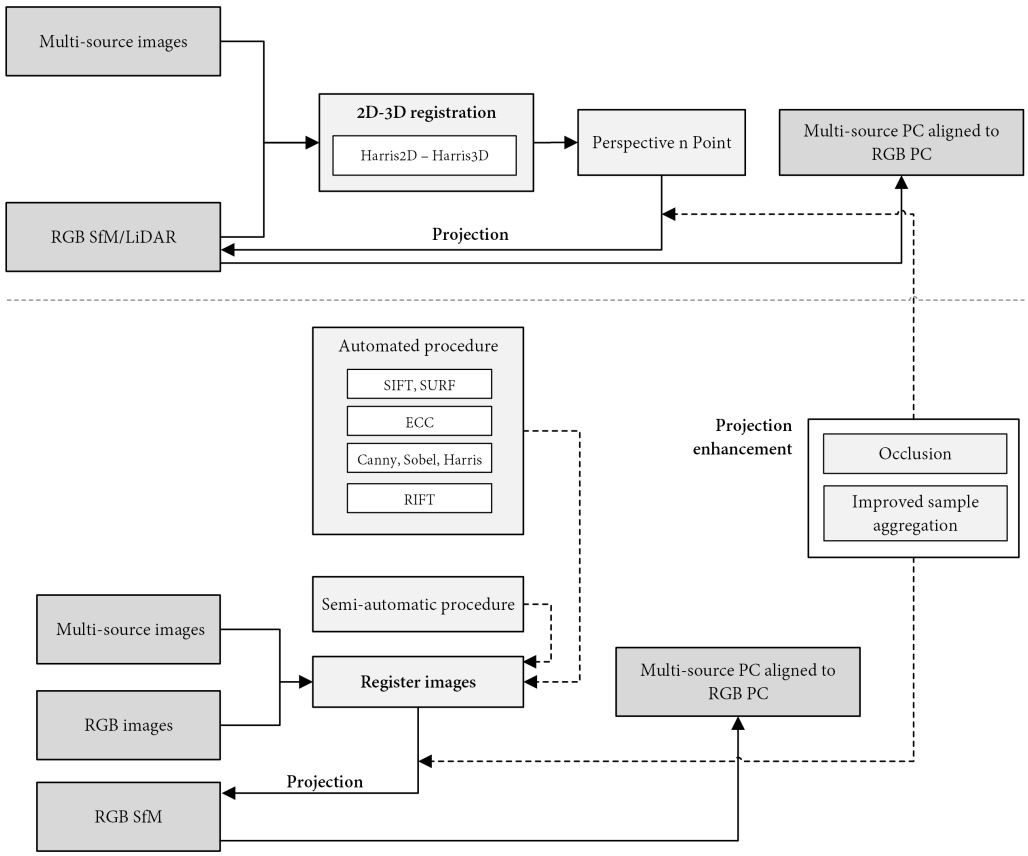


Figure 3.7: Two different methods for reconstructing 3D point clouds. First, images are mapped into a baseline point cloud once 2D and 3D features are correlated and used to estimate pose parameters. In the second approach, features are only sought in 2D to compute the transformation matrices between RGB imagery and other data source.

projects IR data over this baseline point cloud [58, 184]. Visible and thermal image matching has been traditionally performed using 2D and 3D feature descriptors, whereas mismatches are filtered out. Once two or more image datasets are aligned, these can be projected to 3D RGB points. This approach is able to generate larger IR point clouds by upsampling thermal data.

The projection of IR data into RGB point clouds can also be achieved by mixing 3D products and thermal imagery by means of 3D feature descriptors. Following this approach, the Perspective n Point (PnP) optimization has been solved by pairing 3D and 2D features, thus estimating the camera poses (see Figure 3.7). Most of these features are extracted with edge-based operators, from which SIFT and SIFT3D stand out. However, these have been reported to identify insufficient key points and have been often exchanged by the Harris and Harris3D operators [172, 185]. These are more robust and extract features that are less likely to appear in other locations within a single image. Instead of using the bare contours, Lin et al. [60] used the edge intersections as key points. However, most of these methods for matching 2D and 3D features are not robust enough to work under most scenarios.

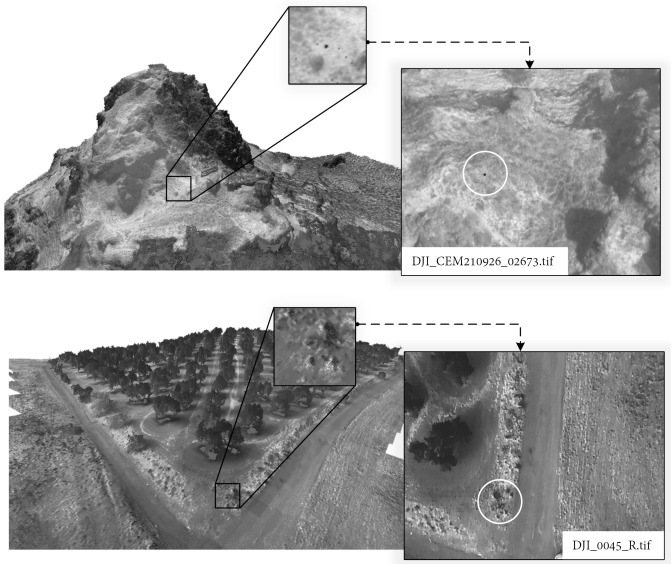


Figure 3.8: Non-metal-coated GCPs made of plastic under two different scenarios. GCPs are barely noticeable in the second scenario since the surroundings have a similar temperature. In the first scenario, they are significantly more visible to the naked human eye since the surroundings are bare soil.

Therefore, the feature matching has also been aided by human operators [172, 185] or by restricting the search space with GPS/GNSS data, with RANSAC helping to discard mismatches [60].

Reconstructing thermal point clouds and aligning them with RGB products is also tempting in spite of the differences concerning density and noise. The distance between both point clouds can be described with a transformation matrix, whose error can be minimized with the Iterative Closest Point (ICP). It returns a composite matrix involving translation, rotation, and optionally, scaling [42, 177, 180]. Rather than solely applying the ICP algorithm, the noise is firstly filtered out, followed by a coarse global registration with a rigid body transformation estimated using the camera poses [186]. Still, ICP can operate as a local registration that provides more fine-grained results. An alternative to ICP is Fast Global Registration (FGR), which is proven to achieve better results with noisy datasets [60]. Point clouds can also be registered by means of at least three GCPs if they can be accurately marked [175]. In addition, the use of ICP along with KNN has been investigated to assign IR data to 3D RGB points, with the outcome being a dense 3D model with upsampled thermal data. More recently, a rotation invariant descriptor that extracts features from icosahedral groups has been investigated [187].

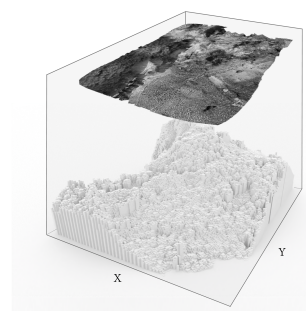


Figure 3.9: Projection of an orthomosaic over a 2.5D point cloud.

IR 3D models can also be constructed by combining LiDAR and visible point clouds with thermal maps in the same coordinate system (Figure 3.10). Comba et al. [57] fused an RGB point cloud with a thermal orthomosaic (Figure 3.9), whereas Adán et al. [173] combined LiDAR results with 360-degree maps of IR data. These studies generate a single thermal image with a limited resolution which is subsequently projected in a 3D model. The former approach obtains IR heightfields (2.5D) [188], whereas the main drawback of the work of Adán et al. [173] is that some points lack IR data

as a result of foreground surfaces occluding background surfaces. This problem was mitigated by using a mobile scanning platform and joining multiple thermal point clouds using the platform odometry and the ICP algorithm.

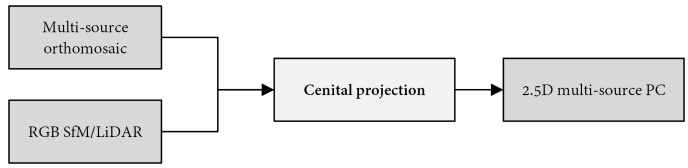


Figure 3.10: 2.5D point clouds are generated by projecting images from any data source into a baseline point cloud.

Finally, supervised procedures have also been proposed to match RGB and thermal features. Huang et al. and Macher et al. [178, 189] used checkerboard images and relevant key points, such as building corners, to link features. Lafi et al. [130] performed semi-automatic stitching of IR images by marking relevant features, thus generating a panoramic image that was finally mapped to a 3D RGB point cloud. However, semi-automatic registration is time-consuming and hard to apply over large datasets. Indeed, this procedure is more appropriate for datasets composed of a few images.

3.1.4 Multispectral imaging

The previous work focused on reconstructing 3D point clouds from multispectral imagery is coarsely split according to the use of satellite and UAS platforms. Traditionally, most of the previous work is based on satellite image mapping over 3D photogrammetric point clouds and LiDAR models for monitoring the forest structure [190, 191], multi-temporal observation of crops [192, 193] and semantic segmentation of urban and natural scenarios [194, 195]. State-of-the-art reconstruction methods from satellite data typically generate elevation data. DSMs can be efficiently obtained with automatic image matching from multiple images acquired in the same orbit [196, 197]. Another promising field is the generation of super-resolution satellite images obtained with DL [198, 199]. Consequently, the cited advances and modern high-resolution satellite sensors enable recovering 3D surfaces from multi-view satellite panchromatic images [200, 201]. The resulting 3D point clouds can be directly fused with multispectral bands of satellite imagery.

Multispectral images acquired from UAS platforms have lower resolution than the majority of RGB images, and yet, it is higher than the resolution provided by most IR sensors. Therefore, 3D multispectral point clouds can be directly generated with photogrammetry [202–206]. Recently, Jurado et al. [126] proposed a novel pipeline to generate dense multispectral point clouds by mapping images on dense RGB point clouds. Firstly, a sparse NIR point cloud was reconstructed and aligned with the RGB point cloud. The rest of the cameras were following projected and multispectral data was assigned to 3D RGB points using KNN. Other approaches are based on multispectral data projection over 3D point clouds from RGB-D sensors embedded on aerial and terrestrial platforms. Clamens et al. [207] implemented a method based on feature-based and corner-based registration of

RGB-D and multispectral images.

The integration of LiDAR and multispectral sensors is another widely adopted approach. In recent years, the proliferation of lightweight LiDAR sensors enables fusing data from a multispectral camera and LiDAR sensor coupled to a UAS. Both sensors have usually the same GPS and IMU feed to avoid inconsistencies in the recorded time and space. Data fusion can be carried out by back-projecting data, i.e., rendering LiDAR results from the optical camera's perspective in order to correlate pixels and returns [208]. Then, the pixel attributes are fetched and the LiDAR returns are shaded. This projection is developed considering the optical sensor system (internal parameters) and the platform's position and bearing (external parameters). Sankey et al. [209] proposed a novel methodology for UAS-based multispectral, hyperspectral and LiDAR data fusion in shrub-encroached desert grassland. However, this study was based on a comparison between 2D multispectral and hyperspectral orthomosaics generated by Pix4Dmapper and SpectralView software, respectively. Gu et al. [210] integrated multispectral and LiDAR sensors in a single system that simultaneously collected data.

Photogrammetry can also help to generate orthomosaics of multispectral imagery, which can be later projected into RGB point clouds [206]. This procedure involves an orthogonal projection of multispectral orthomosaics to obtain 2.5D point clouds (see Figure 3.9). The main shortcomings of this approach are the geometric and radiometric inaccuracies obtained after computing an orthomosaic.

3.1.5 Hyperspectral imaging

The challenges of fusing HSI and 3D models arise due to both data dimensionality and the required storage capacity. 3D reconstructions are easier to achieve with single-snapshot sensors since they can be generated using photogrammetry. However, push broom sensors pose several challenges due to the geometrical inaccuracies and the stitching of multiple swaths. This was already addressed in the fusion of 2D hyperspectral swaths, and it represents one of the first steps before fusing them with photogrammetric point clouds.

The reconstruction of 3D hyperspectral point clouds has been previously achieved by fusing LiDAR and HSI [80, 211]. Otherwise, it has been mainly addressed in close-range applications. For instance, Zia et al. [212] developed a method that applies SfM to images from every collected wavelength and then, the wavelength-wise reconstructions were aligned. Kim et al. [213] described a system prototype composed of a laser sensor and a hyperspectral imager to characterize solid objects. Behmann et al. [214] proposed a method to generate 3D models from push broom sensors in close-range observations. This work used a polynomial transformation to describe the non-linear effects occurring in plant phenotyping, considering not only the linear model of the push broom sensors but also other distortion factors. The fusion of HSI with models from laser scanners has also been addressed

[215]. Liu et al. [216] published an in-depth review of HSI and 3D technologies for plant phenotyping by revising literature ranging from close-range applications to remote sensing.

However, methodologies operating over measurements collected in the laboratory have a more controlled setup and stability in the sensor orientation. In contrast, these conditions are not guaranteed in outdoor scenarios with constantly changing conditions. For instance, the wind induces changes on the sensor and monitored scenes [217]. Other works were capable of merging hyperspectral data in 3D models generated from standard RGB cameras or laser scanning, thus enabling 3D geological modelling [218] and 3D mapping of underwater environments [219].

Despite small-sized UAS enabling data acquisition with higher spatiotemporal resolution [220], the integration of hyperspectral sensors is restricted due to payload capacity and flight autonomy [221]. These conditions harden their applicability to the monitoring of small-scale areas and research work. Even so, 3D hyperspectral modelling from UAS platforms is a recent technology that has become available to small-sized UASs during the last decade [222]. Some approaches use non-imager spectrometers in UAS platforms along with an RGB sensor to accurately display the acquired spectral signatures over 3D information and orthomosaics acquired with RGB data and photogrammetry [223]. For instance, Astor et al. [224] combined UAS-based RGB data and ground-based HSI for biomass estimation in different vegetable crops.

HSI missions collect data by scanning and therefore, only a few perspectives of the area of interest are acquired in the flight direction, thus hardening the reconstruction of 3D scenarios. There is a void of methods accounting for this spatial heterogeneity to correctly merge hyperspectral data from push broom sensors into 3D models. With this approach, it would be possible to have a seamless data fusion integration that would lead to a very high spatial and spectral resolution. It can be expected that in the near future, UAS-based hyperspectral sensors will be capable of acquiring data in other parts of the EM spectrum, such as SWIR, and even support on-board real-time processing [225, 226].

3.2 Simulation of Remote Sensing data

Remote sensing imagers were conceived as mapping tools for generating products that help in the modelling and understanding of Earth's processes. However, the uprising combination of Remote sensing and AI requires large datasets that enable trainable models to seek patterns in various scene perception tasks such as semantic segmentation, instance segmentation and classification applied over indoor and outdoor sceneries. Nonetheless, publicly available datasets may not suffix specific application requirements in terms of data density, attributes, kinds of scenes, recording platforms, etc. Amongst the most common examples of this is the advent of autonomous driving, which required large and variate annotated LiDAR datasets to achieve safe systems capable of working under most conditions. Similarly,

the semantic segmentation of LiDAR has drawn the attention of both industry and academia. Multiple real-world LiDAR datasets have been published over the last years, most of them focused on urban scenarios recorded with static or mobile platforms. Still, some of them are not labelled, do not include trajectories or GPS/IMU data, lack supplementary cameras, etc. [227].

The generation of synthetic LiDAR data is among the most frequent computerized processes. However, the synthetic modelling of images has also been gaining interest [228, 229], especially for those spectral ranges which are not as widespread in publicly available datasets. For example, this is the case of infrared imagery, which is also influenced by the increasing demand for autonomous systems. Unlike visible cameras, infrared cameras do not rely on external light sources and therefore are appropriate for making autonomous systems understand their context even at night and in adverse daylight conditions. Among its applications, infrared may be required for reading traffic signals, wildlife and human detection and object tracking [33].

3.2.1 Motivation of synthetic LiDAR simulation

LiDAR sensors are expensive imagers that require mobile or static platforms to generate a point cloud of the target scenario. To this end, one or multiple scans can be collected and joined by means of overlapping features. On the other hand, LiDAR simulators are much more cost- and time-efficient once the emulation challenge has been resolved, which is neither trivial nor efficient to address. Simulating data avoids carrying out fieldwork to generate massive datasets and enables optimizing the planning of real LiDAR scans [230, 231]. Instead of performing scans on real-world scenarios, simulations are performed over synthetic models that can be annotated with any LOD. Thus, these can feed the resulting point cloud with ground-truth data concerning a large number of attributes, from normal vectors to semantic annotations.

There are multiple applications of LiDAR simulations, along with their corresponding research niches. From a hardware perspective, some simulators are focused on the design, validation and calibration of LiDAR sensors [232]. Their objective is to detect and reduce errors in the scanning, both systematic and random. There are also simulators focused on the optimization of scanning processes [233, 234], such as the integration of LiDAR data into enhanced and synthetic vision systems [235]. From a software perspective, one of the most popular research topics is autonomous driving [236, 237] and navigation [238]. Most of these methods, especially those based on DL, benefit significantly from synthetic data.

Beyond generic research purposes, including teaching and learning, there are systems aimed at the custom configuration of the simulated sensors, including their mounting platform and setup. These simulators typically allow the user to fine-tune all the parameters needed for a physically-based simulation of the laser beams and the interaction with the virtual

environment, where photon mapping, Monte Carlo simulation, and full LiDAR waveform generation are their core foundations [239–241]. These frameworks can simulate multiple scattering of the laser beams, taking into account their propagation through wind-driven rough air-water interface [240], or vegetation [234, 239]. There also exist commercial and open-source software such as LGSVL (LG) [242], Simcenter (Siemens Software) [243], and CARLA [244] for generating autonomous driving datasets.

3.2.2 Public LiDAR datasets

The vast majority of LiDAR datasets focus on autonomous driving and perception tasks such as classification, semantic segmentation, instance segmentation and tracking of mobile objects [245], using geometrical [246] and intensity information [247]. Most of them record metropolitan environments from a mobile vehicle coupled with a LiDAR, along with visible and infrared cameras in a few case studies [248]. On the other hand, aerial LiDAR datasets covering large portions of the Earth's surface are frequently published by governmental institutions at land inventorying portals. Since these are intended to cover large areas, they are acquired from mid-altitude platforms with a density of a few points per squared meter (up to two points according to the United States Geological Survey [249] and 0.5 points for the Spanish National Plan of Land Observation [250]). More recently, despite not being frequent, large and denser aerial LiDAR datasets have also been published [251].

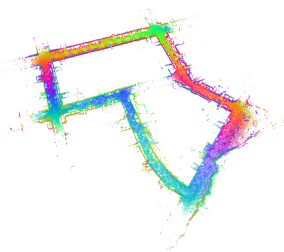


Figure 3.11: Example of a metropolitan LiDAR point cloud in the SemanticKITTI dataset [246], composed of a sequence of scans recorded from a vehicle coupled with a Velodyne LiDAR.

Both kinds of datasets, terrestrial and aerial, are required to be annotated to be applied to DL. Previously revised datasets are recorded by real sensors, and therefore, their classification must be performed manually [246, 247, 252] or automatized with networks trained with very limited data [253]. Manual labelling induces errors, but another shortcoming is the lack of detail. For example, these are some real datasets and the number of different semantic labels: SemanticKITTI (25) [246], nuScenes (23) [254], SemanticPOSS (14) [252], Toronto-3D (8) [247] and Semantic3D (8) [255]. More recently, LiDAR datasets have been released together with video and audio data [256] (see Figure 3.12). A more complete compilation of open-source real LiDAR datasets can be found in [227].

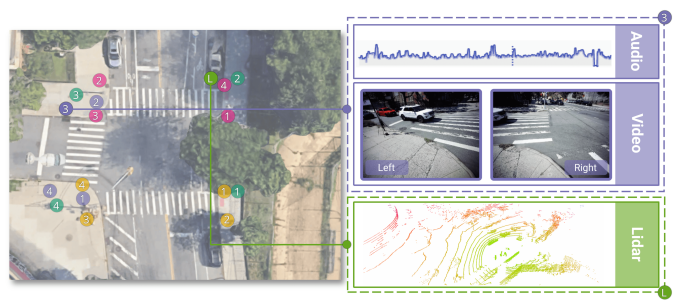


Figure 3.12: Overview of the data collection proposed by Piadyk et al. [256], where LiDAR is captured along with audio and video feed.

3.2.3 LiDAR simulation

The early LiDAR simulators produced graphically feasible LiDAR point clouds [257] by emulating the sensor's FOV, occlusion and beam divergence. LiDAR returns can be solved with ray-casting [93, 258, 259] in annotated environments. Remark that ray-casting implies solving impacts of laser beams in the followed path, frequently ranging from one to five returns, whereas ray-tracing refers to emulating light interaction among surrounding surfaces. The latter represents a physically-based simulation that requires an enormous computation effort and therefore is not yet practical [258]. In recent years, LiDAR simulators have addressed the realistic simulation of geometry returns, the impact of surface properties as well as other environmental conditions.

Some of the aspects of what is expected from physically-based results have been simulated by including a wide number of errors and limitations. First, Xhao et al. [259] modelled atmospheric effects such as rain, snow and fog by including signal attenuation and noise. Haider et al. [260] have recently modelled the signal processing of LiDAR as well as scan patterns to emulate optical losses and defects. Among the cited features, signal attenuation [93, 244, 261], beam divergence [93, 259–261] and drop-off in intensity [258] are frequent in the literature. Very few address multiple returns [94] despite simulating beam divergence. Other not-that-frequent errors are mirror reflection [262], motion distortion [263], non-uniform density and distortion of point attributes (colour, intensity, etc.), either as a result of an inappropriate setup [264], or systematic and random errors [265–267]. Random errors depend on aspects like the signal-to-noise ratio of the received signal, the accuracy of the electronics (including again the IMU and the GPS), the divergence of the laser beams (jittering) and their wavelength, as well as the reflectivity of the objects. Instead of physically emulating these features, previous work has deviated ideal synthetic LiDAR point clouds with DL [238, 268, 269]. The leveraging of real LiDAR and synthetic point clouds has also been investigated by mixing scanned urban scenarios and synthetic objects [238]. Nonetheless, annotation and ground-truth shortcoming remains when leveraging both kinds of data.

Most of the revised simulators are focused on Terrestrial Laser Scanning (TLS), while Airborne Laser Scanning (ALS) has barely been addressed [94]. Westling et al. [234] included ALS as an unrealistic extension of TLS. Bechtold and Höfle [93] simulated ALS with different scanning geometries (Figure 3.14), which is one of the main challenges of ALS. More recently, Winiwater et al. [94] extended the work of Bechtold and Höfle to incorporate full-waveform LiDAR (Figure 3.13). Rather than capturing discrete returns, this kind of sensor returns the intensity recorded over a path. For instance, it helps to measure the amount of vegetation at different altitudes for forest inventory. One of the main inconveniences of this solution, published under the name of HELIOS++, is that input environments are simplified by means of voxels and therefore lose part of the scene details.

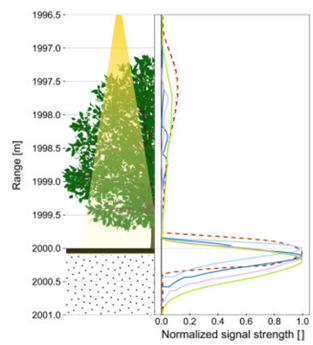


Figure 3.13: Simulation of a full-waveform LiDAR traversing a tree [94].

On the other hand, the modelling of materials typically estimates the reflectance using phenomenological BRDFs such as Oren-Nayar, Lambertian

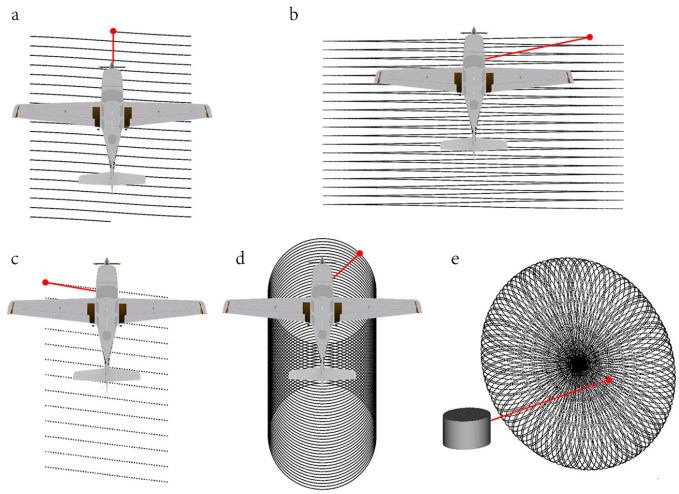


Figure 3.14: Overview of different scanning geometries from the HELIOS++ simulator: a) rotating mirror, b) oscillating mirror, c) fibre-optic line scanner, d) slanted rotating mirror (Palmer scanner), and e) Risley prisms (i.e., Livox scanner).

and Blinn-Phong [263], regardless of the sensor wavelength [241, 257, 263]. Other studies use diffuse, specular and transmissive properties acquired from commercial frameworks, such as CarMaker [260]. Intensity has also been simulated using DL with LiDAR data represented as images instead of points [268, 270]. However, these networks are constrained to the variety of materials and lighting conditions in the training datasets.

3.2.4 Time efficiency

Most studies do not evaluate the efficiency of the proposed solution, and DL networks are constrained to the response time of the network. To the best of our knowledge, only Peinecke et al. [235] have considered a GPU-based simulation in OpenGL. Also, the efficient semantic labelling and generation of synthetic environments have been poorly addressed, including the annotation of virtual scenes composed of CAD models. Another key factor is the precision of the synthetic scans: previous work has simulated LiDAR scans using depth buffers [236, 238, 271] with a limited resolution, frequently 2^8 unless the target texture is composed of floating-point values.

3.3 Optimization of LiDAR scans in indoor environments

3.3.1 Motivation

LiDAR is widespread in the construction industry for the tracking of building progress by enabling the acquisition of the environment geometry in a precise and highly detailed way. Instead of polygonal meshes, a discretized representation of buildings is given by point clouds. To achieve this goal, TLS is increasingly being used to collect large sets of building data [267]. This technology can be applied to a wide range of applications, including

building inspections [85], monitoring of natural environments (landform dynamics [86], ecological resilience [87], etc.), autonomous driving [88] or preservation of cultural heritage [89, 272, 273], among others. Besides terrestrial scanners, LiDAR technology presents multiple variants according to their capabilities (range, spatial resolution and covering, etc.) and the platform from which they are operated (ALS, TLS, MLS, BMMLS, etc.) [274, 275].

Some of the main challenges of TLS in the surveying of 3D facilities are the occlusion and range limitations [276]. Consequently, appropriate planning of TLS scans is necessary in order to 1) minimize the number of different acquisition points, 2) generate a uniformly dense point cloud, and 3) reduce the occlusion from scene objects. These three objectives are equally influenced by the configuration and placement of the scanner. On the other hand, periodic scanning and monitoring of buildings are especially relevant for digitized representations, such as the widely known Building Information Modelling (BIM) [277]. They encode characteristics of a building, including 3D design drawings, materials, costs and safety specifications [278], and provide an interface for the management of 4D applications. Together with TLS, it allows the monitoring of continuously evolving buildings to preserve cultural heritage, track its current state and maintain repair records [89, 273, 279, 280]. However, the monitoring of buildings over time is time-consuming, especially in dynamic environments. Also, TLS surveys generate multiple point clouds that need to be fused either by placing target marks [281] or by estimating the rigid transformation that minimizes the distance among overlapping point clouds. In order to speed up this acquisition task, the development of tools for the planning of TLS surveys plays a key role.

3.3.2 Planning for Scanning

The Planning for Scanning (P4S) has previously been addressed using a wide range of environments and techniques. If the number of target locations is known, then the selection of an optimum set is also known as the NP-complete set-coverage problem [282–284]. Previous research can be categorized according to the input environment: model-based or non-model-based (discovered as being explored). The latter is mainly applied to robotic applications whose environment is unknown [285]. Otherwise, input scenarios are either defined as 2D or 3D models, with 2D representations being cross-sections of buildings [286]. Solutions for 2D inputs present lower computational complexity and are frequently solved following an iterative selection of viewpoints (Next Best View; NBV) or heuristic algorithms [287]. The main drawback of these algorithms is that they do not consider the details underneath complex buildings. Instead, they are focused on 2D sketches composed of wall edges. 3D-based approaches are far more complex and guided by metrics that allow filtering and ordering spatial locations, as in Figure 3.15. Despite this, the exploration of 3D buildings is also approached with heuristics and NBV. However, the high latency frequently leads to simplifications of the environment using voxelizations

[288, 289], axis-aligned bounding boxes (AABB) and object-oriented bounding boxes (OOBB) [231]. A similar work to ours was published at the same time, also partly implemented in GPU [289]. However, the environment was voxelized and the number of optical locations was set by users, instead of being part of the optimization. On the other hand, Chen et al. [290] solved the P4S optimization over 3D scanned façades using a Greedy algorithm. The weight of each location was estimated from a LiDAR simulation. Finally, 2.5D models, such as DSMs, have also been investigated in a similar way to 2D environments [291].

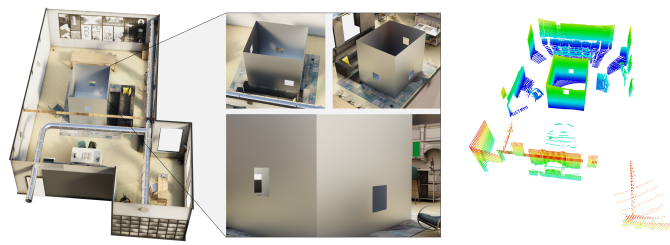


Figure 3.15: Example of 3D model for experimentation of P4S. The main scene has another inner room which some gaps which serve to stress the planning for scanning algorithm. It helps to show how algorithms perform in terms of LOD, LOA and LOC.

3.3.3 Metaheuristics

Heuristics are the most frequent solver in P4S. Although they do not provide optimal solutions, they are proven good enough to cover environments with minimum scanning locations. Among heuristics, the Greedy algorithm has been extensively investigated [286, 292, 293], followed by Simulated Annealing (SA) [294], Genetic Algorithms (GA) [295], Particle Swarm Optimization [295] and Integer Programming [288]. Greedy algorithms are based on the iterative selection of locations, according to an objective function. Other Greedy variations weight the locations using a visibility score [295], while others are followed by SA [296], or optimized with Divide and Conquer (D&C) [292]. Instead of providing an automatic pipeline, [297] proposed an interactive semi-automatic system.

Heuristic solvers are guided by objective functions measuring the quality of achieved solutions. To evaluate this, four metrics are frequently used in previous work [231, 287]: Level of Detail (LOD), Level of Accuracy (LOA), Level of Overlap (LOO) and Level of Coverage (LOC). LOD refers to the point cloud resolution, LOA measures the quality of LiDAR returns, as higher distances and angles deteriorate the quality of the measurements [287], LOO refers to the overlapping area among point clouds so that TLS scans can be joined with rigid transformation estimations (e.g., ICP), and LOC refers to the number of polygons reached by scans. Thus, the optimization is very influenced by the LiDAR range and incidence angles.

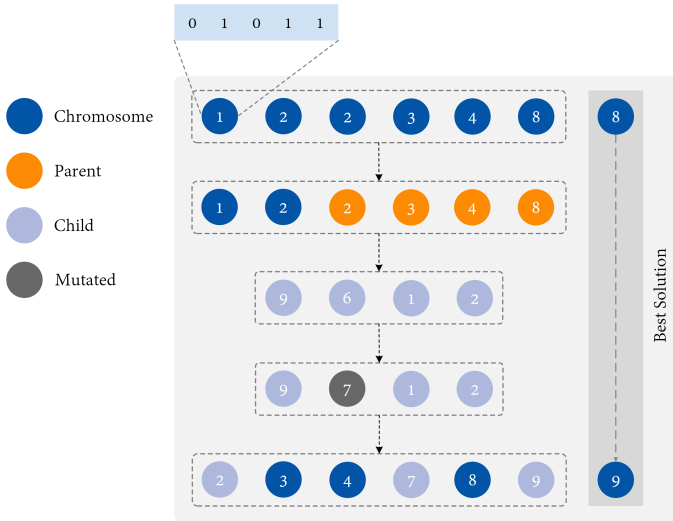


Figure 3.16: Overview of the iterative process and data encoding in a Genetic Algorithm (GA).

Research on heuristics in 3D is also frequent in the literature, though they are mostly linked to path planning [298, 299] (see Figure 3.17) and the selection of subsets [298]. However, the number of scans needed to meet the required quality is unknown in a non-finite 3D space. The set of possible solutions can be narrowed either by selecting random locations [294] or sampling the environment as a 2D grid [286, 291, 295]. For uniform subdivisions, the level of detail of the tessellation is a key factor with regard to response time. Coarse subdivisions present lower latency, though they are prone to yield far from optimal solutions. Starek et al. [291] enhanced initial locations by applying minor translations while assessing their quality through an objective function, whereas Soudarissanane and Lindenbergh [276] improved the sampling by increasing the grid subdivisions, at the expense of higher response time. For 3D locations, Starek et al. [291] described a SA procedure to translate uniformly sampled points into surrounding locations that improve the covering metric. Kim et al. [300] tried to find the optimal LiDAR position over an autonomous vehicle using Genetic Algorithms (GA). For that purpose, the specifications of commercial LiDARs were used to compute the occupancy grid, defined as a discretized 360° map represented by several views acquiring the coverage region and dead zones. Beyond theoretical/simulation approaches, multiple studies focus on evaluating the set-up of several sensors, concerning height, angles and location in autonomous driving [301, 302].

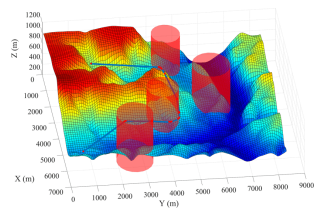


Figure 3.17: Path planning of a fixed-wing military UAS with the objective of minimizing flight altitude and fuel consumption.

Once locations are locally optimized, they are processed as a classic set-coverage problem [276]. Recent research has solved this with GA, using different operators and configurations [283, 284, 303]. Local searches [282] and other nature-inspired heuristics [304] are also reviewed. Despite heuristics being allowed to solve hard problems with optimal or nearly optimal solutions, these are not efficient in most cases. Previous studies regarding P4S required hours, even days, to determine the optimal scanning configu-

ration in 3D environments. Even 2D-based approaches are time-consuming [286] if they are implemented in a sequential manner. Thus, multi-core and GPU-based approaches offer a huge improvement in the response time, reducing it to a few seconds or minutes [286, 299, 303].

3.4 Analysis of Remote sensing data

3.4.1 Vineyard phenotyping

Motivation

Understanding vegetation development is a crucial aspect of crop management that impacts the effectiveness and productivity of agricultural efforts. Precision Agriculture (PA) involves observing agricultural variables that affect crop production, which enables accounting for spatial and temporal variations, resulting in enhanced crop performance, reduced costs, and improved sustainability. Additionally, it provides a forecasting tool to accurately supply crop needs, such as water and nutrients. If applied to vines, the concept becomes Precision Viticulture (PV), with a wide variety of applications, ranging from the detection of biomass [305], water content [306, 307] and other compounds [308], to vigour estimation [309, 310], detection of plant diseases, pest surveillance [311], analysis of grape maturity [312] or yield estimation [313]. They are either focused on grape clusters, leaves, stems or vineyard support [314].



Figure 3.18: Wide angle view of a vineyard plot.

The characterization of vineyard plots using UASs is particularly challenging due to their variability, regarding the tree structure, inter-row spacing and surrounding elements (bare soil, shadowed areas, grassing, etc.), as depicted in Figure 3.18. Therefore, high-detailed images are relevant for discriminating vegetation, soil and weeds, which have been previously shown to affect grape estimations [315, 316]. Consequently, a significant effort of previous studies is oriented toward the segmentation of the canopy [317]. In this regard, UASs help to support decision-making systems by gathering precise information that enables the estimation of biophysical and performance-related features [309].

Traditional hyperspectral classification

DL methods have recently become the preferred approach for classifying hyperspectral imagery. However, earlier techniques relied on comparing the acquired data to reference reflectance shapes that were ideally measured

in a laboratory. The primary objective of these methods was to measure the similarity between labelled and unlabelled spectral shapes. Spectral libraries, containing data measured from a spectrometer, were used for this purpose. For instance, there exist spectral libraries for minerals, trees, and daily surfaces [100, 318, 319]. These methods varied from the widely used Euclidean distance to more sophisticated techniques such as Spectral Angle Matching (SAM), Cross-Correlogram Spectral Matching (CCSM), and probabilistic approaches like Spectral Information Divergence (SID) [22]. Among these techniques, SID and CCSM have been found to perform better in mineral classification from Aviris data. Similarly, van der Meer [320] proved that SAM introduced much more confusion than SID and Spectral Correlation Matching (SCM) in a case study of mineral classification. SAM has the advantage of being invariant to different scales, making it useful for heterogeneous acquisition devices and conditions. Other techniques derived from SAM include Spectral Correlation Angle (SCA), based on the Pearson correlation coefficient, and Spectral Gradient Angle (SGA) [321]. In addition to similarity, angular and probabilistic measures, the literature describes other error- and colourimetric-based methods. The former group includes the widely used Mean Square Error (MSE), Root Mean Square Error (RMSE), Mean Relative Absolute Error (MRAE), Back-Projection MRAE (BPMRAE) and the Peak Signal-to-Noise Ratio (PSNR) [322]. More recently, Kumar et al. [323] introduced three new metrics (Dice Spectral Similarity Coefficient (DSSC), Kumar–Johnson Spectral Similarity Coefficient (KJSSC), and a hybrid of the previous, KJDSSC$_{tan}$) that outperformed traditional techniques on mineral and vegetation classification.

Colourimetric methods measure the distance in various colour spaces, such as pro-Lab. Agarla et al. [322] have compared all these techniques to assess their correlation and determine the most significant ones. Additionally, spectral derivatives, which are finitely approximated considering the previous spectral sample and wavelength distance, are used to remove or compress illumination variations resulting from acquisition conditions [22, 324]. These techniques are still applied when the reflectance profile of different materials exhibits notable variations. Furthermore, semi-automatic classification methods have been reviewed for situations where classification involves only a few labels. For instance, Ahmed et al. [325] used PCA to extract features from multispectral imagery and proposed a vegetation index to label different trees. Pádua et al. [326] described similar work on classifying chestnuts with phytosanitary problems using two new indices: ExNIR and ExRE, where Ex refers to Excess and Re to Red. However, these techniques are not suitable for differentiating a significant number of vineyard varieties since they all exhibit similar shapes. We refer the reader to [327] for an in-depth revision of spectral matching and attributes conditioning the construction of spectral libraries.

Hyperspectral transformation and feature extraction

In this section, the transformations that facilitate classification using traditional methods are discussed. Due to the extensive coverage of land by

satellite imagery, it is uncommon for hyperspectral pixels to depict the spectral signature of a single material. Therefore, there is a prevalent topic in the hyperspectral literature, which involves breaking down the acquired Earth's surfaces by analyzing the hyperspectral images. The problem is illustrated with $\rho = MF + \epsilon$, where M is the spectral signature of different materials, F is the weight, ϵ is an additive noise vector and ρ is an $L \times 1$ matrix where L is the number of bands. Hence, the difficulty of finding a solution to M and F is lowered if M is fixed, i.e., the end-member signatures are known. The Multiple end-member spectral mixture analysis (MESMA) was the initial approach taken, followed by the Mixture-Tuned Matching Filtering technique (MTMF), which eliminates the need to know end members in advance. This approach was further refined with the Constrained Energy Minimization (CEM) method, which effectively suppresses undesired background signatures.

The current state-of-the-art techniques for Linear Mixture Models (LMM) can be categorized based on their dependency on libraries. Additionally, the level of supervision and computational cost also determines the classification of these techniques. The taxonomy of methods, as described by Borsoi et al. [328], varies depending on these factors. For instance, Bayesian methods and Local unmixing do not necessitate the need for known end-member signatures, although Bayesian methods are less supervised and more time-intensive. Besides MESMA, other proposed methods that require spectral signatures are based on AI techniques such as Machine Learning and Fuzzy unmixing. The latter is less supervised but more time-consuming. In recent years, interest in DL has grown, with techniques such as autoencoders, Convolutional Neural Networks (CNN), and Generative Adversarial Networks (GAN) being utilized for training with synthetic data [329]. Non-negative matrix factorization (NNMF) has also attracted attention as it can extract sparse and interpretable features [330]. Recently, the incorporation of spatial information into hyperspectral unmixing has been investigated [331]. This involves considering the surrounding pixels using kernels of varying sizes and shapes, such as squared, cross, or adaptive. Weights can also be assigned based on the distance to the centre and the measured similarity using functions like SID, SAM, Euclidean distance, etc. Current state-of-the-art methods, such as NNMF, have been combined with spectral information [332].

Besides discerning materials, the results of HSI present a large number of layers that can be either narrowed or transformed, as many of them present a high correlation. Otherwise, the large dimensionality of HSI data leads neural networks and other classification algorithms to be hugely complex. An example of data transformation into a few representative features is depicted in Figure 3.19. The larger the distance among clusters, the better is the embedding to tell apart different labels. The most frequent projection method is PCA [333–335], whereas Independent Component Analysis (ICA) is a variation of PCA that not only decorrelates data but also identifies normalized basis vectors that are statistically independent [22]. Least Discriminant Analysis (LDA) is another commonly used technique, but it is primarily applied after PCA to increase inter-class and intra-class

PCA projects an hypercube of size $X \times Y \times \lambda$ into DB, where D has a size of $X \times Y \times F$, and B is a matrix such as $F \times \lambda$. In this formulation, F is the number of target features [65].

distance [334]. In the literature, it is also referred to as Partial Least-Square Discriminant Analysis (PLS-DA), mainly as a classifier rather than a feature selection method.

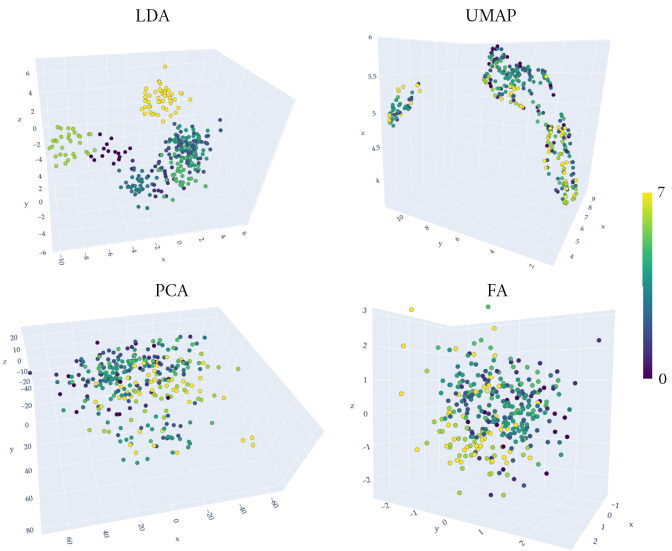


Figure 3.19: Feature transformation of hyperspectral data from a spectrometer using LDA, Uniform Manifold Approximation and Projection (UMAP) [336], PCA and Factor Analysis (FA).

Instead of projecting features into another space, these can be narrowed into the subset with maximum variance according to the classification labels of HSI samples. There are many techniques in this field, including the Successive Projection Algorithm (SPA), which reduces colinearity in the feature vector. The Competitive Adaptive Reweighted Sampling (CARS) method selects features with Monte-Carlo sampling and iteratively removes those with small absolute regression coefficients. Two-Dimensional Correlation Spectroscopy (2DCS) aims to characterize the similarity of variance in reflectance intensity. Liu et al. [337] used the Ruck sensitivity analysis to discard bands with a value below a certain threshold. Agilandeeswari et al. [338] calculated the band entropy, vegetation index, and water index for wavelength subsets, generating a narrower cube only with bands above three different thresholds. Finally, the work of Santos-Rufo et al. [339] presents an in-depth evaluation of methods based on Partial Least Squares (PLS) regression. To this end, HSI data from olive orchards were first narrowed and then classified with LDA and KNN. In conclusion, the Lasso method [340] as well as Genetic algorithms [341] showed the best performance with LDA.

Remote sensing data typically contains inherent noise, which means it is rarely used as-is. To address this issue, Gutiérrez et al. [342] used a combination of Standard Normal Variate (SNV) and de-trending to remove the scatter effect. The hyperspectral signature was then smoothed by applying the Savitzky-Golay filtering with different step sizes over the first and second derivatives.

Classification of HSI with ML and DL

This section focuses on reviewing studies related to the classification of vineyard varieties using HSI. Although there are numerous studies on HSI classification, only those relevant to vineyard varieties will be discussed. In addition, state-of-the-art DL networks achieving high accuracy in HSI classification will also be briefly reviewed.

Despite there exists considerable research on segmentation, only a few studies have addressed the classification of vineyard varieties using RGB and multispectral imagery. In these studies, binary masks or grayscale maps were first extracted to distinguish soil, shadows, and vineyards. Clustering, line detection, or ML algorithms and artificial neural networks (ANN) were then applied to segment vineyard rows [317, 326, 343–346]. Geometrical information from depth maps, DEMs, LiDAR data, and photogrammetric reconstructions were also assessed [109, 347, 348]. DL approaches for semantic segmentation and skeletonization algorithms have also been discussed [108, 349, 350]. Further insight into this field is provided in [351].

The classification of different vineyard varieties has been previously achieved with traditional methods and proximal hyperspectral sensing. Samples were selected by averaging the signature of each grapevine variety and keeping those with a high correlation to such a signature. Support Vector Machine (SVM) and Multilayer Perceptron (MLP) were then trained with k-fold to distinguish thirty varieties, with the latter obtaining better results. Kicherer et al. [352] presented a land phenotyping platform that segments grapes from the depth map and discerns between sprayed and non-sprayed leaves. To this end, several learning models were tested: LDA, Partially Least Square (PLS), Radial Basis Function (RBF), MLP and softmax output layer, with RBF and PLS showing the best results. Besides phenotyping, the detection of plant diseases [353–355] and plagues [311, 356] are also recurrent research topics. However, these applications formulate a binary problem where signatures of distinct classes are significantly different regarding scale [354] and shape [354]. Despite this, previous learning models are also implemented (MLP, RBF, PLS and LDA) and almost achieved the perfect discrimination performance [355]. Nguyen et al. [353] conducted a study similar to ours, where they attempted to differentiate healthy and infected leaves with a comparable spectral signature. However, their data was obtained from land, and they used the flattened layer of 2D and 3D convolutional networks as input for Random Forest (RF) and SVM algorithms. They found that combining PCA reduction (50 features) and RF resulted in the best performance (97%), and RF improved SVM classification regardless of data reduction. Additionally, ML and DL techniques have been extensively applied to various crops, such as maize, sugarcane, rice, and bread wheat, using both satellite and proximal imaging. Transfer learning, attention-based, and residual models are commonly used in the literature [357]. A lightweight CNN composed of several inception blocks was also developed to classify up to 15 plant species [358]. The authors compared their proposed CNN model to other commonly used models for classifying RGB images, including AlexNet, VGGNet, and GoogLeNet. They found that

the best results were achieved using a combination of six RGB and NIR features, with an accuracy of 94.7%. The use of PCA with only six features achieved an accuracy of 88%. Nezami et al. [359] also applied a 3D CNN to classify three tree species using hyperspectral and visible images as well as canopy height models, with an overall accuracy below 95%.

When it comes to DL for the classification of HSI, satellite imaging is more frequent than UAS imaging. To this end, there exists a standard dataset over which experiments are conducted to establish a fair comparison [360]. The top-performing models for classifying HSI are discussed below according to their contributions and overall accuracy (OA). Moraga and Duzgun [361] presented an Inception-based model with parallel convolutional pipelines of increasing size, achieving near-perfect classification. Chakraborty and Trehan [362] proposed the SpectralNet model, which combines wavelet decompositions with a traditional convolutional path (OA: 98.59%-100%). Roy et al. [363] developed HybridSN, which includes both spectral-spatial and spatial feature learning using 3D and 2D convolutional layers (OA: 99.63%-100%). Roy et al. [364] also introduced a network based on residual blocks and spectral-spatial attention modules with varying architecture (start, middle and ending ResBlock) (OA: 98.77%-99.9%). Lastly, Xue et al. [365] presented the A-SOP module composed of matrix-wise operations that output a second-order pooling from the attention weights, after extracting the first-order features (OA: 98.68%-100%).

Similar to the work of Moraga and Duzgun [361], the FSKNet model employs a combination of 2D and 3D convolutional layers with an intermediate separable convolution to reduce training latency while achieving comparable overall accuracy results. The FSKNet model achieved an OA above 99% with significantly fewer parameters and a shorter training time. Most of the revised work performs single-output classification, where the output is either hot-encoded or given as a single value. However, other approaches have gained attention, such as contrastive learning and multi-instance segmentation, which propose outputs of higher dimensionality. Zhu et al. [366] investigated pixel-wise semantic segmentation of HSI patches using the popular U-Net architecture with additional convolutional Long Short-Term Memory (LSTM) and attention-based mechanisms. Xin et al. [367] used transformers to independently encode spatial and spectral features and then combine them. Contrastive learning has also been used to address the lack of labelled datasets, where HSI patches and 1D data are jointly used during training so that the network learns by comparing pairs of samples [368]. Finally, Meerdink et al. [369] accurately labelled HSI with multi-instance learning by creating bags of samples with distinct labels.

3.4.2 Inspection of archaeological remains

UASs have been widely applied in archaeological fieldwork during the last decade [370], and their products have been long studied for analyzing archaeological landscapes [371]. Among them, the results of UAS can be used to generate orthophotos and DEM that facilitate the observation of archaeological marks and microreliefs as proxy indicators of the presence

of buried remains [372, 373]. Furthermore, the archaeological remains are frequently located in natural landscapes barely accessible by humans with irregular features. As a result, UAS-based solutions have been applied to 3D reconstructions as an efficient technology that covers large and inaccessible areas.

The preservation of cultural heritage has recently benefited from the use of high-resolution digital cameras that allow the reconstruction of a scenario with high precision. Aerial images acquired from drones not only allow us to generate 3D models using photogrammetric and SfM techniques but also to obtain information beyond the visible range depending on the coupled sensors. Aerial thermographic imaging has been extensively studied in archaeology as an alternative to visible sensors [374–377], since it allows recording the radiation emitted from object surfaces, either it proceeds from the object itself or surrounding objects [33]. Therefore, it is possible to detect buried archaeological remains through thermal imagery whether heat transfer occurs [374]. Nevertheless, their detection is enhanced if (1) there exists a significant thermal contrast between the background and relevant features, (2) the UAS flight is performed at specific day intervals where the contrast of ambient temperature and sunlight radiation is higher (Figure 3.20), and (3), buried features are close to the surface. Despite thermal orthomosaics being mostly sufficient to reveal artefacts [376, 377], the visualization and understanding of the scene can be improved with 3D reconstructions.

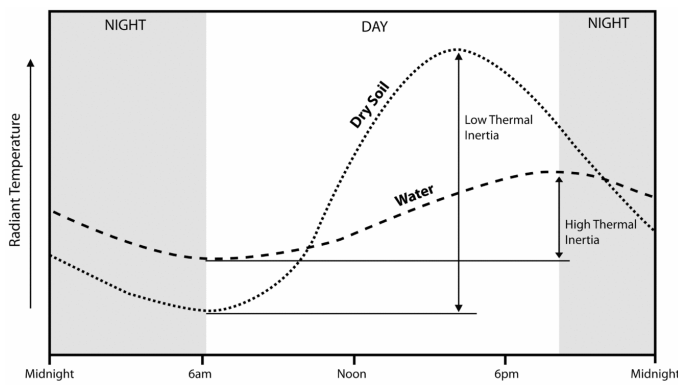


Figure 3.20: Variation of ground and vegetation temperature over time, thus showing which is the best time of the day for collecting thermal imagery (maximum difference among both plots) [374].

However, consumer-grade thermal cameras present low resolution (typically 640×512 or 640×480 pixels) that limits the capability of conventional photogrammetry to generate dense and large thermal point clouds required to analyze the archaeological site [58]. Previous work has achieved the fusion of thermography and 3D data, either from LiDAR or photogrammetry [378], though it has been mainly achieved in close-range inspections and façades. In contrast to thermal imagery, visible sensors acquire images of higher resolution, enabling the estimation of larger and more dense point clouds. Accordingly, RGB point clouds can be used as a constrained data source to map thermographic information, thus taking advantage of their spatial characteristics. For that purpose, the relative difference between visible and thermal imaging sensors needs to be estimated in order

to project 3D points into IR images. The calibration of both sensors can be performed before the flight, by detecting key points in checkerboards or similar patterns [58, 173], or afterwards, by registering co-acquired images [58]. Although multiple projection algorithms are described in the literature, these outperform others based on the ICP [42] or simply on SfM reconstruction [40, 50], both in terms of precision and density.

Part II

Materials and image fusion

The first part of this chapter introduces the materials utilized in this research work. Specifications and shortcomings of each device are here illustrated and solved for the sake of clarity and to avoid its repeatability through this dissertation. Then, images are processed and matched to compose a multi-layer system that will help to construct enriched 3D models in later chapters.

Materials and software 4

About this chapter

This chapter comprises the specifications of sensors used throughout this research work, the drawbacks of each of them and how these were solved. The core data of this dissertation is imagery taken from UASs; therefore, this section is dedicated to both imagers and drones applied to capturing input data. Materials either belong to the Graphics and Geomatics Group at the University of Jaén (TIC-144) or the Science and Technology Faculty at the University of Trás-os Montes e Alto Douro (Vila Real, Portugal).

4.1 UAS platforms 85
4.1.1 DJI Matrice 210 85
4.1.2 DJI Matrice 600 Pro 85
4.1.3 DJI Phantom 4 86
4.2 Sensors 86
4.2.1 Parrot Sequoia 86
4.2.2 DJI Zenmuse XT2 88
4.2.3 Nano-Hyperspec 93

4.1 UAS platforms

The UAS platforms used during any of the surveys are following described. The first belongs to the University of Jaén whereas the latter two have been used over the region of Vila Real, Portugal.

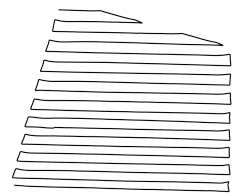


Figure 4.1: Boustrophedon path planning.

4.1.1 DJI Matrice 210

DJI Matrice 210 is a quadcopter drone that has been coupled with multispectral and thermal imagers. Missions were planned and launched using DroneDeploy (California, CA, USA) in the remote control device. Flights have been typically established at 45-50 m from the take-off position, whereas the path planning has always been set to define a Boustrophedon path (see Figure 4.1), i.e., back and forth parallel lines. View direction was configured as *nadir* when possible, with high frontal (~90%) and side (~85%) overlap to ensure images can be matched.



Figure 4.2: Quadcopter DJI Matrice 210 equipped with a thermal sensor, DJI Zenmuse XT2, and the Parrot Sequoia multispectral sensor.

4.1.2 DJI Matrice 600 Pro

DJI Matrice 600 Pro (M600) is a hexacopter which has been used to record hyperspectral data with a Nano-Hyperspec device. It is equipped with a Ronin-MX gimbal to minimize geometric distortions in HSI acquisition. Its location was captured at different timestamps using two GPS in the arms, and its altitude was determined by the other three GPS mounted on the upper plate. Angular data (yaw, roll and pitch angles) were recorded using an IMU. With this equipment, the flight autonomy is approximately 20 min [21]. Flights were planned using Universal Ground Control Station at an altitude of 50 m with a 40% side overlap for the observation of vineyard rows and the University of Trás-os Montes e Alto Douro campus.



Figure 4.3: Hexacopter DJI Matrice 600 Pro.

4.1.3 DJI Phantom 4



Figure 4.4: Quadcopter DJI Phantom 4.

DJI Phantom 4 is a quadcopter with an integrated CMOS camera of 2.8 mm optical lens and 12.4 megapixels (MP). Images have a resolution of 4000×3000 px, whereas video recording is also allowed at different qualities, including Ultra High-Definition (UHD: 4096 × 2160 px at 24/25 frames per second (FPS)). It can fly up to an altitude of 6000 m above water level with an autonomy of approximately 28 min. Integration with other sensors such as the Parrot Sequoia is not trivial as it requires the design of a coupling platform and feeding the sensor with an energy source, e.g., a battery. Although it is possible to integrate further sensors, this UAS has only been applied to RGB recording with the integrated camera. Path planning has been performed with DroneDeploy (DroneDeploy, San Francisco, CA, USA).

4.2 Sensors

4.2.1 Parrot Sequoia



Figure 4.5: Parrot Sequoia multispectral device.

Parrot Sequoia is a multispectral sensor from Parrot that has been mainly advertised as an imager for Precision Agriculture. It covers four different spectral bands (see Table 4.1) using wide-angle lenses with a focal length of ∼4 mm. Along with multispectral observations, it is also able to capture information in the visible range with an RGB sensor of 15.9 MP and a focal length of 4.88 mm. The latter is captured in rolling shutter mode, and therefore, not all parts of the image are captured at the same time. Thus, the UAS pace ought to be configured low enough to avoid effects such as wobble, skew as well as spatial and temporal distortion. Regarding the dimensionality of outputs, the resolution is 1280×960 px and 4608×3456 px for multispectral and visible imagery, respectively. Examples of shots from the Parrot Sequoia device for two different study areas are depicted in Figure 4.6.

Table 4.1: Specifications of the Parrot Sequoia multispectral device.

Band	Spectral range	Image size	Focal length
Green (GRE)	530 – 570 nm	1280 × 960 px	3.8 mm
Red (RED)	640 – 680 nm		
Red-edge (REG)	730 – 740 nm		
Near-infrared (NIR)	770 – 810 nm		
Visible (RGB)	Visible	4608 × 3456 px	4.88 mm

Wide-angle lenses cover wider areas with a single shot, but also present visual distortions such as the well-known fisheye effect (Figure 4.7). Surfaces with rectilinear features are depicted as domed with this distortion, especially for features close to the image corners. It can be corrected with a transformation matrix that creates an image of the same dimensionality,

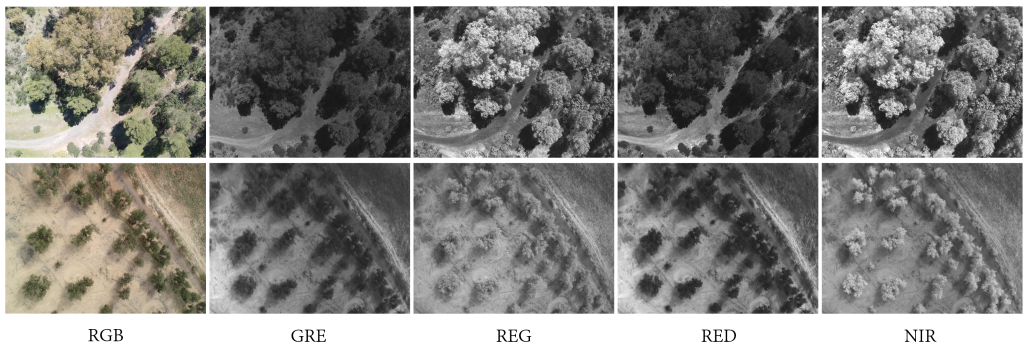


Figure 4.6: RGB and multispectral bands as captured from a Parrot Sequoia device.

whose pixels carry out the interpolated colour from several pixels of the source image. Remark that interpolation algorithms are required since the transformation of $x' \in \mathbb{N}$ yields $x \in \mathbb{R}$.

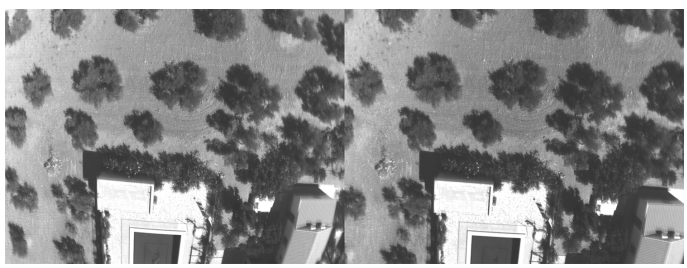


Figure 4.7: Image with fisheye effect and the target image after the removal of such a distortion.

Figure 4.8 shows this relation on the basis of α and β angles. β is the angle between the optical axis and the segment that goes from the lens' origin, c_o, to a source point, p, that is paired with a target point, p'. For the target image, α is used instead of β, and p is transformed to p'. Note that $p \neq p'$, though both lie on the segment $\overline{c_p p}$. The explained transformation push points into c_p, and it is formally defined by means of $f(x) = x'$ and $f(y) = y'$. Therefore, α is easily calculated with trigonometry using the optical axis and any segment $\overline{c_o p'}$, as shown in Equation 4.1. Besides this, α is normalized in $[0, 1]$:

$$\alpha = \frac{2}{\pi} * \tan^{-1}\left(\frac{\sqrt{(p'_x - c_{p_x})^2 + (p'_y - c_{p_y})^2}}{f}\right) = \frac{2}{\pi} * \tan^{-1}\left(\frac{r}{f}\right) \quad (4.1)$$

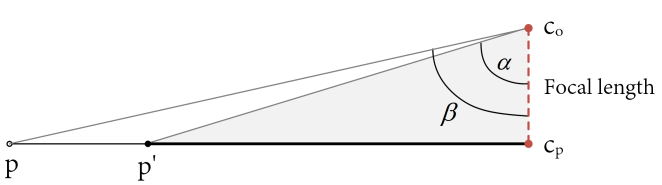


Figure 4.8: Relation between a pixel from the target image, p', and another from the source image, p.

Then, four polynomial coefficients (k_1, k_2, k_3, k_4) are read from the image metadata to compute the angle β between the optical axis and the

segment $\overline{c_o p}$, as expressed in Equation 4.2. Note that this correction model is provided by Parrot.

$$\beta = k_1 + k_2 * \alpha + k_3 * \alpha^2 + k_4 * \alpha^3 \tag{4.2}$$

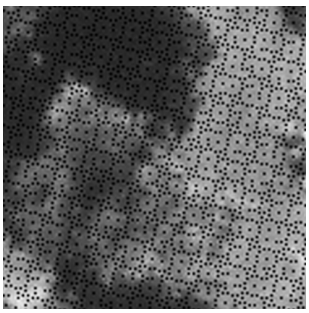


Figure 4.9: Undetermined values are obtained when the undistortion process is carried out from the source image to the target one.

The new image has the same size, with p' receiving the colour of p. It is relevant to perform this operation in this order, $p' \leftarrow p$, to determine the colour of each pixel. Instead, the inverse mapping ($p \rightarrow p'$) leaves undetermined values since multiple pixels from the source image could be mapped to the same pixel in the target image (see Figure 4.9). Thus, the last step is to map every (x', y') to (x, y), with $x, y \in \mathbb{R}$. A bilinear interpolation algorithm is used to solve the problem of accessing float-point indices from an image.

4.2.2 DJI Zenmuse XT2



Figure 4.10: DJI Zenmuse XT2 dual-payload sensor.

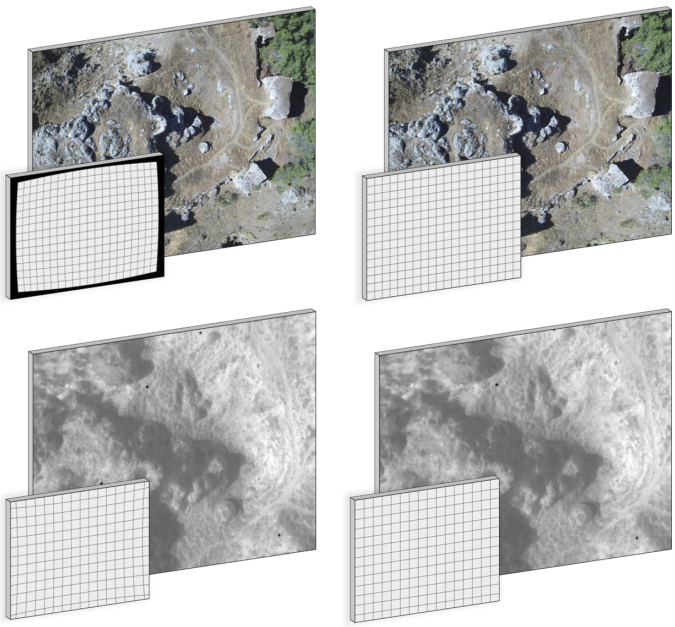
DJI Zenmuse XT2 is a dual payload device that includes RGB and thermal cameras. The RGB camera provides high-resolution images (CMOS sensor with 12 MP) with a focal length of 8 mm, whereas the thermal imager acquires results of lower resolution (Table 4.2) in the LWIR interval from surfaces whose temperature range between $-40°$ and $550°$. IR radiation was configured to be represented as grayscale values enhanced by edges from objects. The device as a whole is stabilized by a gimbal. Both types of images are acquired synchronously and despite this, there exist differences among them due to possible minor delays in the triggering, drone movement and distance of lenses. RGB images are stored in Joint Photographic Experts Group (JPEG) file format, whereas thermal imagery is either stored as RJPEG (Radiometric JPEG) or TIFF (Tag Image File Format). The first allows calculating the temperature as a function of a large number of parameters, whereas the latter saves the temperature as observed and configured. Hence, the first is better if some of the parameters must be tweaked on post-processing, e.g., the atmospheric temperature. As occurred for previous devices, these images are also affected by distortions either from wide-angle or very narrow lenses.

Table 4.2: Specifications of images acquired by a DJI Zenmuse XT2 device.

Band	Spectral range	Image size	Focal length
Thermal	$750 - 1,350$ nm	640×512 px	19 mm
Visible (RGB)	Visible	4000×3000 px	8 mm

Geometric correction

Images captured by the DJI Zenmuse XT2 are affected by distortions that must be removed for later projection procedures. First, thermographic imagery presents pincushion distortion as a result of the narrow field of view ($32° \times 26°$), whereas RGB images are distorted following the barrel effect. Unlike Parrot, the camera manufacturer does not define a specific distortion model and thus can be removed as for any other image. Geometric



a) Original images b) Corrected images

Figure 4.11: Result of geometrical corrections regarding visible and thermal imagery. Barrel and pincushion effects are depicted through distorted grids.

distortions can be removed by means of the camera matrix, K, as well as radial and tangential distortion coefficients, $(k_1, k_2, p_1, p_2, k_3)$. Also in contrast to Parrot Sequoia, these factors are not provided as image metadata. Instead, they must be determined through calibration and as such are estimated as part of the initial processing of SfM. Whether an image follows a pincushion or barrel distortion model is determined by the main distortion coefficient, k_1. It is below zero for thermal images, i.e., surfaces are not rectilinear; instead, they bow inwards as a consequence of the pincushion distortion. On the other hand, RGB images have a barrel effect that is removed in the same way, though their principal radial coefficient, k_1, is greater than zero. For the latter, surfaces are bowed outwards and therefore, removing the distortion does not require decimating the image (Figure 4.12).

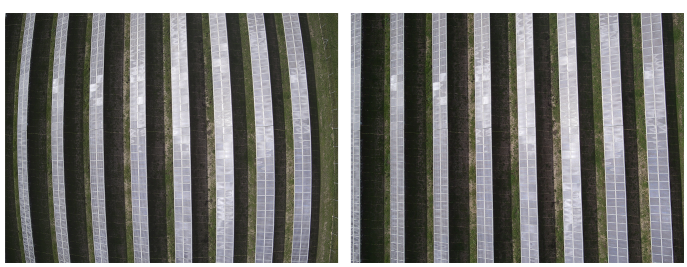


Figure 4.12: Comparison of distorted RGB image and the corrected result.

As depicted in Figure 4.11, corrected images have null values if the result has a lower dimensionality than the starting image. Hence, the minimum area that presents non-null colours can be determined by calculating the new corners of the image and subsequently cropping it (Equation 4.3). As

a result, corrected images can have lower resolution than the initial ones, as occurs for thermal images. Figure 4.13 compares three stages of the correction procedure.

$$\begin{aligned}
\min_x &= \max\{\max_x\{M \cdot [0,0,1]^\mathsf{T}, M \cdot [0,h-1,1]^\mathsf{T}\}, \min_x\} \\
\min_y &= \max\{\max_y\{M \cdot [0,0,1]^\mathsf{T}, M \cdot [w-1,0,1]^\mathsf{T}\}, \min_y\} \\
\max_x &= \min\{\min_x\{M \cdot [w-1,0,1]^\mathsf{T}, M \cdot [w-1,h-1,1]^\mathsf{T}\}, \max_x\} \\
\max_y &= \min\{\min_y\{M \cdot [0,h-1,1]^\mathsf{T}, M \cdot [w-1,h-1,1]^\mathsf{T}\}, \max_y\}
\end{aligned} \tag{4.3}$$

Distortion is removed by following the transformations defined in Equations 4.4, whereas the inverse produce can also be followed to guarantee every pixel is mapped to a colour.

$$\begin{aligned}
\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} &= \begin{bmatrix} \frac{x-c_x}{f_x} \\ \frac{y-c_y}{f_y} \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\
r^2 &= x'^2 + y'^2 \\
x_u &= x' \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2(r^2 + 2x'^2) \\
y_u &= y' \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y'^2) + 2p_2 x' y'
\end{aligned} \tag{4.4}$$

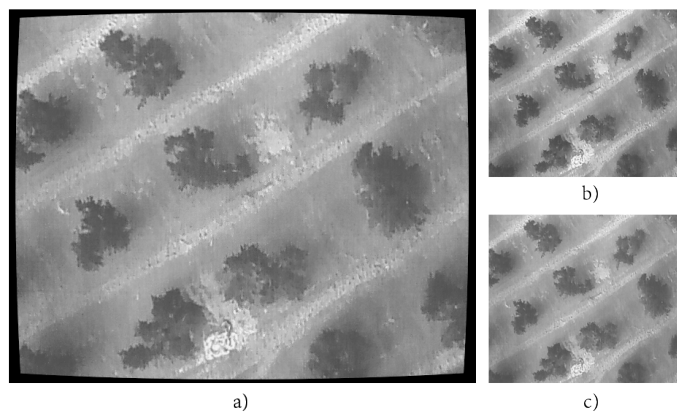


Figure 4.13: Comparison of a) free of distortion thermal image with blank values preserving the original size, b) distorted thermal image and c) free of distortion thermal image with reduced size.

x_u, y_u refer to coordinates from an image without distortion effects, although corrected images are generated in the opposite order. For each pixel, defined by its integer coordinates, the distorted pixel from where colour is obtained, $x_d, y_d \in \mathbb{R}$, is calculated using Equation 4.5. Again, an interpolation such as the bilinear function is required in this process.

$$\left[x_d, y_d, 1\right]^\mathsf{T} = \left[x' f_x + c_x, y' f_y + c_y, 1\right]^\mathsf{T} = K \left[x', y', 1\right]^\mathsf{T} \tag{4.5}$$

Radiometric correction

This section is mainly referred to thermography stored using the proprietary RJPEG file format. With this encoding, images are depicted as a normalized representation of the temperature observed in the scene. However, these values are not the observed temperature. Still, the temperature can be extracted using the metadata of RJPEG files. Thermal cameras record the radiation emitted from the objects' surface, either it comes from the object itself or surrounding objects. Additionally, it is conditioned by several environmental parameters, such as air humidity, air temperature and background temperature. Besides this, the object reflectivity and emissivity, as well as the distance between the surface and the camera are also relevant to the final measurements [33]. Each manufacturer models the transmissivity of the atmosphere through a theoretical or empirical law, using some constant values embedded in the image metadata [379]. Consequently, several parameters must be retrieved from the embedded data, including the emissivity, atmospheric temperature, reflected apparent temperature, infrared window temperature, infrared window transmission, relative humidity, Planck's constants (P_{R1}, P_{R2}, P_O, P_F, P_B) and atmospheric transmission constants (α_0, α_1, β_0, β_1, $\mathfrak{X}$). Most of them are calibrated by the manufacturer against a blackbody radiation source and thus only a few parameters can be adjusted. The final temperature T is given by Equation 4.6:

Thermographic devices must be calibrated so that temperature readings are appropriate once the emissivity is properly chosen. Either this operation must be performed by users, or calibration is performed by the manufacturer following some standards. In the latter scenario, there exist what is called blackbody radiators with a stabilized temperature made of materials with emissivity factors such as $\varepsilon = 0.98$. These are intended to absorb almost every incident radiation and no material can have a higher response than this. Also, the emitted radiation is not affected by the viewing angle; it is only affected by wavelength.

$$T = \frac{P_B}{ln\left(\frac{P_{R1}}{P_{R2}(r+P_O)} + P_F\right)} \tag{4.6}$$

where r is the raw radiance of a pixel, transformed as listed in Code 4.1. Note that the following code works for a single pixel, although it can be vectorized to be applied over an entire image.

```
E=meta['Emissivity']
OD = 1.0
if 'SubjectDistance' in meta:
    OD = meta['SubjectDistance']
RTemp=meta['ReflectedApparentTemperature']
ATemp=meta['AtmosphericTemperature']
IRWTemp=meta['IRWindowTemperature']
IRT=meta['IRWindowTransmission']
RH=meta['RelativeHumidity']
PR1=meta['PlanckR1']
PB=meta['PlanckB']
PF=meta['PlanckF']
PO=meta['PlanckO']
PR2=meta['PlanckR2']

# constants
ATA1 = 0.006569
ATA2 = 0.01262
ATB1 = -0.002276
ATB2 = -0.00667
ATX = 1.9

# transmission through window (calibrated)
```

Listing 4.1: Process of converting radiance as a DN to temperature in Celsius units.

```
emiss_wind = 1 - IRT
refl_wind = 0

# transmission through the air
h2o = (RH / 100) * exp(1.5587 + 0.06939 * (ATemp) - 0.00027816 * (ATemp) ** 2 + 0.00000068455 * (ATemp) ** 3)
tau1 = ATX * exp(-sqrt(OD / 2) * (ATA1 + ATB1 * sqrt(h2o))) + (1 - ATX) * exp(-sqrt(OD / 2) * (ATA2 + ATB2 * sqrt(h2o)))
tau2 = ATX * exp(-sqrt(OD / 2) * (ATA1 + ATB1 * sqrt(h2o))) + (1 - ATX) * exp(-sqrt(OD / 2) * (ATA2 + ATB2 * sqrt(h2o)))

# radiance from the environment
raw_refl1 = PR1 / (PR2 * (exp(PB / (RTemp + 273.15)) - PF)) - PO
raw_refl1_attn = (1 - E) / E * raw_refl1
raw_atm1 = PR1 / (PR2 * (exp(PB / (ATemp + 273.15)) - PF)) - PO
raw_atm1_attn = (1 - tau1) / E / tau1 * raw_atm1
raw_wind = PR1 / (PR2 * (exp(PB / (IRWTemp + 273.15)) - PF) ) - PO
raw_wind_attn = emiss_wind / E / tau1 / IRT * raw_wind
raw_refl2 = PR1 / (PR2 * (exp(PB / (RTemp + 273.15)) - PF)) - PO
raw_refl2_attn = refl_wind / E / tau1 / IRT * raw_refl2
raw_atm2 = PR1 / (PR2 * (exp(PB / (ATemp + 273.15)) - PF)) - PO
raw_atm2_attn = (1 - tau2) / E / tau1 / IRT / tau2 * raw_atm2
raw_obj = (raw / E / tau1 / IRT / tau2 - raw_atm1_attn - raw_atm2_attn - raw_wind_attn - raw_refl1_attn - raw_refl2_attn)

# temperature from radiance
temp_celcius = PB / log(PR1 / (PR2 * (raw_obj + PO)) + PF) - 273.15
```

However, reading the embedded metadata and estimating the temperature for every image is time-consuming. For this reason, these values can be calculated once and stored using a binary encoding to speed up later readings. Figure 4.14 shows the minimum and maximum temperature observed in a scene as well as the normalized temperature using the colour ramp depicted on the right side.

On the other hand, the TIFF file format stores the absolute temperature and is simpler to process. Still, DNs must be transformed according to Equation 4.7, with h being a factor that takes values $\{0.4, 0.04\}$. The value of h depends on whether the sensor is configured in high-gain mode or not. Starting values are presented using 16-bit floating-point numbers that

can be normalized in [0, 1], thus allowing us to visualize them.

$$T_{absolute} = h * DN - 273.15\ ^{\circ}\mathrm{C} \tag{4.7}$$

$$h = \begin{cases} 0.04 & \texttt{High gain mode} \\ 0.4 & \texttt{Low gain mode} \end{cases} \tag{4.8}$$

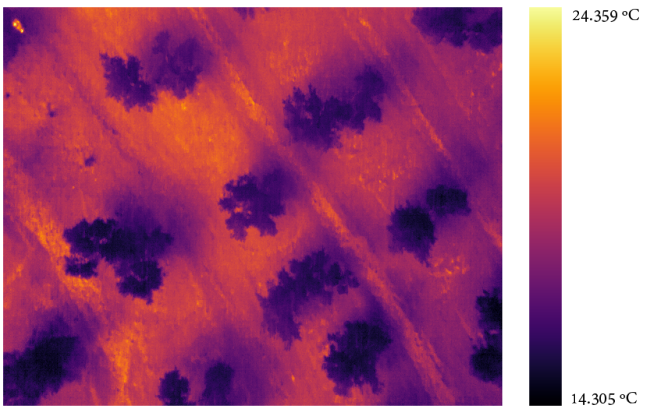


Figure 4.14: Normalized thermographic image encoded with the inferno colour-ramp texture.

4.2.3 Nano-Hyperspec

Nano-Hyperspec is a push-broom sensor that acquires spectral information for each spatial line. Light enters through the lens and it is dispersed in the spectral axis, thus allowing the observation of the spectra with a dimensionality of 640 spatial pixels. From here, the area is scanned following a flight direction and generating up to 2000 spatial lines per swath. If scanning the area requires more than 2000 spatial lines, several swaths are obtained for each sweep. 272 bands are acquired for every pixel covering a spectral interval ranging from 400 to 1,000 nm. The sampling frequency is 2.2 nm, though it increases to 6 nm at half-maximum.



Figure 4.15: Nano-Hyperspec sensor.

The processing of hyperspectral data has been performed with Headwall SpectralView™ software. The data collection procedure returns several swaths for each study area, with a grayscale tarp visible in at least one of them. Then, a white sample is marked from the white area in the grayscale tarp, while the dark reference is obtained by collecting a hyperspectral sample with the lens cap on before the flight. The sensor exposure and frame period are also adjusted before the flight by pointing at a bright reference to avoid clamping samples from white surfaces. The white and dark references are then used to convert the raw data to reflectance.

Ortho-rectified swaths are calculated using high-resolution DEMs (25 m) from Copernicus' observation program [19] and the drone's GPS and IMU data. However, non-ortho-rectified swaths have also been used in this work for analysis to avoid distorting the hyperspectral samples and working with smaller image sizes.

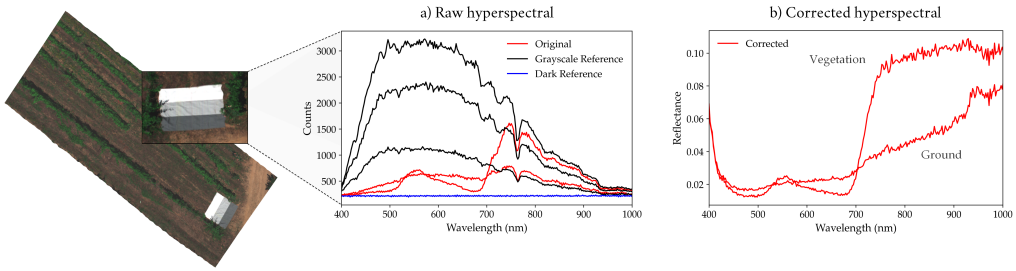


Figure 4.16: Conversion of a) hyperspectral DNs into b) reflectance using a white and dark reference. The three grey levels are sampled from a).



Figure 4.17: Hyperspectral orthomosaic obtained by adjusting the yaw, roll and pitch angles as well as the UAS's altitude in SpectralView™ software for every swath.

Fusion of UAS-based image datasets

5

About this chapter

This chapter describes the image registration algorithm on which this dissertation is articulated. The matching of multispectral and thermographic images with visible information is here addressed using the Enhanced Correlation Coefficient (ECC) for maximizing the correlation of images in different spectral ranges. A comparative assessment is then conducted to discuss which is the best configuration to balance both time efficiency and quality of matching results. Shortcomings that are frequently found in remotely sensed aerial images were presented and solved in Chapter 4.

5.1 Matching multispectral imagery 95
5.2 Matching RGB and multispectral imagery . 97
5.3 Matching RGB and thermographic images . 98
5.4 Segmentation of individual trees 99
5.5 Results and discussion 100
5.5.1 Multispectral registration 101
5.5.2 RGB and multispectral image-matching 102
5.5.3 Thermal and RGB image-matching 104
5.6 Conclusions and future work 105

5.1 Matching multispectral imagery

The Parrot Sequoia multispectral sensor captures four spectral bands: Green (GRE), Red (RED), Red Edge (REG) and Near Infrared (NIR). These are acquired by the same device but with different lenses aimed at perceiving different spectral wavelengths. It can be checked that there exists a lack of alignment between them if they are overlapped as collected, as shown in Figure 5.2. At least the following differences were identified:

- Images are translated as a result of each lens working in a **different fixed position within the multispectral device**.
- **Every lens has its own optical axis and principal point**. The ideal configuration consists of parallel optical axes that help to solve the misalignment with a simple translation. Even if this was the configuration after manufacturing, the lens calibration deteriorates over time. Indeed, the image metadata shows the viewing conditions of each band with respect to the master one, Green, in what is known as a camera ring. It is defined as a set of cameras which are connected and defined by geometric constraints. Every band defines its own yaw, roll and pitch angles with respect to the master, and thus optical

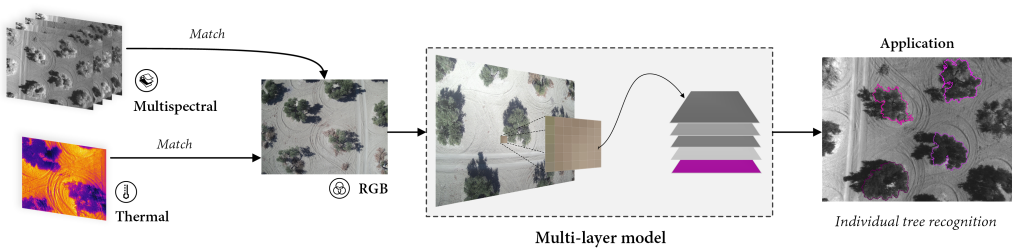


Figure 5.1: The fusion of multispectral, thermographic and RGB imagery compose a multi-layer model which is later applied to the characterization of individual trees.

axes cannot be assumed to be parallel.

- **Timestamps are different even for multispectral images triggered at the same time**. It is in the order of ms, and yet it implies a harder matching because of the UAS's movement.

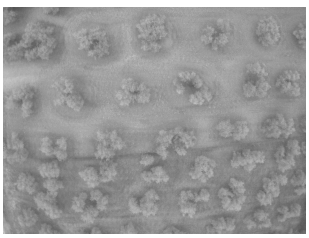


Figure 5.2: Image ghosting effect obtained by overlapping multispectral bands using $\alpha < 1$.

The image matching of multispectral data can be achieved with the parameters of the image's metadata, thus relying on sensor calibration. However, another approach is to match images using only colourimetric information. Traditional image-matching approaches, such as SIFT, SURF and ORB, do not work well over images from different wavelength intervals that depict surfaces with different radiation in distinct bands. Although there exist variants from SIFT that work under photometric differences [380], these are not as robust as the **Enhanced Correlation Coefficient (ECC)** [381]. Given two images of the same dimensionality, ECC seeks the matrix that maximizes the intensity correlation of both images, and it is more convenient for our work for the following explained reasons. First, it works by estimating a transformation matrix with variable complexity (see Figure 5.3), ranging from a simple translation (2×3) to a homography (3×4); the larger the matrix, the more time-consuming the process. Therefore, the motion model can be determined considering the desired time efficiency and what kind of transformations the misalignment presents. Even for the homography, the algorithm has linear complexity ($\mathcal{O}(n)$). The configurable parameters are the number of maximum iterations upon convergence (n) as well as the aimed precision (p). Another intrinsic parameter is the dimensionality of input images, as lower dimensionality considerably reduces latency.

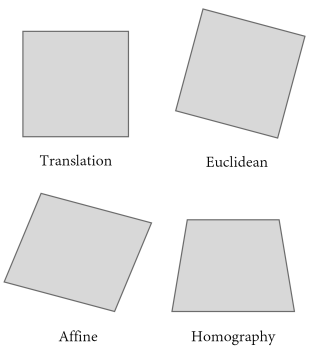


Figure 5.3: Transformation models that can be estimated using ECC.

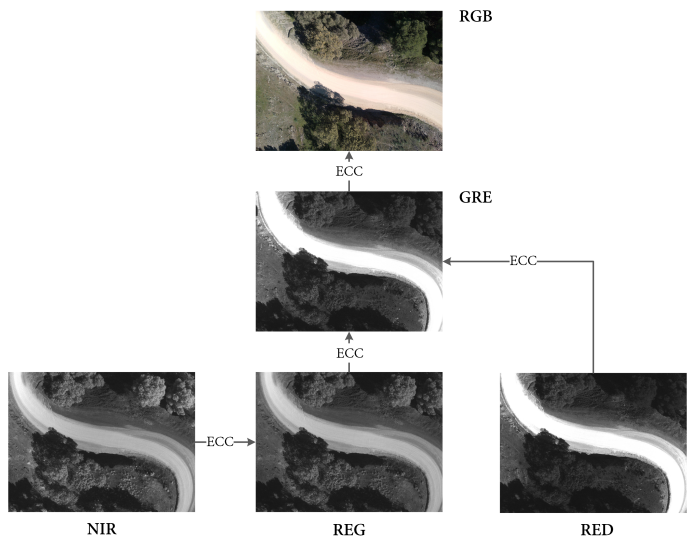


Figure 5.4: Hierarchical registration of multispectral imagery.

Homography matrices are able to describe any kind of differences among images, but the Euclidean model was observed to be enough for matching multispectral bands. Hence, the complexity was reduced by estimating a matrix of size 2×3 explaining translation, rotation and scaling, with the first being the most relevant. The procedure can be further improved by

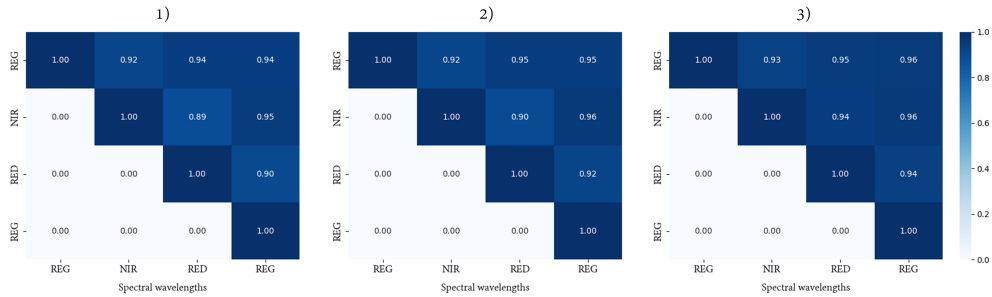


Figure 5.5: Confusion matrices depicting the similarity of multispectral images in three different datasets.

first matching those images which are more similar in terms of intensity. According to the peculiarities of multispectral images, these are fused hierarchically rather than sequentially. With this approach, images with more similar intensity are first registered. More specifically, the pipeline depicted in Figure 5.4 is followed: $GRE \longrightarrow RGB$, $RED \longrightarrow GRE \longrightarrow RGB$, $REG \longrightarrow GRE \longrightarrow RGB$ and $NIR \longrightarrow REG \longrightarrow GRE \longrightarrow RGB$. Still, each image is only registered once, as the whole chain concatenation is computed using matrix composition. However, the algorithm does not always converge to the expected precision for every pair of images nor the results are optimal. When this occurs, the resulting quadrilateral shape is very distorted, and it can be identified by setting a threshold angle, β, that is compared against the angles enclosed in the obtained shape. Additionally, the transformed result can have areas null values, as occurred with the correction of fisheye distortion. Once again, the minimum area to be cropped can be calculated as defined in Equation 4.3. It cannot be simply computed from the lower left and upper right corners, $[0, 0]$ and $[w - 1, h - 1]$, since Euclidean transformations also involve rotations. Thus, it is not enough to check two non-adjacent corners.

The hierarchical alignment of multispectral images is constructed on the basis of the similarity between multispectral bands, as shown in Figure 5.5. It shows the similarity between any pair of bands by calculating the normalized correlation coefficient (CC), which shows how similar two grayscale images are. For the sake of simplicity, the Green image is considered the root as in the camera ring. Accordingly, Figure 5.5 justifies the hierarchy shown in Figure 5.4. Note that NIR and REG layers are reciprocally the most similar. However, REG images are more similar to the root, Green, and thus NIR layers were aligned to Red-Edge (REG) images.

5.2 Matching RGB and multispectral imagery

The contribution of RGB imagery to this model is to add more accurate geopositioning data. However, multispectral and RGB datasets were not acquired by the same device in this study, nor they were synchronized on the triggering. Thus, images to be matched are selected according to the timestamp, and yet, the differences can be in the order of seconds. Similarly

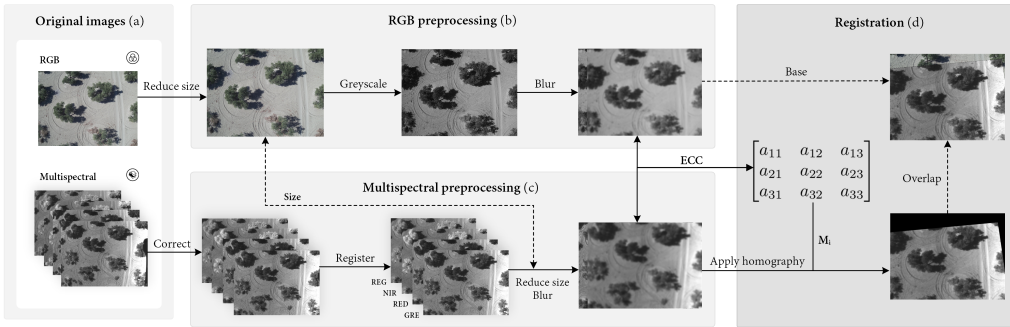


Figure 5.6: Image-matching of RGB and multispectral datasets. The blur size was increased for a better understanding of the figure.

to the previous case study, the ECC algorithm suits well the matching of RGB and multispectral imagery since they have some relevant differences concerning the observed intensity. Instead of matching every multispectral image, only the Green band is required according to the hierarchy proposed in Figure 5.4.

RGB images have higher dimensionality than multispectral images, and therefore, the former was downscaled to match the size of the latter. The ECC algorithm can be applied at a sub-pixel level and its parameters can be configured for a fine-grained registration. Furthermore, a faster convergence can be achieved by blurring both images, despite including the overhead from blurring. The complete workflow is shown in Figure 5.8, using a blur mask of size 3, a high number of iterations, $n \leftarrow 400$, and a high precision factor, $p \leftarrow 1^{-60}$. The values of these parameters are further discussed in Section 5.5, whereas Figure 5.7 shows an example of this matching procedure.



Figure 5.7: An example of image-matching between an RGB image and the Green multispectral band, with the initial images being depicted on the right side.

5.3 Matching RGB and thermographic images

The last step of this multi-layer system is to match thermal and RGB imagery, with RGB images being the link between multispectral and thermal datasets. Previous shortcomings related to different triggering timestamps are here alleviated since both kinds of images are captured by the same device. Still, there exist large differences in the optical acquisition of RGB and thermal images. The RGB lens captures a wider area with a fisheye distortion model,

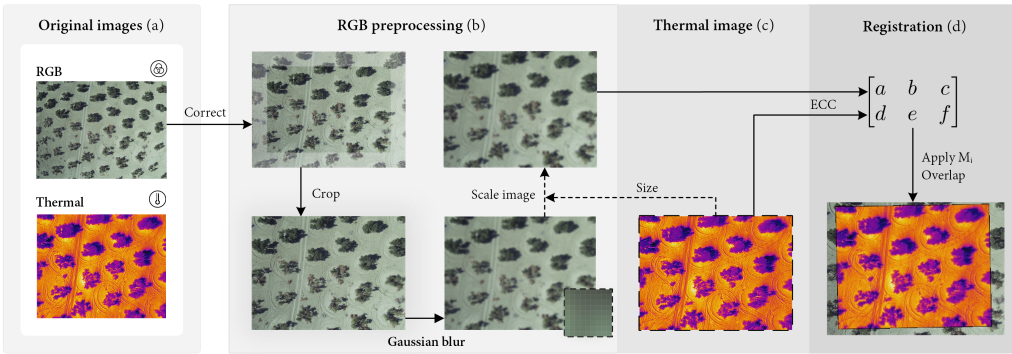


Figure 5.8: Flow chart of the image-matching procedure between RGB and infrared imagery.

whereas the thermal lens has a more narrowed field of view. Similarly to multispectral images, the dissimilarities between these two types of images can be attributed to translations resulting from variations in the physical distance between lenses, as well as rotations that arise from the deployment of distinct optical systems. The latter factor is particularly significant, as it poses a challenge for accurately extracting the region of interest captured by both devices.

After identifying the shortcomings, it was observed that an affine transformation sufficed for addressing them. Translation and rotation were found to be effective in mitigating the visual disparities between the RGB and thermal imagery, while scaling was necessary to ensure appropriate alignment between the two images. Two approaches could be employed to achieve a successful match: a) selecting a larger area of the RGB image that fully encompasses the thermal image, resulting in some null values, or b) selecting a smaller area that does not cover the entire thermal image, thus addressing the previously mentioned issue (see Figure 5.9). To improve the response time, the RGB images were downscaled to fit the thermal dimensions. The precision factor and the number of iterations were fine-tuned to achieve accurate registration. The complete procedure is outlined in Figure 5.8.

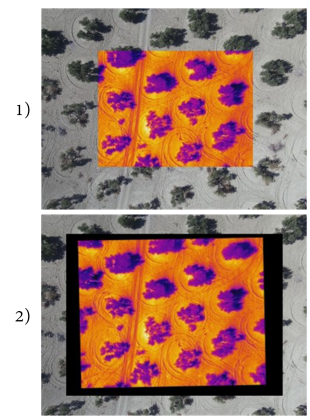


Figure 5.9: Matching of RGB and thermal images using 1) an RGB area smaller than thermal images and 2) a larger RGB area.

5.4 Segmentation of individual trees

The proposed framework enables obtaining RGB, multispectral and thermographic data from an area of interest. This stack of layers can be further applied to the temporal tracking of individual trees. More specifically, multispectral images are sensitive to vegetation and thus can help in the identification of trees. Once identified, points belonging to each one can be further analyzed to extract vegetation indices such as NDVI.

The observed surface reflectance, $f(\theta)$, exhibits distinctive peaks where the reflectance of vegetation reaches a maximum value. This contrast between layers with higher and lower reflectance enables differentiation between soil and canopy. While the data lacks the finer resolution of hy-

perspectral data, the relative maximum and minimum reflectance values for vegetation are visible in the NIR and Red layers, respectively. As soil reflectance is generally constant or low across these layers, it can be effectively filtered out. Consequently, applying the formula $NIR - Red$ yields the outcome depicted in Figure 5.10.

The following steps are performed before the image subtraction:

1. **Gaussian blur** to smooth the image colour and remove noise.
2. **Image thresholding** to isolate the vegetation areas.
3. **Extraction of the hierarchy of contours**. Contours that are inside another one, those that belong to small low vegetation or noise, and partially visible (cropped) contours are all deemed irrelevant. It is easy to identify the latter issue as cropped contours have horizontal or vertical edges in the image boundaries.

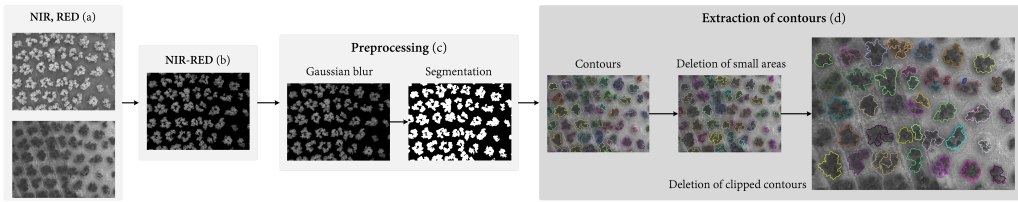


Figure 5.10: Transformation of multispectral images to distinguish vegetation and soil.

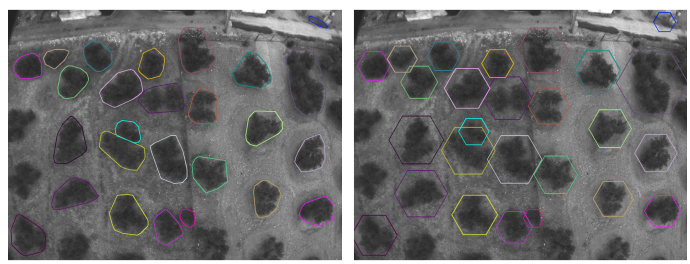


Figure 5.11: Conversion of polygons into convex and hexagonal hulls.

The identification of individual trees is determined by the size of a blur mask, the maximum area to be discarded, and a threshold value. This threshold must be large enough to prevent the merging of closely located trees. Although this procedure produces shapes with a high LOD, they typically contain hundreds of points that can be simplified using a convex hull [382] or other polygons, such as hexagons (see Figure 5.11).

5.5 Results and discussion

This methodology has been evaluated with four different datasets depicting olive orchards located in Jaén, a Southern region of Spain. The first three datasets have multispectral and RGB images, whereas the last one comprises RGB and thermal imagery. The main study area covers two hectares of olive trees in which the proposed methodology has been checked. Figure 5.12 presents a general overview of the main study area.

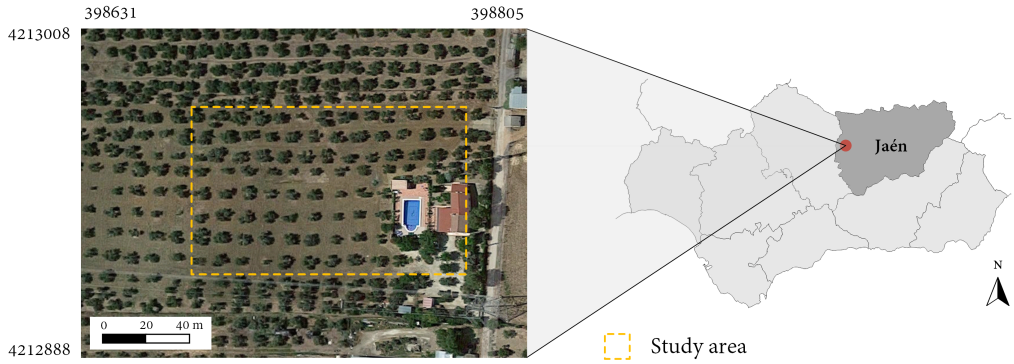


Figure 5.12: An overview of the study area presented in the Universal Transverse Mercator (UTM) coordinate system.

The measurements were carried out on a computer equipped with an Intel Core i7.6700 processor running at 3.4GHz, 16GB RAM, an NVIDIA GTX 1070 graphics card with 8GB RAM, and Windows 10 operating system (OS). The ECC algorithm was implemented using the Open Source Computer Vision Library (OpenCV) in a C++ application. The accuracy of the matching process was assessed using the normalized CC metric, which was defined as shown in Equation 5.1. The CC metric produces a coefficient $\rho \in [0, 1]$ that measures the similarity between two images - the source and the template. Although it is not a perfect measure of the similarity between images, it estimates the intensity difference between two images and is useful for evaluating the matching accuracy. Finally, the images to be compared were randomly selected from the four datasets.

$$CC = \frac{\sum_{x=1}^{w} \sum_{y=1}^{h} T_{x,y} \cdot I_{x,y}}{\sqrt{\sum_{x=1}^{w} \sum_{y=1}^{h} T_{x,y}^2 \cdot \sum_{x=1}^{w} \sum_{y=1}^{h} I_{x,y}^2}} \tag{5.1}$$

with w and h being the width and height of both T and I images.

The response time and obtained accuracy were checked against several configurations with the following parameters:

- **Precision to converge**. The closer it gets to zero, the harder is for the algorithm to converge.
- **Number of iterations**. A larger number of iterations helps to give some room for the algorithm to converge.
- **Image size**. A smaller dimensionality helps to decrease the response time and achieve faster convergence, and it does not necessarily lead to worse matching results.

5.5.1 Multispectral registration

To evaluate the accuracy of the proposed image-matching methodology, four experiments were conducted using the four multispectral bands. The initial normalized CC values are presented in Table 5.1 to demonstrate the

improvement achieved by the four image-matching experiments. Since the dimensionality of the multispectral images was already low, downsampling was unnecessary.

Table 5.1: Normalized CC between original multispectral images.

		GRE-RED	GRE-NIR	GRE-REG
Dataset	Viewpoint	ρ	ρ	ρ
1	1	0.8954	0.8777	0.9306
	4	0.9777	0.9381	0.9618
	6	0.9756	0.9520	0.9727
2	37	0.9286	0.9100	0.9371
	48	0.9265	0.8918	0.9316
	58	0.9146	0.8878	0.9242
	53	0.9777	0.9381	0.9618
3	9	0.9669	0.9545	0.9714
	35	0.9578	0.9453	0.9739
	78	0.9707	0.9504	0.9672
	107	0.9602	0.9537	0.9686
	136	0.9755	0.9612	0.9692
	141	0.9732	0.9551	0.9744
	142	0.9728	0.9375	0.9656
	144	0.9731	0.9316	0.9622
	Average	**0.9564**	**0.9323**	**0.9581**

The first experiment used parameters that were close to the optimal configuration but resulted in a higher response time. The results in Table 5.4 indicate that this approach achieved good alignment for most images, but not for those with significant intensity differences among several spectral bands, such as human-made objects. Nonetheless, the images were well-aligned according to visual inspection. The second test aimed to decrease the response time at the expense of a lower CC. The third experiment used an intermediate configuration that achieved a high CC while also reducing the latency. Finally, the fourth experiment downscaled the image size by $1/3$, achieving a higher CC with a lower response time. However, as noted by visual inspection, this approach returned worse results for some images.

The four experiments were configured as follows, where n is the number of iterations and p is the aimed precision. Test 1: $n \leftarrow 30, p \leftarrow 1^{-10}$, test 2: $n \leftarrow 15, p \leftarrow 1^{-2}$, test 3: $n \leftarrow 15, p \leftarrow 1^{-10}$ and test 4: $n \leftarrow 150, p \leftarrow 1^{-60}$.

5.5.2 RGB and multispectral image-matching

Table 5.4 presents the outcomes of three experiments to match images captured from RGB and multispectral datasets. Note that the observed intensity changes are even more significant in RGB and multispectral imagery, mak-

Table 5.2: Normalized correlation coefficient and response time on the matching of a random subset of multispectral images over four different tests.

		Test 1				Test 2			
Dat.	Viewpoint	GRE-RED ρ	GRE-NIR ρ	GRE-REG ρ	#time (s)	GRE-RED ρ	GRE-NIR ρ	GRE-REG ρ	#time (s)
1	1	0.9545	0.9186	0.9403	3.627	0.9545	0.9181	0.9395	1.128
	4	0.9888	0.9473	0.9681	3.576	0.9814	0.9473	0.9681	0.950
	6	0.9861	0.9591	0.9772	3.622	0.9817	0.9591	0.9767	1.079
2	48	0.9654	0.9107	0.9368	3.580	0.9654	0.9107	0.9368	1.282
	58	0.9546	0.9088	0.9321	3.607	0.9547	0.9088	0.9321	1.224
	53	0.9888	0.9473	0.9681	3.573	0.9814	0.9473	0.9681	0.955
	37	0.9628	0.9265	0.9461	3.569	0.9463	0.9265	0.9460	1.100
3	35	0.9973	0.9654	0.9785	3.543	0.9973	0.9645	0.9766	1.090
	107	0.9973	0.9621	0.9776	3.558	0.9973	0.9621	0.9776	1.356
	136	0.9973	0.9677	0.9767	3.577	0.9781	0.9623	0.9760	0.581
	78	0.9957	0.9640	0.9731	3.575	0.9956	0.9615	0.9696	1.353
	142	0.9962	0.9632	0.9749	3.567	0.9772	0.9632	0.9748	0.989
	144	0.9964	0.9580	0.9715	3.586	0.9781	0.9580	0.9715	0.994
	9	0.9967	0.9713	0.9810	3.585	0.9967	0.9713	0.9810	1.471
	141	0.9966	0.9746	0.9791	3.518	0.9967	0.9713	0.9810	1.110
	Average	**0.9849**	**0.9496**	**0.9654**	**3.5775**	**0.9788**	**0.9488**	**0.965**	**1.1108**
	Initial results	0.9564	0.9323	0.9581		0.9564	0.9323	0.9581	

Table 5.3: Continuation of Table 5.2.

		Test 3				Test 4			
Dat.	Viewpoint	GRE-RED ρ	GRE-NIR ρ	GRE-REG ρ	#time (s)	GRE-RED ρ	GRE-NIR ρ	GRE-REG ρ	#time (s)
1	1	0.9545	0.9186	0.9403	1.933	0.9540	0.9181	0.9394	1.797
	4	0.9866	0.9474	0.9682	1.900	0.9883	0.9473	0.9665	1.780
	6	0.9838	0.9592	0.9772	1.927	0.9849	0.9610	0.9767	1.818
2	48	0.9654	0.9107	0.9367	1.884	0.9651	0.9113	0.9368	1.746
	58	0.9546	0.9088	0.9321	1.994	0.9651	0.9113	0.9368	1.774
	53	0.9866	0.9474	0.9682	1.921	0.9883	0.9473	0.9665	1.760
	37	0.9628	0.9265	0.9461	1.895	0.9618	0.9266	0.9455	1.774
3	35	0.9973	0.9654	0.9785	1.905	0.9972	0.9659	0.9778	1.805
	107	0.9973	0.9621	0.9776	1.920	0.9970	0.9618	0.9762	1.788
	136	0.9966	0.9645	0.9766	1.900	0.9969	0.9671	0.9752	1.761
	78	0.9957	0.9640	0.9731	1.903	0.9948	0.9633	0.9714	1.798
	142	0.9941	0.9632	0.9749	1.909	0.9961	0.9632	0.9740	1.773
	144	0.9909	0.9580	0.9715	1.898	0.9963	0.9580	0.9702	1.774
	9	0.9967	0.9713	0.9810	1.911	0.9964	0.9713	0.9806	1.789
	141	0.9966	0.9746	0.9791	1.873	0.9965	0.9750	0.9786	1.789
	Average	**0.9839**	**0.9493**	**0.9654**	**1.9048**	**0.9852**	**0.9499**	**0.9648**	**1.7817**
	Initial results	0.9564	0.9323	0.9581		0.9564	0.9323	0.9581	

Table 5.4: Normalized CC and response time on the matching of a random subset of multispectral and RGB images from the fourth dataset.

	Test 1		Test 2		Test 3	
Viewpoints	ρ	#time (s)	ρ	#time (s)	ρ	#time (s)
1, 916	0.9683	8.090	0.9683	2.810	0.9569	7.405
148, 918	0.9664	7.882	0.9664	2.839	0.9408	7.022
142, 904	0.9711	7.506	0.9711	2.757	0.9453	7.058
137, 892	0.9761	7.221	0.9761	2.718	0.9747	7.448
144, 906	0.9690	7.266	0.9690	2.724	0.9680	7.087
Average	**0.970**	**7593**	**0.9701**	**2.769**	**0.9571**	**7.204**

ing image-matching more challenging and increasing the response time. The second test achieved the lowest latency, while the first attained the same results with an increased response time, indicating the ECC algorithm's early convergence. Selecting a large number of iterations that ensures convergence is a safe approach, while a real-time application may prefer a lower number of iterations at the expense of not converging. Both tests utilized images with a size equivalent to half the multispectral dimensionality. Such size, together with image blurring, enables image smoothing and better image-matching. The third test used the original multispectral size and downscaled RGB images to fit them, returning worse results than previous tests since the number of iterations and precision were reduced to avoid high latency.

The three experiments were configured as follows, where n is the number of iterations, p is the aimed precision and d is the image size. Test 1: $d \leftarrow d_{mult.}/2, n \leftarrow 400, p \leftarrow 1^{-60}$, test 2: $d \leftarrow d_{mult.}/2, n \leftarrow 150, p \leftarrow 1^{-30}$ and test 3: $d \leftarrow d_{mult.}, n \leftarrow 100, p \leftarrow 1^{-20}$.

5.5.3 Thermal and RGB image-matching

Three experiments were conducted with the same parameters as before (Table 5.5). Although reducing the image dimensionality did not have a significant impact on the results due to edges being smoothed out in IR images, the CC values were lower than in previous tests since some registration results had areas with null values, as shown in Figure 5.9. The first and third experiments, which used images with smaller dimensionality, reached the best CC and returned results that were accurate by visual inspection. The second experiment, which used the original thermographic size, obtained almost the same CC at the cost of a slightly higher response time. In contrast with the multispectral tests, the first and third experiments used smaller dimensionality images, but their results were not as inaccurate by visual inspection. Thus, it can be concluded that thermal imagery can be aligned with RGB data by downscaling the dimensionality of both to speed up convergence.

The three experiments were configured as follows, where n is the number of iterations, p is the aimed precision and d is the image size. Test 1: $d \leftarrow d_{thermal}/2, n \leftarrow 300, p \leftarrow 1^{-60}$, test 2: $d \leftarrow d_{thermal}, n \leftarrow 100, p \leftarrow 1^{-20}$ and test 3: $d \leftarrow d_{thermal}/2, n \leftarrow 60, p \leftarrow 1^{-20}$.

Table 5.5: Normalized CC and response time on the matching of a random subset of RGB and thermal images over three different tests.

		Test 1		Test 2		Test 3	
Dataset	Viewpoint	ρ	#time (s)	ρ	#time (s)	ρ	#time (s)
3	394	0.9756	2.779	0.9771	3.204	0.9756	1.419
	626	0.9728	2.834	0.9745	3.320	0.9728	1.389
	726	0.9728	2.951	0.9741	3.211	0.9728	1.411
	928	0.9756	2.761	0.9770	3.341	0.9756	1.401
	467	0.9416	2.731	0.9385	3.134	0.9417	1.381
	643	0.9511	2.789	0.9494	3.137	0.9511	1.404
	963	0.9444	2.881	0.9425	3.177	0.9444	1.403
	475	0.9396	2.946	0.9358	3.257	0.9396	1.390
4	922	0.9674	2.791	0.9690	3200	0.9674	1.396
	554	0.9786	2.790	0.9803	3.267	0.9786	1.402
	405	0.9266	2.716	0.9265	3.187	0.9266	1.400
	607	0.9511	2.794	0.9502	3.236	0.9511	1.382
	Average	**0.958**	**2.8135**	**0.9579**	**3.222**	**0.9581**	**1.398**

5.6 Conclusions and future work

In this chapter, a novel framework for accurately matching images collected from multiple remotely sensed datasets was proposed. The images were acquired from UAS platforms to construct a multi-layer model with a high LOD. To enable the registration of multiple images, the Enhanced Correlation Coefficient (ECC) algorithm was proven to be effective for images that have significant colourimetric differences, resulting from being sensitive to distinct spectral wavelengths. The proposed methodology can be applied to a wide range of remote sensing applications where visible, thermal and multispectral images are used.

In addition, a case study was conducted to demonstrate the effectiveness of the proposed methodology in identifying individual trees from matched multispectral images. This capability is crucial for various applications such as forest management and environmental monitoring. The results show that the proposed framework can accurately match multispectral images with varying degrees of colourimetric differences and enable high-resolution shape reconstruction of individual trees. The proposed methodology has the potential to be applied to various other remote sensing applications that require accurate and high-resolution multispectral image matching.

As a future work, this framework could be extended to images with larger differences, particularly in terms of geometry. For instance, changes in consecutive remotely sensed images may be smaller than those obtained from terrestrial surveys, requiring affine and homography transformations to approach the image-matching problem. An adaptive algorithm that starts from a very reduced size and gradually increases it until reaching the original size could be evaluated to work with sparser images collected at different times. This approach could help to find an approximate transformation with a lower response time, whereas using a higher dimensionality enables estimating more fine-grained transformations. Additionally, exploring different

techniques such as deep learning-based methods could be investigated to enhance the accuracy of the image-matching process. Furthermore, the proposed framework can be applied to various applications such as land-use classification, crop health monitoring, and forest inventory, among others.

Part III

Point cloud enrichment

This chapter comprises the generation of thermal and multispectral point clouds. Remotely sensed data is projected over large RGB point clouds to obtain denser point clouds from low-resolution imagery. The evolution of this approach is shown throughout this chapter; the first proposal is solved in a sequential manner, whereas the methodology of the last work was completely implemented in the GPU.

Generation of thermal point clouds

6

About this chapter

This chapter describes the generation of dense thermographic point clouds by mapping image datasets into large RGB point clouds. The GPU hardware is partly used to accomplish this time-consuming task with low response time, outperforming currently available approaches for generating thermal point clouds. Occlusion and visibility tests are evaluated for projecting accurate thermal data, together with the use of penalty and aggregation functions. Finally, the generated point cloud is applied to the thermal characterization of vegetation. The complete workflow of this methodology is shown in Figure 6.1.

The proposed methodology offers several advantages over the current state-of-the-art in thermal point cloud reconstruction. It is implemented from scratch, allowing for low-level optimizations and taking advantage of GPU capabilities for faster processing. The methodology begins by reconstructing a baseline point cloud using SfM-MVS and high-resolution RGB imagery, resulting in large and dense point clouds as input. The next step involves registering the co-acquired RGB and thermal image datasets, followed by projecting the 3D point cloud into the RGB-thermal image space, taking into account the geometric properties of the scene. Additionally, the methodology evaluates several aggregation functions for each point to minimize dissimilarity between aggregated values (in 3D points) and the initial image samples. The end result is a dense thermal point cloud that retains important image details.

The full source code is available at
`https://github.com/AlfonsoLRz/RGBThermalFusion`.

6.1 On the generation of thermal point clouds . . 109
6.2 Estimation of RGB point clouds 110
6.3 Image-matching 111
6.4 Thermal image mapping 113
6.5 Occlusion 114
6.5.1 Visibility test 114
6.5.2 Occlusion test 115
6.6 Visualization 116
6.7 Aggregation of image samples 118
6.8 Thermal characterization of vegetation 119
6.9 Results and discussion 121
6.9.1 Study area 122
6.9.2 Performance of point cloud reconstruction . . . 122
6.9.3 Number of points 123
6.9.4 Area and point density . . 124
6.9.5 Absolute and normalized response time 124
6.9.6 Aggregation of thermal data 125
6.9.7 Analysis of thermal point cloud 128
6.9.8 Visualization of the point cloud 128
6.10 Conclusions and future work 129

6.1 On the generation of thermal point clouds

Infrared radiation provides valuable data that enable detecting and describing the scene surfaces. Furthermore, the acquisition of high-resolution UAS-based imagery provides an opportunity to estimate highly precise three-dimensional models from a scene, such as point clouds or DSMs. 3D models in Remote sensing are mainly achieved by using laser sensors, e.g., LiDAR [43], and photogrammetry. Furthermore, 3D representations have been gaining interest; firstly, the interaction with 3D environments speeds up the tasks of human operators such as detecting building failures [60]. Also, the results of 3D reconstructions are much more valuable and informative for the client counterpart. Finally, it reduces the analysis of multiple datasets into a single one where comparisons are effectively performed. Among 3D representations, point clouds have been significantly preferred

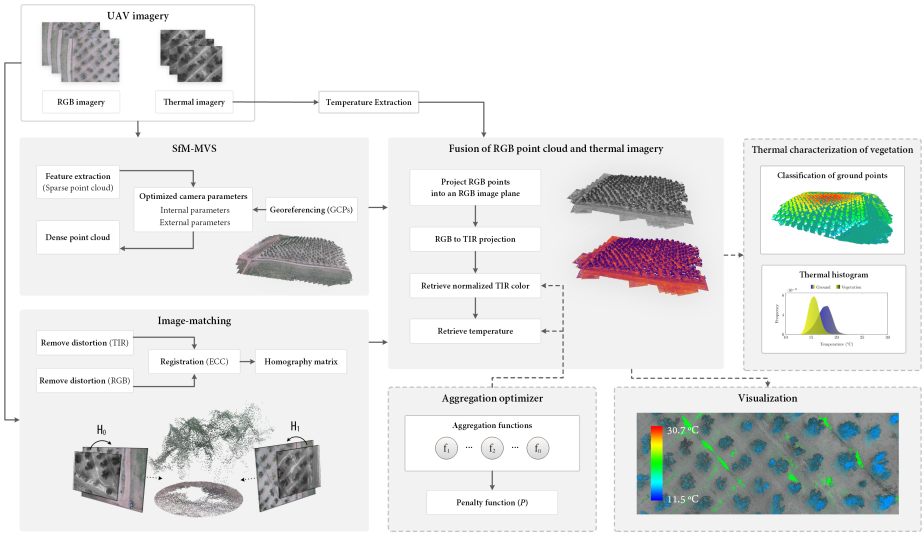


Figure 6.1: Summary of the procedure of this work. SfM-MVS and image registration stages build isolated results and thus can be performed in parallel. Then, thermal data is linked to foreground 3D points using aggregation operators that reduce the dissimilarity from the initial samples to the aggregated value. The visualization of thermal point clouds is improved by highlighting outliers, and finally, the vegetation is identified and characterized with the observed temperature.

over mesh models as they do not involve the reconstruction of surveyed surfaces [383] into polygons. Instead, a discretized version is provided, though its rendering has been proved to achieve nearly mesh-like results with modern hardware and large point clouds [122].

As previously stated in Section 2, SfM relies on pixel data for finding key points visible in several images. Therefore, estimating 3D point clouds from low-resolution thermal imagery is challenging. First, these present more substantial noise in comparison with RGB images [63] as well as aberration-induced blurring which causes the spreading of the object radiance. Consequently, the number of extracted tie points in thermography images is smaller, and the resulting 3D point clouds are much sparser and less accurate [36]. The outcome is proven to improve if further calibration is considered [384]; however, identifying GCPs in thermography may be challenging because of the material appearance and radiance spreading [58]. All these shortcomings also worsen the performance of naive 3D registration algorithms, such as ICP.

Table 6.1 shows the nomenclature of the parameters used in this chapter. Some of them are calculated by the SfM-MVS algorithm, whereas others are extracted from the device specifications.

6.2 Estimation of RGB point clouds

RGB point clouds were estimated with SfM-MVS using high-resolution RGB images. Pix4Dmapper was used in this stage, and the reconstruction was further optimized with the marking of GCPs. The density and storage

volume of the result depends on the image resolution, and therefore, it can be adjusted to estimate results with lower density. This factor is relevant as it will be required for the later visibility tests.

Table 6.1: Parameters obtained from the calibration of RGB and thermal imagery, as well as from their metadata.

Parameter	Definition
Image size (w_{image}, h_{image})	Initial image size.
Principal point (c_x, c_y)	Intersection of the principal axis and the image plane.
Focal length (f_x, f_y)	Distance in pixels between the centre of projection and the image plane.
Sensor width and height (w_{sensor}, h_{sensor})	Size in millimeters (mm).
ω, ϕ, κ	Rotation between the image coordinate system and the world system.
Camera position (t_{local})	Camera position in the point cloud system.
World offset (t_{world})	Offset between the local coordinate system and the UTM system.
Camera matrix (K)	Calculated from previous parameters as $\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$
Rotation matrix (R)	Composite matrix calculated as $R(\omega)R(\phi)R(\kappa)$.
Projection matrix (P)	Distortion-free projection in a pinhole model: $K \cdot \left[R \middle\vert -Rt_{local}\right]$.
Radial distortion coefficients (k_1, k_2, k_3)	Distortion coefficients, mostly visible on straight lines.
Tangential distortion coefficients (p_1, p_2)	Misalignment of device lens with respect to the image plane.

6.3 Image-matching

The fusion of RGB point clouds with thermal data can be performed using the previously proposed image-matching methodology. This approach tackles challenges derived from estimating thermal point clouds directly from TIR imagery. Another advantage is that processing tasks related to 3D point clouds, such as the marking of GCPs, are carried out only once. To this end, 3D RGB points are projected into the RGB image plane as shown in Equation 6.1:

$$\begin{aligned} \left[x', y', z'\right]^\intercal &= P \left[x, y, z\right]^\intercal \\ \left[x'', y''\right]^\intercal &= \frac{1}{z'} \left[x', y'\right]^\intercal \end{aligned} \tag{6.1}$$

provided that $\left[x, y, z\right]^\intercal$ is expressed in the local coordinate system where both the point cloud and the cameras are defined. Otherwise, the following expression is applied if it is defined in a global coordinate system such as UTM:

$$\left[x, y, z\right]^\intercal = \left[x_{global}, y_{global}, z_{global}\right]^\intercal - t_{world} \tag{6.2}$$

In the proposed approach, every 3D point is associated with two coordinates, x'' and y'', for each image. To avoid operating with large numeric values, t_{world} and t_{local} are defined as an arbitrary shift of real-world points and camera positions, respectively.

The matching of RGB and thermal datasets is following achieved using ECC, and the misalignment between RGB and thermal images is described by means of homography matrices, H_i, of size 3×3. This procedure starts by cropping each RGB image. However, the homography matrix H_i also involves rotations and therefore, the area to be cropped must be computed using a rectangle centred at $(\frac{w_{image}}{2}, \frac{h_{image}}{2})$. It must be large enough to enclose the area of interest depicted in thermal images. With this approach, RGB images are defined as source images, while TIR images are transformed to align both of them. The computed homography matrix projects thermal images into cropped RGB images as shown in Equation 6.3, whereas the inverse matrix performs the backward projection.

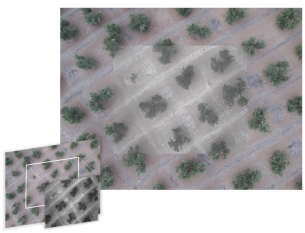


Figure 6.2: An example of image-matching between RGB and thermal images. The latter is overlapped within the RGB image and depicted with transparency ($\alpha = 0.7$). The bottom image presents the final quadrilateral shape.

In summary, $C_i \cdot {H_i}^{-1}$ projects 2D RGB points into a thermal image, provided that C_i is a composite matrix that defines the inverse cropping transformation. Note that projected points may fall out of the thermal image boundaries and a point in polygon test must be carried out. It is very simple to perform in this work since the inverse homography matrix ${H_i}^{-1}$ leads to an ideal rectangle, and thus only four Boolean operations are required.

$$
\begin{aligned}
C_i &= T_{crop} * S_{ratio} * T_{-center_{RGB}} = \\
&= \begin{bmatrix} 1 & 0 & \frac{w_{thermal_u}}{2} \\ 0 & 1 & \frac{h_{thermal_u}}{2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_x & 0 & 0 \\ 0 & r_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -\frac{w_{RGB_u}}{2} \\ 0 & 1 & -\frac{h_{RGB_u}}{2} \\ 0 & 0 & 1 \end{bmatrix} = \\
&= \begin{bmatrix} r_x & 0 & \frac{1}{2}(-r_x w_{RGB_u} + w_{thermal_u}) \\ 0 & r_y & \frac{1}{2}(-r_y h_{RGB_u} + h_{thermal_u}) \\ 0 & 0 & 1 \end{bmatrix} \\
(r_x, r_y) &= \left(\frac{crop_x}{w_{thermal_u}}, \frac{crop_y}{h_{thermal_u}} \right)
\end{aligned}
\tag{6.3}
$$

where $(crop_x, crop_y)$ are the dimensions of the cropped area, $(w_{thermal}, h_{thermal})$ are the dimensions of undistorted thermal images and u subscript refers to the indices of undistorted images. Figure 6.2 shows the result of the registration methodology as well as the warped quadrilateral shape.

Composite matrices C_i were calculated once and stored in binary files to avoid repeating this time-consuming operation after the first time. A few pairs of images were not aligned, and in those cases, the ECC algorithm either failed to converge or the resulting quadrilateral shape was very distorted. The second case was identified using a strict angle threshold that indicates the minimum/maximum allowed angles in the transformed rectangle. Despite some images not being aligned, 3D points were more likely to be visible in multiple images as the drone flight mission was planned with a high overlapping factor. Table 6.2 shows the percentage of thermal images which could be matched to RGB images as well as the number of 3D points that were not visible from any registered thermal image. Remark that the FOV is wider for the RGB device and thus edge points refer to points that were not visible in any thermal image, regardless of the matching results. Hence, it was observed that unregistered points were not visible from any viewpoint and all of them were located on the

a) 2D

#Images	Successfully registered
410	89.7561%

b) 3D

#Points	Not registered points	Edge points
19,161,076	18.8952%	100% of 18.8952%
98,016,324	15.5823%	100% of 15.5823%

Table 6.2: a) Percentage of pairs of RGB-thermal images that were successfully matched, and b), percentage of points which were not visible from any thermal image for two different point clouds. Edge points refer to points that would not be visible even if the complete image dataset was successfully registered. A large radius ($r \leftarrow 80$m) was utilized during the search of candidate points to avoid omitting visible values.

edges. For larger point clouds, the number of edge points grows unevenly with respect to the rest of the scenario, thus decreasing the percentage of unregistered points and edge points.

6.4 Thermal image mapping

Merging RGB and TIR imagery was performed with the previously cited projection. First, a naive approach that neither considered occlusion nor visibility was implemented. As a result, thermal data were mapped to 3D points as long as they were visible from any viewpoint. Equation 6.1 enables projecting each 3D point into the RGB image plane. Points were considered not visible from a viewpoint and thus discarded if their projected coordinates had values smaller than zero or larger than the dimensionality of RGB images. In addition, the image matching imposes another rejection condition due to the smaller thermal FOV. 2D coordinates within thermal images ($x_{thermal}, y_{thermal} \in \mathbb{R}$) were sampled using a bilinear interpolation in which surrounding pixels also affect the outcome. The weight factors were given by the distance from a neighbour to the required coordinates, ($x_{thermal}, y_{thermal}$). Finally, the projected values were aggregated using the arithmetic mean as the default approach.

The main challenge in this stage is to search for candidate points in every image. Points are not assumed to be uniformly sparsed, and therefore, using regular grids is not necessarily the best solution. Indeed, it requires a large memory footprint for fine-grained indexing and efficient searches. Instead, the point cloud was indexed with an octree optimized for searching neighbours within a radius [385]. A radius search is preferred over a quad-based search since previous corrections involve interpolations that leave more imprecise values at the boundaries. This data structure indexes 3D RGB points rather than image viewpoints; otherwise, an efficient search is not necessary due to the relatively low number of images. The search radius is significantly easier to define whether it is expressed at ground level as ($x, aabb_{center_y}, z$), given that x and z depend on the local position of the image viewpoint and $aabb_{center_y}$ is the Y-axis coordinate of the central point from the scene's AABB. Figure 6.3 compares the described data structure and a conventional octree. Finally, Figure 6.4 shows a top view of the reconstructed thermal point cloud.

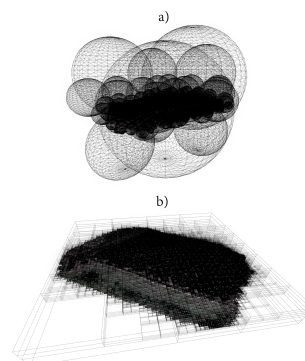


Figure 6.3: Rendering of a) an octree whose nodes are spheres, optimized for searches within a radius, and b) a traditional octree whose nodes are boxes.

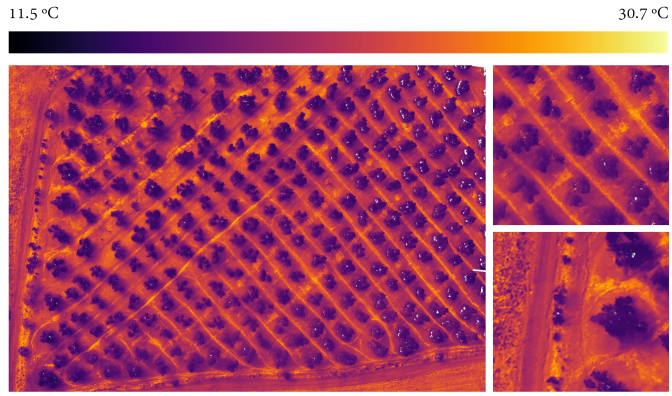


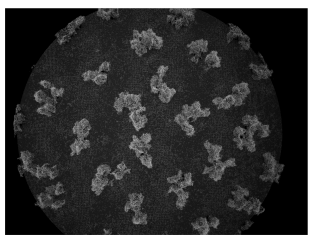
Figure 6.4: Top view of a thermal point cloud captured by a *nadir* camera. Absolute temperature values are normalized and rendered with the colour ramp on the top side. The search radius was fixed to 30 m.

6.5 Occlusion

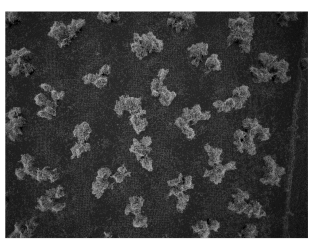
Although the previous methodology presents a valid and comprehensible approach for reconstructing thermal point clouds, it is theoretically inaccurate. To obtain a more precise projection, occluded points must not obtain data from foreground objects. Occlusion tests have been previously solved by ray-casting voxelized environments; however, this solution lacks accuracy and flexibility since octrees are mainly constructed with top-down methodologies. Therefore, the size of a voxel depends on the subdivision of the point cloud's AABB, either using a maximum depth or a maximum number of primitives in a single node. At least two approaches are following proposed to solve this challenge. Both lead to point clouds where the amount of thermal information to be aggregated over each 3D point is lower, although these are ensured to be more valuable and precise.

6.5.1 Visibility test

Images are discrete representations from where the SfM-MVS algorithm can estimate at least one 3D point from every RGB pixel. A depth buffer (or z-buffer) of the same size as the resolution used during the point cloud estimation can be constructed for each image. Still, this size can be increased to build a sub-pixel level method. In this approach, the index of the closest 3D point is stored for every viewpoint and pixel, along with the minimum depth (z). The latter determines if a point could be visible. Depth buffers were represented as sparse matrices to reduce the memory footprint and implemented as a hash map that stores pairs of values. The keys of such a hash map were the pixel indices and the values were the nearest point observed at such a position. A depth buffer is depicted in Figure 6.5, showing a circular pattern as a result of the radius neighbour search. The workflow for determining which points are visible is presented in Algorithm 1.



a)



b)

Figure 6.5: Depth buffer of a camera during a visibility test carried out with a radius of size a) 20 m and b) 30 m.

Algorithm 1 Visibility test to project thermal data into an RGB point cloud.

Input RGB point cloud: P
Input RGB-thermal image pairs: c
Input RGB resolution during SfM-MVS procedure: (w_{SfM}, h_{SfM})
Input Search radius: R
Output Array of normalized and absolute temperature
for Every image pair c_i captured from the position p_{c_i} **do**
    Create a new buffer of size (w_{SfM}, h_{SfM}) with $d_{x,y} \leftarrow \infty$, provided that $0 \leq x < w_{SfM}$ and $0 \leq y < h_{SfM}$.
    Retrieve candidate points from the octree with a radius R
    for Every 3D candidate point p_j **do**
        Calculate 2D TIR point (x, y) (Eq.: 6.1)
        if (x, y) in TIR polygon **then**
            if $\|p_{c_i} - p_j\| < d_{x,y}$ **then**
                Add point to the depth buffer and update distance at $d_{x,y}$
            end if
        end if
    end for
    for Every point p_j stored in depth buffer **do**
        Accumulate normalized and absolute temperature from c_i in p_j
    end for
end for
Aggregate results for each point in P

6.5.2 Occlusion test

Occlusion tests can be performed with ray-casting as long as the target scenario is represented as a triangle mesh. However, estimating the mesh from a point cloud is challenging. Multiple algorithms are found in the bibliography; some require the normal vector, while others are robust enough to work without surface orientation, e.g., the Advancing Front reconstruction [386]. However, mesh reconstructions are time-consuming and rely on multiple parameters whose values are not trivial to define. Additionally, reconstructing vegetation is more likely to fail since modelling leaves is very intricate. Figure 6.6 shows the result of the Advancing Front reconstruction for a subset of RGB points. Faces were oriented considering that the viewpoint is above the scene. Hence, faces were flipped if $\hat{n} \cdot \widehat{(v - p)} > \frac{\pi}{2}$, provided that $\hat{n}$ is the surface normal, v is an image viewpoint and p is a 3D point over the triangle mesh surface. A schematic representation of the occlusion methodology is depicted in Figure 6.7.

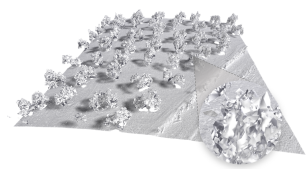


Figure 6.6: Triangle mesh estimated for a point cloud subset through Advancing Front algorithm. Reconstruction of vegetation is clearly erroneous in comparison with the ground.

Otherwise, using point clouds requires representing each point with a volumetric shape to check whether it is occluded or not. This assumption helps to identify occlusion with ray-casting, and despite being time-consuming, there are data structures optimized for rapidly discarding large parts of the scene, e.g., BVHs. In this work, volumetric points are modelled using spheres as an efficient alternative to viewpoint-oriented boxes. The radius was computed as a factor relative to the viewpoint altitude (Figure 6.7), as shown in Equation 6.4. It is an efficient yet flexible approach, whereas other shapes, such as an OOBB, require more time-consuming intersection tests.

$$R = \frac{\frac{a \cdot w_{sensor}}{f_x \cdot w_{image}}}{2} = \frac{\frac{a \cdot h_{sensor}}{f_y \cdot h_{image}}}{2} = \frac{GSD}{2} \tag{6.4}$$

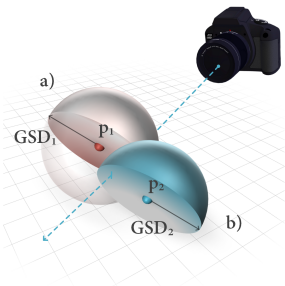


Figure 6.7: Occlusion test with two points represented as spherical volumes. a) Point where the ray impacts first (p_1), occluding the point p_2 (b).

with *GSD* being the size of a pixel if the camera is at an altitude of *a*. *R* is defined in m/pixel, despite w_{sensor} and f_x being defined in mm.

Spherical volumes change for each 3D point and image viewpoint, and therefore, a different BVH must be constructed for every viewpoint and set of candidate points. However, these were constructed in the GPU as proposed by Meister and Bittner [146]. Primitives were ordered using their Morton codes (30 bits representing *xyz* coordinates) and subsequently merged by efficiently searching the nearest neighbour, which also happens to be in a close index of the sorted buffer. On the other hand, the ray-casting was also solved in the GPU by generating a ray for each 3D point; rays which were cast towards occluded points intersected a closer volume, and vice versa (Figure 6.7). The pseudocode of this methodology is presented in Algorithm 2.

Algorithm 2 Occlusion test to determine the temperature of sphere-shaped RGB points with adaptive radius.

```
Input RGB point cloud: P
Input RGB-thermal image pairs: c
Input Search radius: R
Output Array of normalized and absolute thermal values.
for Every image pair c_i captured from the position p_{c_i} do
    Retrieve candidate points from the octree with a radius R
    Build a new BVH with the array of candidate points
    for Every 3D candidate point p_j do
        Cast a ray towards p_j and find the nearest intersected volume
        if Intersected volume is volume_{p_j} then
            Calculate 2D TIR point (x, y) (Eq.: 6.1)
            if (x, y) in TIR polygon then
                Accumulate normalized and absolute temp. from c_i in p_j
            end if
        end if
    end for
end for
Aggregate results for each point in P
```

6.6 Visualization

Javadnejad et al. [58] visualized thermal point clouds by linearly interpolating RGB and thermal data. RGB colours were converted to grayscale and temperature values were mapped to an RGB value through a colour-ramp function. The weight, $w(t)$, was computed as the distance of each temperature sample, t, to the mean, $\bar{t}$, so that $w(t) = 1$ omits RGB colours. Although it accentuates cold and hot regions ($w(t) \leftarrow 1$), intermediate values were also merged with grayscale colours. Therefore, the representation of non-anomalous temperature values distorted the colour distribution of the RGB point cloud and did not help with the detection of outliers.

Instead, outlier values can be identified using a conventional numeric outlier method. To this end, thermal values were sorted in the GPU to retrieve first and third quartiles, Q_1 and $Q3$, as well as the interquartile range ($IQR \leftarrow Q3 - Q1$). Values under $Q_1 - k(IQR)$ and above $Q3 + k(IQR)$,

where $k \geq 0$, were considered to be anomalous values and thus $w(t)$ was assigned to 1. The visualization of intermediate values was computed as proposed by Javadnejad et al. [58]; however, RGB colours were not converted to grayscale nor we utilized a linear interpolation. Instead, a Hermite interpolation was used to emphasize values of t close to t_{min} and t_{max}.

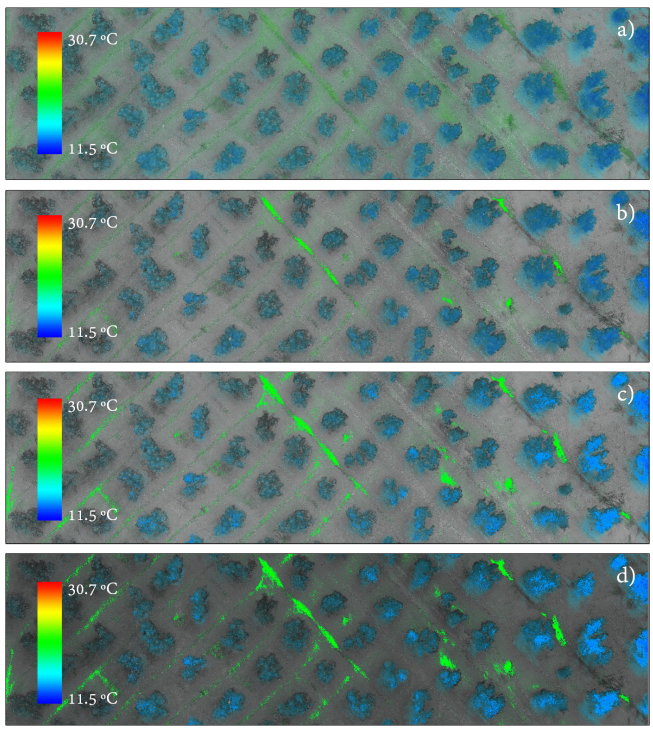


Figure 6.8: Visualization of the study area using multiple configurations. a) Visualization proposed by Javadnejad et al. [58], while b), c) and d) show our modified procedure. b) $\gamma = 0.8, k = 0.8$, c) $\gamma = 0.8, k = 0.5$, d) $\gamma = 0.5, k = 0.5$.

Equation 6.5 shows the procedure for computing the proposed enhanced visualization. Note that γ enables modifying the brightness of RGB colours, where $0 \leq \gamma \leq 1$. Figure 6.8 depicts the use of γ to darken irrelevant regions. On the other hand, $\zeta(t)$ represents a colour mapping function that is also parameterized by the temperature. In this work, $\zeta(t)$ is given by the texture depicted on the left side of every image in Figure 6.8.

$$\begin{bmatrix} r' \\ g' \\ b' \end{bmatrix} = H(w(t))\zeta(t) + \gamma(1 - H(w(t))) \begin{bmatrix} r \\ g \\ b \end{bmatrix}$$

$$w(t) = \begin{cases} 1 & t_{min} \leq t < Q_1 - k(IQR) \\ \frac{\bar{t}-t}{\bar{t}-t_{min}} & Q_1 - k(IQR) \leq t \leq \bar{t} \\ \frac{t-\bar{t}}{t_{max}-\bar{t}} & \bar{t} < t \leq Q_3 + k(IQR) \\ 1 & Q_3 + k(IQR) < t < t_{min} \end{cases} \tag{6.5}$$

$$H(w(t)) = \begin{cases} w(t) & w(t) = 1 \\ w(t)^2(3 - 2w(t)) & \text{otherwise} \end{cases}$$

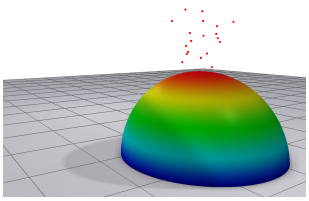


Figure 6.9: Representation of a point cloud with nineteen TIR radiation samples acquired by different viewpoints from the same point. The semi-sphere represents a Lambertian radiator.

6.7 Aggregation of image samples

3D points are mapped to several images with values that may be different from one another, despite all of them being valid. For instance, Figure 6.9 depicts nineteen radiation samples that were collected for a single point. A frequent shortcut is to average them with the arithmetic mean [58] since it is not intrusive at all. However, it is also sensitive to extreme values. Another approach is to use multiple aggregation functions that can be either utilized separately or combined. When combined, their outputs are compared to a set of inputs that help in selecting the aggregation that minimizes the outcome of a penalty function. Consequently, the penalty function determines the similarity of each aggregation to the initial thermal samples.

The definitions of aggregation operators can be considerably simplified in this work as samples are represented in 1D. Therefore, we simply defined a collection of aggregation functions such as the following:

- Minimum: $f(x) = \underset{i}{\operatorname{argmin}}\, x_i$
- Maximum: $f(x) = \underset{i}{\operatorname{argmax}}\, x_i$
- Arithmetic mean: $f(x) = \frac{\sum_{i=1}^{n} x_i}{n}$
- Geometric mean: $f(x) = \sqrt[n]{\prod_{i=1}^{n} x_i}$
- Harmonic mean: $f(x) = n \left(\sum_{i=1}^{n} \frac{1}{x_i} \right)^{-1}$

On the other hand, penalty functions are yet less intricate since their objective is simply to measure the error between the inputs and the aggregated value, y. In this regard, three penalty functions [135] were proposed for minimizing the dissimilarity of aggregations and input samples.

- $P_1(x_i, y) = \left| x_i - y \right|$
- $P_2(x_i, y) = (x_i - y)^2$
- $P_3(x_i, y) = \left| x_i - y \right|^3$

The frequency on which every aggregation function was selected was monitored for the proposed penalty functions, $P(x_i, y)$. Table 6.3 shows the preference ratio of three local penalty functions against five aggregation operators. The arithmetic mean is the preferred aggregation, followed by the geometric and harmonic mean. Maximum and minimum operators were, on the other hand, barely chosen, and their highest frequency was observed for occlusion-based approaches that handle a lower number of samples. From here, the number of operators could be reduced to make this stage more efficient by discarding aggregations which are seldom used.

Table 6.3: Preference ratio of each aggregation operator over three different penalty functions. Frequency is normalized in $[0, 1]$, and therefore the sum of each row is one. The first and second preferred choices for each configuration are highlighted in bold.

		Aggregation functions				
Penalty function	Mapping approach	Arithmetic μ	Geometric μ	Harmonic μ	Maximum	Minimum
$g_1 : \underset{y}{\text{argmin}} \sum_{i=1}^{n} \lvert x_i - y \rvert$	Naive	**0.64699**	0.07563	**0.23348**	0.02530	0.01858
	Depth buffer	**0.65672**	0.06017	**0.21589**	0.03815	0.02905
	Occlusion	**0.67193**	0.03362	**0.17048**	0.07004	0.05392
$g_2 : \underset{y}{\text{argmin}} \sum_{i=1}^{n} (x_i - y)^2$	Naive	**0.67787**	**0.15036**	0.14783	0.01312	0.01080
	Depth buffer	**0.66193**	0.13732	**0.16330**	0.02067	0.01676
	Occlusion	**0.63223**	0.10407	**0.19052**	0.03972	0.03344
$g_3 : \underset{y}{\text{argmin}} \sum_{i=1}^{n} \lvert x_i - y \rvert^3$	Naive	**0.57092**	0.14996	**0.25392**	0.01412	0.01106
	Depth buffer	**0.56471**	0.13513	**0.26142**	0.02171	0.01701
	Occlusion	**0.57647**	0.09821	**0.25101**	0.04057	0.03371

6.8 Thermal characterization of vegetation

Analyzing the temperature of individual surfaces requires segmenting the environment so these can be further studied. The collected datasets comprised a natural environment with soil as well as low and high vegetation, where only the latter two are of interest. A geometric segmentation based on the surface orientation of a point cloud is proposed in this work.

Normal vectors were first computed with several algorithms that will be following compared. Firstly, the surface orientation can be computed using a KNN search in the GPU, where the surroundings of a point contribute to its normal estimation. The KNN algorithm can also be optimized by spatially ordering the points using the Morton encoding and the radix sort algorithm. This solution has been previously explored in high-performance works [137] since it ensures neighbour searches with linear complexity. Furthermore, the complete search methodology is highly parallelizable in the GPU, thus allowing us to solve normal estimations in a few minutes at most for large point clouds. Once neighbours are found, normal vectors can be estimated using multiple methods.

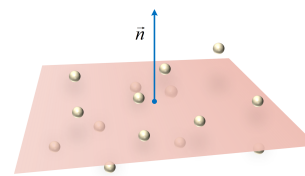


Figure 6.10: Schematic representation of normal estimation by detecting the plane that better represents a group of points.

- First, normal vectors can be computed as the average vector from multiple cross-products calculated between vectors formed by the main point and a neighbour. However, this naive approach is very sensitive to noise.
- A more robust approach is to perform the KNN search and estimate the normal via plane fitting with PCA. It enables estimating the plane that better represents a subset of points [387] using the eigenvector decomposition. As a result, the calculated normal vectors fit better the expected surface (see Figure 6.10).
- Normal estimation is also implemented in open-source libraries, such as Point Cloud Library (PCL) and the Computational Geometry Algorithms Library (CGAL). PCL provides CPU and multi-core CPU methods, whereas CGAL also implements a much slower estimation that was not included in later tests.

Figure 6.12 shows a comparison of the response time obtained by the

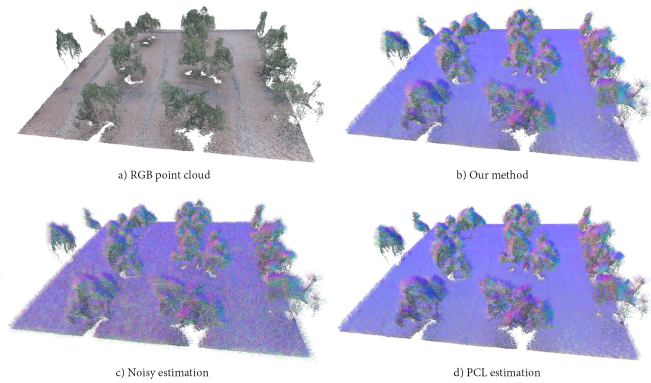


Figure 6.11: Comparison of normal estimations for a small point cloud. Normal vectors are up-scaled and not normalized to improve the visualization, whereas colours are encoded as the normal vector. The proposed method and PCL obtain similar results, although ours is more rapid for denser point clouds.

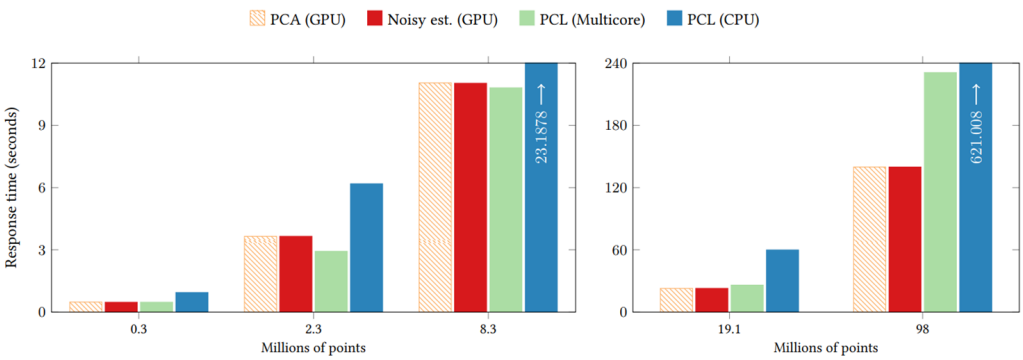


Figure 6.12: Response time, in seconds, measured in the estimation of normals from four different methods. The results were calculated as the arithmetic mean of five different measurements for each algorithm.

four explained algorithms, whereas Figure 6.11 shows the estimated normal vectors. The naive estimation provides the baseline results for GPU-based approaches since it is very fast, albeit being inaccurate. As observed, the proposed solution has a better performance with larger point clouds, whereas the processing of smaller point clouds offers minor improvements in comparison to the multi-core PCL implementation. Therefore, the proposed method is proven to be more convenient for large point clouds such as those utilized in this work.

The dot product of the estimated normal vectors, $\hat{n}$, and the Y-axis in a Cartesian coordinate system enable separating feasible ground points by defining a threshold, $\hat{n}_{threshold}$. Consequently, the complexity of subsequent stages is also reduced, and yet, some ground points remain not filtered. The following step is designed based on the fundamentals of LiDAR, where the last returns are more likely to belong to the ground. Again, points were modelled as spheres with a radius $R \leftarrow \frac{GSD}{2}$ (Equation 6.4). Still, this method may have some errors since points with a slightly higher elevation could prevent detecting surrounding ground points. These false negatives are detected using a tolerance radius, given by $k \cdot GSD$. k is ideally equal to one, i.e., $k \cdot GSD$ is equal to the diameter of spheres. This method is rapidly solved in the GPU since it first uses the dot product to determine

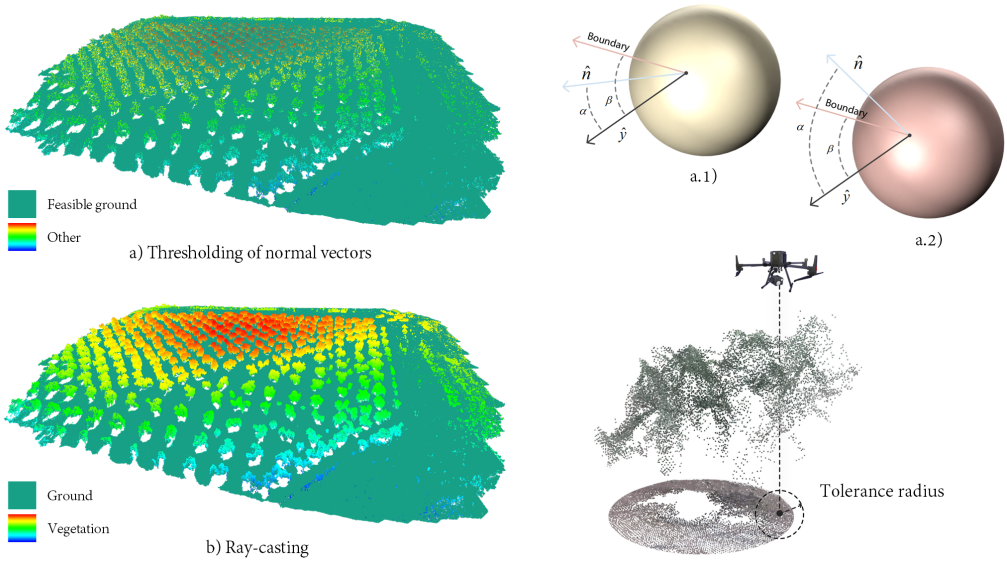


Figure 6.13: Classification of ground points with $k = 1$ and a normal threshold of 0,7. a) First step of our methodology, with candidate points being displayed as green points. a.1) Valid candidate point ($\alpha \leq \beta$), a.2) Point discarded for further processing ($\alpha > \beta$). The angles that both normal and boundary vectors form with respect to the Y-axis are defined as α and β respectively. b) Evaluation of ray-casting for each candidate point. Note that fewer points are rendered as green at this stage.

the candidate points, and then, n rays are traced from $p_i + (0, y, 0)$ to p_i, provided that n is the number of candidate points and y places the ray's source above the boundaries of the point cloud. Both stages are depicted in Figure 6.13.

Application of the proposed classification to other environments

The proposed method is robust enough for detecting feasible ground points and therefore is designed to work over other kinds of crops. Since it relies on the detection of the last return of a set of traced rays, the method can be configured using a normal vector threshold and a factor, k, to scale the sphere radius. This parameterization enables adapting the algorithm to different terrain slopes and point cloud densities.

6.9 Results and discussion

The proposed methodology has been evaluated against three different point clouds of the study area depicted in Figure 6.14, ranging from 98M to barely 300k points. The performance of the methodology was compared with two widely-used commercial software, Pix4Dmapper and Agisoft Metashape, based on three key factors: point density, computational efficiency, and similarity of colour distribution. The objective was to demonstrate the superiority of the proposed solution in the reconstruction of 3D thermal point clouds. For the comparison, the third-party software was used with

the recommended configuration for achieving the highest level of detail, both for image alignment (first phase) and dense point cloud generation (second phase). To account for the non-deterministic nature of SfM-MVS in Pix4Dmapper and Agisoft Metashape, the results were computed as the arithmetic mean of five different tests for each solution.

All measurements were performed on a PC with Intel Core i7-7700 3.6 GHz, 16 GB RAM, GTX 1070 GPU with 8 GB RAM (Pascal architecture) and Windows 10 OS. The proposed methodology is implemented in C++ along with OpenGL for rendering. Therefore, parallel algorithms were developed in GLSL (OpenGL Shading Language) using general-purpose compute shaders.

6.9.1 Study area

The case study is an olive orchard located in Mancha Real (Jaén, Spain), depicted in Figure 6.14 along with the network of GCPs. The surveyed area covers ~1.7 ha, although reconstructed areas show a larger area due to the sensors' FOV. The ground presents a variable elevation, ranging from 552 m to 572 m. Some of the trees were affected by the pathogen *Xylella fastidiosa*, whose symptoms can be revealed through thermal imagery [45]. Although this work is not focused on detecting this pathogen, it shows the relevance of estimating a thermal point cloud with reliable radiance. The dataset comprises 820 images, 410 images for each kind of image. Additionally, the network of GCPs was distributed on the boundaries as well as inside the study area, as proposed by Martínez et al. [388] with some of them being placed on points with significant altitude changes.

6.9.2 Performance of point cloud reconstruction

An RGB point cloud with 98,016,324 points was used as the main input to assess the performance of three different approaches: naive mapping, visibility-based mapping and occlusion-based mapping. Commercial software is, on the other hand, fed only with thermographic images. Two different search radii were evaluated: $r_1 \leftarrow 20$ m and $r_2 \leftarrow 30$ m, centred at $y = \frac{aabb_{min_y} + aabb_{max_y}}{2}$ for every image viewpoint, provided that $aabb_{max}$ and $aabb_{min}$ are the maximum and minimum points of the point cloud's bounding box. The visibility test is configured so that the size of depth buffers is equal to the size of the original RGB images (4000×3000 px). The occlusion experiment builds BVHs using Morton codes of 30 bits, searching neighbours with a radius of 50 and sorting the encoded values with the Radix ordering. The obtained results are summarized in Table 6.4. The point cloud area was measured by calculating its convex hull with the Delaunay triangulation [389]. The estimated area was diminished according to the area of internal gaps. Finally, it must be noted that penalty and aggregation functions were omitted during these experiments since further insight will be provided in a later section.

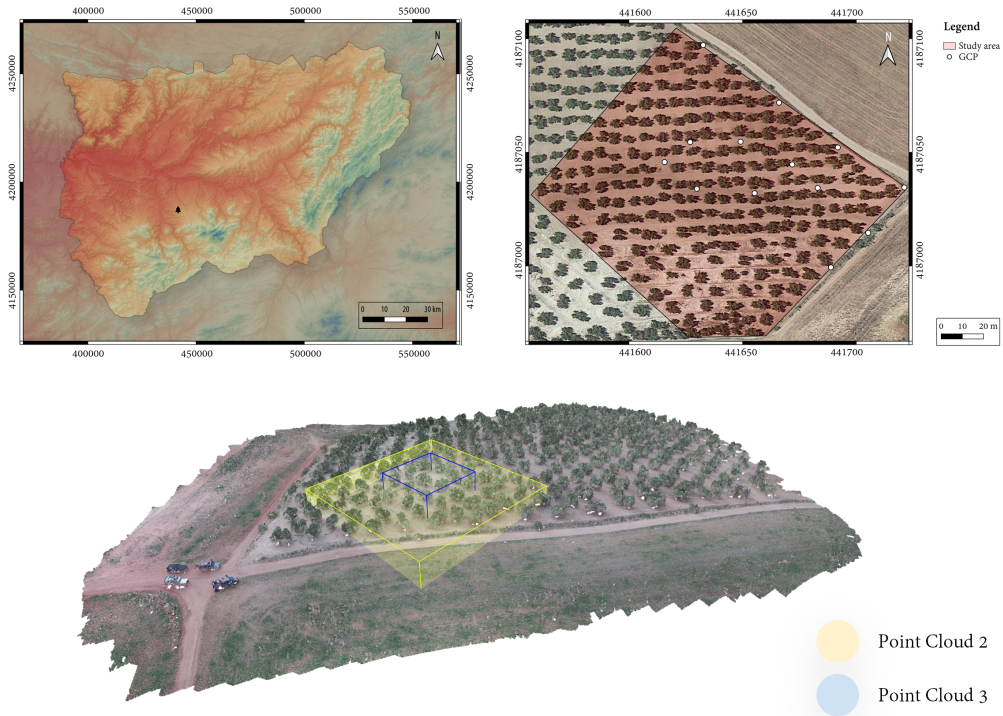


Figure 6.14: An overview of the study area. On the left side, the location of the surveyed area in the region of Jaén, whereas the right side shows the polygon of the study area. Coordinates are given in WGS84 (EPSG:4326). The image at the bottom shows three point clouds used in this work, from higher to lower size.

6.9.3 Number of points

Thermal point clouds generated with the proposed method have the highest number of points. Despite this metric not reflecting the quality of the result, the geometry was indeed more accurate as it was obtained from an RGB reconstruction optimized with the marking of GCPs. In comparison, point clouds reconstructed solely from thermal data have a significantly lower point density. Similarly, the spatial covering is also lower due to the scarce tie-points obtained by SfM, as reported in Table 6.4 and Figure 6.18. The naive approach discards ~16M points which were not visible in any thermal image, whereas the number of points using r_2 increased by 358.64% and 746.09% with respect to Agisoft Metashape and Pix4Dmapper results. Occlusion-based experiments constructed point clouds of lower size using a radius equivalent to the GSD. Still, this approach greatly improved the result of commercial software. On the other hand, incrementing the radius size ($r_2 \rightarrow r_1$) barely improved the number of points, whereas the response time notably increased. Therefore, it can be concluded that r_1 is preferred over r_2.

Table 6.4: Performance comparison of the proposed methods. The reported values are averaged over five different tests. r_1 and r_2 correspond to a search radius of 20 m and 30 m, respectively. The best results are highlighted in bold, whereas the number of mapped images shows how many TIR images were successfully projected into the point cloud. Hence, it shows the number of aligned pairs of RGB-thermal images. Third-party software also discarded some images which could not be calibrated.

	Mapping approach			
Attributes	Naive (r_1)	Depth buffer (r_1)	Occlusion (r_1)	Pix4Dmapper
Number of points	82.743.078 points	79.511.469 points	74.181.776 points	9.782.277 points
Area	1.9973 ha	1.9973 ha	1.9973 ha	1.9942 ha
Point density	**4,143 points**/m^2	3,981 points/m^2	3,714 points/m^2	461 points/m^2
Response time	**3m 36.54s**	7m 12.18s	4m 27.83s	13m 4.4s
Response time/point	**2.617** µs	5.435 µs	3.61 µs	80.185 µs
Number of mapped images	368 of 410	368 of 410	368 of 410	**371 of 410**
	Naive (r_2)	Depth buffer (r_2)	Occlusion (r_2)	Agisoft Metashape
Number of points	**82,767,436 points**	79,511,469 points	74,213,254 points	18,045,885 points
Area	**1.9991 ha**	**1.9991 ha**	**1.9991 ha**	1.7202 ha
Point density	4,140 points/m^2	3,981 points/m^2	3,712 points/m^2	1,102 points/m^2
Response time	7m 31.01s	12m 4.38s	9m 24.92s	3m 48.31s
Response time/point	5.449 µs	5.435 µs	7.61 µs	12.113 µs
Number of mapped images	368 of 410	368 of 410	368 of 410	326 of 410

6.9.4 Area and point density

The proposed algorithm generated larger results in terms of covered area. Agisoft Metashape had most of the errors at the boundaries, whereas Pix4Dmapper's results were very similar to ours. However, the latter had some gaps within the surveyed area (see Figure 6.18). As expected, a larger number of points also led to a higher point density. Hence, the highest point density, achieved by the naive mapping, increased the density by 275.95% and 798.69% with respect to Agisoft Metashape and Pix4Dmapper, respectively. As a reference, Javadnejad et al. [58] obtained thermal point clouds with a density of 270 points/m^2 (0.36 ha and 95 images), 336 points/m^2 (2.38 ha and 101 images) and 5,429 points/m^2 (0.35 ha and 165 images), while Webster et al. [42] operated with thermal datasets of 776 points/m^2. Note that the point density could be improved by increasing the image overlapping during flight, thus enlarging the dataset dimensionality.

6.9.5 Absolute and normalized response time

The naive approach had the lowest latency, whereas occlusion-based methods were also quite competitive. Remark that Pix4Dmapper and Agisoft Metashape also benefit of CUDA-compatible GPUs. In comparison, the naive and depth buffer approaches were implemented in a sequential manner, while the occlusion-based approach utilized the GPU to solve intersections. The naive approach shows a small improvement in the overall response time in comparison with Agisoft Metashape, although the normalized response time outperforms commercial software: 78.39% and 96.73% less processing time per point with respect to Agisoft Metashape and Pix4Dmapper,

respectively. On the other hand, the results from the occlusion mapping with r_1 were also remarkable, since this approach requires building a BVH for every image, each one integrating up to ~6M points (r_2 peaks on 14.5 million points).

6.9.6 Aggregation of thermal data

The accuracy of assigning thermal values to 3D points was evaluated to test the effectiveness of the proposed methodology in aggregating thermal samples acquired from different viewing angles. To assess the quality of the results, metrics such as Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and standard deviation were used to measure the distance between the aggregated values and the initial samples. However, it is important to note that global metrics are not appropriate for evaluating the aggregation results since they do not take into account the point-wise nature of the operation. Therefore, metrics that account for local errors are more suitable for this problem. The average RMSE and MAE (Equations 6.6 and 6.7) criteria are better suited for evaluating the quality of the reconstructed thermal values, as they are frequently used over unbalanced datasets with only a few items.

$$RMSE_{\mu} = \frac{\sum_{i=1}^{|P|} \sqrt{\frac{\sum_{j=1}^{|S_i|}(s_j - A_i)^2}{|S_i|}}}{|P|} \tag{6.6}$$

$$MAE_{\mu} = \frac{\sum_{i=1}^{|P|} \frac{\sum_{j=1}^{|S_i|}|s_j - A_i|}{|S_i|}}{|P|} \tag{6.7}$$

where i refers to the index of a 3D point, S_i is a set of samples linked to such a point, and A_i was the aggregated value. Besides the averaging metrics, the global RMSE and MAE were also reported in Table 6.5. RMSE and MAE equations were adapted to point clouds as shown in Equation 6.8 and 6.9):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{|P|}\sum_{j=1}^{|S_i|}(s_j - A_i)^2}{\sum_{i=1}^{|P|}|S_i|}} \tag{6.8}$$

$$MAE = \frac{\sum_{i=1}^{|P|}\sum_{j=1}^{|S_i|}|s_j - A_i|}{\sum_{i=1}^{|P|}|S_i|} \tag{6.9}$$

Likewise, the average standard deviation of the initial samples mapped to every point is shown in Equation 6.10, regardless of aggregation and penalty functions.

$$\sigma_{mean} = \frac{\sum_{i=1}^{|P|} \frac{\sum_{j=1}^{|S_i|}\left|s_j - \{\overline{s_{i_0},\ldots,s_{i_{s_i-1}}}\}\right|}{|S_i|}}{|P|} \tag{6.10}$$

Observing the definitions of penalty functions in Table 6.5, it can be

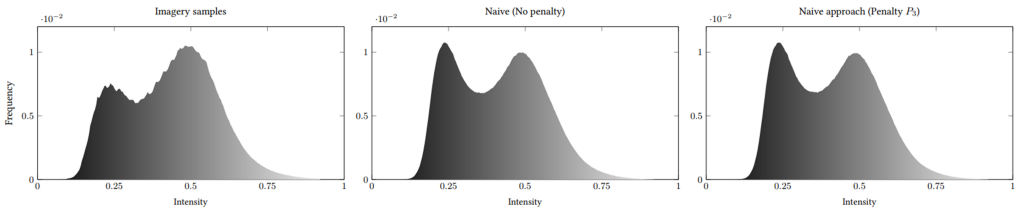


Figure 6.15: Intensity distribution of image samples collected during the reconstruction of a thermal point cloud, as well as the grayscale distribution obtained after using the naive-no penalty and naive-penalty (P_3) approaches. The density functions concerning other variants were omitted as they returned very similar results.

noted that the averaged RMSE and MAE are biased towards P_1 and P_2. However, the penalty functions aim to minimize the dissimilarity between the aggregations and initial samples, and although biases exist towards specific penalty functions, the penalty-based optimization has helped reduce the dissimilarity of the aggregations and initial samples. The approaches implementing occlusion and visibility tests were expected to have better results for averaged metrics since they have a lower number of samples. However, they obtained worse results in terms of global RMSE and MAE since the minimization is applied per point, S_i, instead of globally. The best value regarding the averaged RMSE was achieved with the occlusion approach and P_2, whereas occlusion together with P_1 lowered the dissimilarity measured by the averaged MAE. Nevertheless, penalty-based approaches achieved better results than simply averaging the samples using an arithmetic mean. P_1 minimized the absolute error (MAE_μ), while P_2 and P_3 penalized higher changes, thus minimizing $RMSE_\mu$.

Table 6.5: Dissimilarity of aggregated thermal data and initial thermal samples in a scale of [0, 1], where 0 indicates perfect similarity and 1 indicates maximum dissimilarity.

		Metric				
Penalty metric	**Approach**	Average σ	$RMSE_\mu$	RMSE	MAE_μ	MAE
No penalty	Naive	0.052	0.054	**0.052**	0.045	**0.024**
	Visibility	0.049	0.050	0.055	0.042	0.027
	Occlusion	**0.039**	**0.042**	0.054	0.042	0.054
P_1: $\lvert d(x_i, y)\rvert$	Naive	0.052	0.054	0.053	0.044	**0.024**
	Visibility	0.049	0.051	0.055	0.041	0.027
	Occlusion	**0.039**	0.042	0.055	**0.035**	0.034
P_2: $d(x_i, y)^2$	Naive	0.052	0.053	**0.052**	0.044	**0.024**
	Visibility	0.049	0.050	0.054	0.042	0.027
	Occlusion	**0.039**	**0.041**	0.053	**0.036**	0.035
P_3: $\lvert d(x_i, y)\rvert^3$	Naive	0.052	0.054	**0.052**	0.045	**0.024**
	Visibility	0.049	0.050	0.054	0.043	0.027
	Occlusion	**0.039**	**0.041**	0.053	**0.037**	0.035

Another experiment is to compare the histogram obtained from thermal images and point clouds. To this end, the distance between histograms is assessed using distance functions. Three conventional criteria were used, ranging from the Pearson CC (Equation 6.11) to the Hellinger distance (Equation 6.12) and intersection distance (Equation 6.13) [390]. From now on, we will refer to them as $d_{pearson}$, $d_{hellinger}$ and $d_{intersection}$ for the sake of simplicity. $d_{pearson}$ coefficient ranges from -1 to 1, with 1 being a perfect

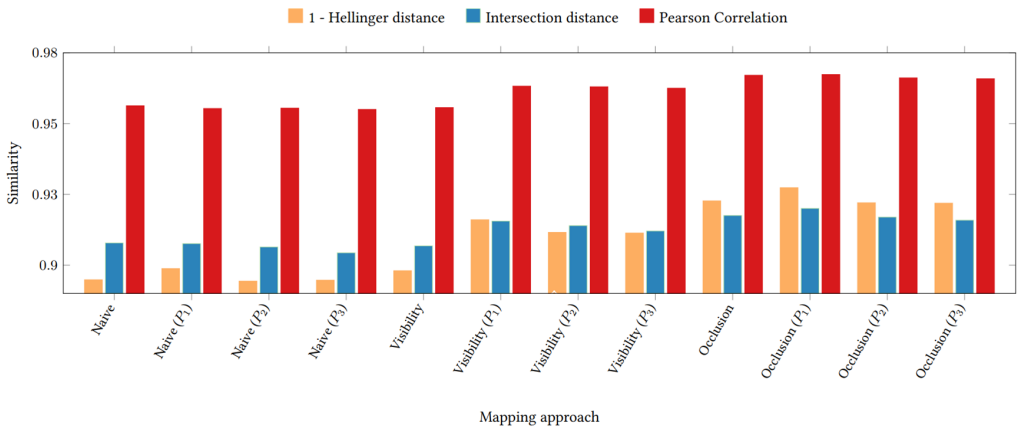


Figure 6.16: Similarity between histograms for each configuration. For the sake of clarity, Hellinger distance was expressed as $1 - d$ to provide the same interpretation for the three metrics. Hence, a value of 1 indicates that both histograms are equal ($h_{1_i} = h_{2_i}, \forall i \in [0, n)$), while 0 denotes complete dissimilarity ($h_{1_i} = 1 - h_{2_i}, \forall i \in [0, n)$).

correlation. Zero implies there exists no linear correlation and -1 indicates a perfect negative correlation. On the other hand, $d_{hellinger}$ measures the similarity of two probability functions, where 1 implies that both distributions are orthogonal. The expression of Equation 6.12 is simplified if histograms h_1 and h_2 are defined as density functions in [0, 1] ($h_{norm_{1_i}}$, $h_{norm_{2_i}}$). Finally, the intersection, $d_{intersection}$, is another widely used form of similarity for probability distributions. It returns 1 when $h_{norm_{1_i}}$ and $h_{norm_{2_i}}$ are completely overlapped.

$$d_{pearson}(h_1, h_2) = \frac{\sum_{i=1}^{n}(h_{1_i} - \overline{h}_1)(h_{2_i} - \overline{h}_2)}{\sqrt{\sum_{i=1}^{n}(h_{1_i} - \overline{h}_1)^2 \sum_{i=1}^{n}(h_{2_i} - \overline{h}_2)^2}} \tag{6.11}$$

$$\begin{aligned} d_{hellinger}(h_1, h_2) &= \sqrt{1 - \frac{1}{\sqrt{\overline{h}_1 \overline{h}_2 n^2}} \sum_{i=1}^{n} \sqrt{h_{1_i} h_{2_i}}} \\ &= \sqrt{1 - \sum_{i=1}^{n} \sqrt{h_{norm_{1_i}} h_{norm_{2_i}}}} \end{aligned} \tag{6.12}$$

$$d_{intersection}(h_1, h_2) = \sum_{i=1}^{n} \min(h_{1_i}, h_{2_i}) \tag{6.13}$$

where n is the number of bins in histograms h_1 and h_2, both with the same dimensionality.

Figure 6.15 compares the source histogram from the naive approach and the histograms obtained by mapping thermal data with no penalty function and using the penalty function P_3. The rest obtained similar results and thus were omitted in this comparison. These small variations are more clear in Figure 6.15. Remark that $d_{hellinger}$ was transformed into $1 - d_{hellinger}$ to make all of the distance functions fall in the same interval. According to the results, occlusion-based mapping obtained more similar results to the baseline histogram, especially using the occlusion test. Furthermore, the

penalty function P_1, which computes the dispersion as $|x_i - y|$, had better results for every approach.

6.9.7 Analysis of thermal point cloud

Figure 6.17 displays the temperature data collected from ground and vegetation points, based on a point cloud comprising nearly 100 million points. The filtered values are presented as a density function that is proportional to the size of the point cloud. The figure indicates that vegetation emits less thermal radiation compared to ground points, although there is a significant overlap between the two datasets. It is important to note that the vegetation label also includes dry weeds that were classified as anomalous hot regions, while ground points cover regions that are shaded by trees. The majority of thermal values range from 12 °C to 23 °C, with anomalous hot regions detected in vegetation and metal surfaces. On the other hand, cold regions belong to isolated trees and ground points.

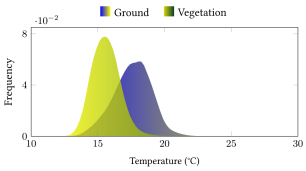


Figure 6.17: Frequency diagram of thermal radiation for 3D points labelled as ground and vegetation.

6.9.8 Visualization of the point cloud

Figure 6.18 provides a better insight into the reconstructed point clouds. Three results are shown in the figure (b, c, and d): the first one is computed using the naive approach, while the other two are the outcomes of Agisoft Metashape and Pix4Dmapper software. It is noteworthy that Pix4Dmapper uses a different grayscale distribution, as the software is capable of extracting temperature from the image dataset. Therefore, these values are normalized by considering the minimum and maximum values. As illustrated in Figure 6.18, the result of Pix4Dmapper shows empty regions that correspond to images that could not be calibrated. Moreover, canopies are poorly estimated and appear much noisier than in the RGB reconstruction, which is known to be more geometrically accurate. Additionally, trees are shown at a higher elevation in comparison with the RGB point cloud.

On the other hand, the result of Agisoft Metashape presents a notable number of errors. Firstly, the estimation of canopies is inaccurate, resulting in planar surfaces and visible outliers. Additionally, a considerable portion of the study area appears several meters below the surrounding environment. Sparsity also affects both point clouds, although it is more pronounced in the Pix4Dmapper result, as indicated in Table 6.4.

Finally, the preservation of details is demonstrated in Figure 6.19, with no visible blurring or distortion effects. The corrections, projection and aggregation operators have enabled the reconstruction of a thermal point cloud with a geometry provided by the baseline RGB point cloud.

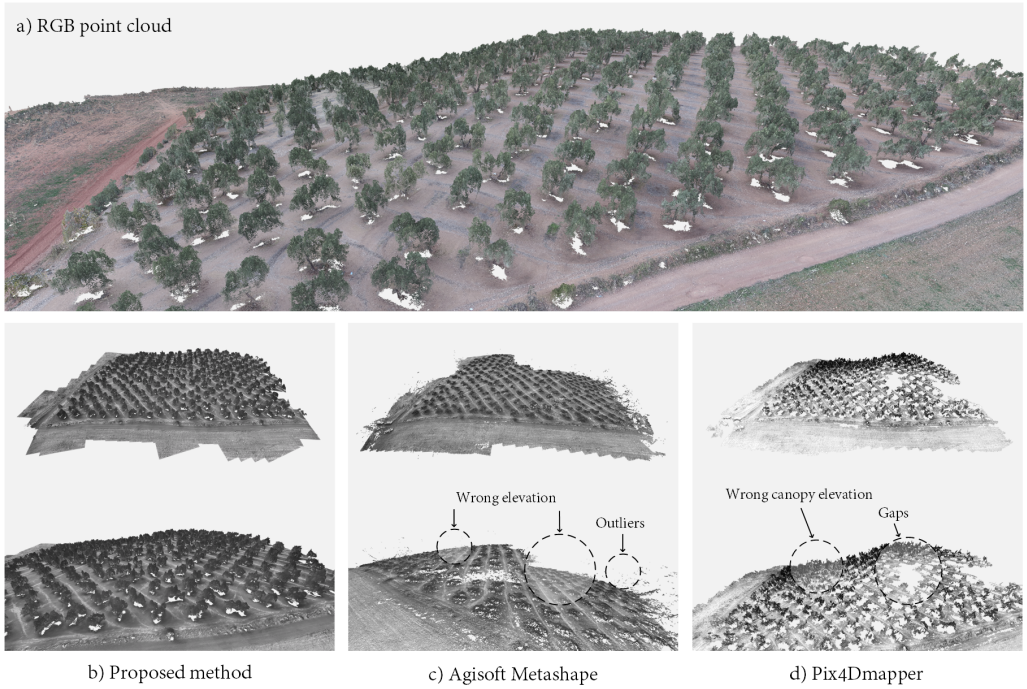


Figure 6.18: a) Original RGB point cloud, reconstructed with the original resolution, b) thermal point cloud computed using the naive approach, as well as the results obtained by c) Agisoft Metashape and 4) Pix4Dmapper.

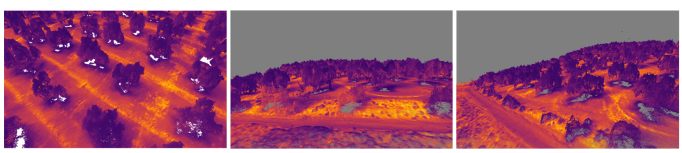


Figure 6.19: Close-up view of a thermal point cloud with ~20M points. Details of thermal images are preserved and no blurring effects are observed.

6.10 Conclusions and future work

This chapter presents an algorithm that efficiently builds dense TIR point clouds. The method employs RGB point clouds as input and produces an output with significantly more points than commercial software (up to 798.69%). GPU-based algorithms are used to accelerate some of the proposed algorithms, resulting in a 78.39% decrease in the processing time per 3D point compared to Agisoft Metashape. An innovative solution for the aggregation of multiple thermal samples is also introduced to minimize the dissimilarity between thermal values in 3D points and image intensity. Dispersion and histogram distance metrics demonstrate that penalty functions effectively aid in this task.

This proposed methodology is a promising alternative to the traditional SfM-MVS algorithm, which relies on extracting features from images. This can be challenging in thermography due to limited texture and contrast. Instead, our approach uses the RGB point cloud generated by SfM as a baseline for projecting thermal imagery, which is more readily available. Additionally, our algorithm produces denser point clouds compared to

other methods, making it better suited for applications that require high levels of detail, such as rendering software. Moreover, the use of GPU-based algorithms reduces the processing time. Overall, this methodology provides a robust and efficient solution for building dense TIR point clouds.

This chapter raises several important questions that require further investigation. One such question is whether in-situ radiometric calibration can improve the accuracy of temperature estimation. A reliable multi-temporal system could also be developed for the analysis of individual trees using the reconstructed thermal point clouds. Another potential application of this methodology is monitoring the spread of the *Xylella fastidiosa* pathogen in affected areas. Additionally, the entire workflow could be implemented in GPU to improve efficiency, which is the focus of the next chapter. The study could also be extended to compare different parallel-computing frameworks and leverage rendering tools like OpenGL's built-in z-buffer.

Aside from acceleration, there is little room for improvement in the ECC image-matching algorithm. However, thermal images were acquired so that the edges from RGB data were overlapped with infrared data. Although it may help on some datasets, these edges may be too fuzzy for vegetation and hinder the image-matching task. Disabling the use of overlapped data may therefore improve the accuracy of the matching algorithm for this type of datasets.

Generation of multispectral point clouds

7

About this chapter

In this chapter, an efficient solution for the fusion of 1) visible, 2) thermographic and 3) multispectral information acquired by UASs is proposed. Accordingly, the methodology proposed for image fusion in Chapter 5 is made effective while also being extended to 3D. Furthermore, the proposed work is developed using massively parallel algorithms, supported by GPUs and multi-core CPUs. Commercial software is used as a baseline for evaluating the efficiency of our framework. Besides parallel processing, this work is also intended for maximizing the size of point clouds by projecting images with lower resolution into 3D point clouds. The overall procedure is depicted in Figure 7.1. It extends the procedure presented in Chapter 6 by translating the projection to the GPU and optimizing the processing with sorting techniques.

Hence, the main contributions presented in this chapter are 1) the fusion of multiple imagery datasets from UASs and 2) the generation of larger point clouds than current commercial software with 3) lower response time. The accurate colour aggregation in 3D points is ensured by taking into account the occlusion of foreground surfaces. Accordingly, the proposed solution reconstructs much more dense point clouds than notable commercial software (286% on average), e.g., Pix4Dmapper and Agisoft Metashape, in much less time (-70% on average).

7.1 On the generation of point clouds augmented with multiple data sources 131
7.2 Correction and fusion of image datasets 132
7.3 Projection in 3D point clouds 133
7.4 Occlusion 133
7.4.1 OpenGL's rendering . . . 134
7.4.2 Compute shaders 135
7.4.3 CUDA 136
7.5 Point cloud sorting . . . 137
7.6 Alignment of heterogeneous data sources . . . 138
7.7 Results and discussion 140
7.7.1 Study area 141
7.7.2 Image processing 141
7.7.3 Response time 142
7.7.4 Point cloud size and normalized response time 144
7.7.5 Overall comparison 145
7.7.6 Data shuffling in GPU . . 146
7.7.7 Visual results 147
7.7.8 Binary data 148
7.8 Conclusions and future work 149

7.1 On the generation of point clouds augmented with multiple data sources

Remote sensing research concerns multiple acquisition devices and multispectral data obtained in different wavelength ranges. These are frequently visible, infrared and multispectral imagery, either collapsed into a few or hundreds of bands (hyperspectral). Thus, the main challenge of multiple sensing systems is to fuse all this information into a sole data model with multiple layers. Fusion methodologies have been long investigated to build multi-layer models that allow subsequent analyses to focus only on the data. From these models, Machine Learning [317] and Deep Learning [391, 392] represent a widespread field to extract further information. Among other factors, the raising interest in 3D models is possible due to the use of UAS in RS. In contrast to satellite imagery, UASs provide a cost-efficient solution with higher spatial and temporal resolution [314]. Nevertheless, vehicles and devices which are less prohibitive also lead to images with more defects and errors, either motivated by UAS instability [71] or the optical mechanism [393].

The main drawback of 3D point clouds with hundreds of millions of points is to process them with commodity hardware. However, they are required to provide in-depth analyses and valuable renderings [122]. As a result, the reconstruction and processing of large point clouds are computationally demanding tasks that require massively parallel algorithms and modern GPUs with increased capacity to operate many parallel threads. Therefore, solutions for fusing RS data are not solely aimed at providing accurate results, but also at efficiently generating them. The baseline for the reconstruction of 3D point clouds can be established using commercial software, with most of them taking advantage of GPUs to speed up the reconstruction [131]. Hence, their processing time is related both to the degree of parallelism as well as the size of input data. Accordingly, large point clouds are expected to worsen the performance, though they provide valuable data for mesh-like renderings and neighbourhood-derived processing (e.g., the estimation of normal vectors).

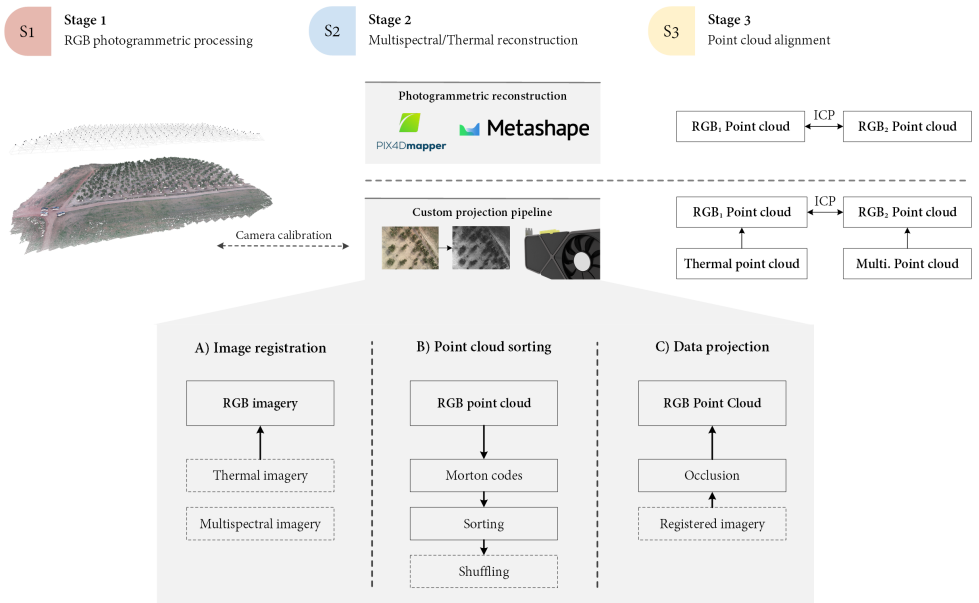


Figure 7.1: Overall procedure of the proposed method. First, the visible point cloud is reconstructed. Then, alternative data sources are projected into the previous point cloud. Finally, point clouds are aligned through the estimation of rigid transformations. The reconstruction of alternative point clouds is compared against popular commercial solutions.

7.2 Correction and fusion of image datasets

Images acquired either by thermographic or multispectral devices followed distinct distortion models, as previously explained. RGB and multispectral imagery present fisheye distortion due to their lower focal length, though it is especially visible in multispectral images. Thermographic lenses have a larger focal length, resulting in a pincushion distortion effect. However, distortion correction can be parameterized by five coefficients (radial and tangential distortion coefficients: k_1, k_2, p_1, p_2, k_3) in both cases, along

with the camera matrix, K. Hence, the correction of every dataset is performed regardless of the camera source. An exception to this is given by multispectral images, which define their own distortion model with four distortion coefficients arranged in a 3th-degree equation. Note that distortion coefficients are not frequently read from image metadata; instead, the bundle adjustment phase of SfM is used to estimate these coefficients.

According to the surveying devices, two different pipelines are here established. The first pairs visible images with thermographic data by determining the image's id and the corresponding co-acquired thermal image. Similarly to the described pipeline, multispectral images are linked to the source RGB image. However, the latter image-matching is performed hierarchically, with the Green layer being the master one. Hence, properties of other layers are calculated according to this, including the matrices for fusing images. The result of this section is a composite matrix for every RGB image, either relating it to a thermal or multispectral image. These are not related to more than one data source as they cannot be considered to be triggered synchronously. At this point, point clouds can be fed with additional information since their projection matrix to the RGB image plane is known.

7.3 Projection in 3D point clouds

The following step is to project 3D points into their original data source. As occurs with photogrammetric pipelines, the RGB point cloud is first generated since it represents the most accurate and dense 3D result. This model can be enhanced by including additional spectral information at every point. Otherwise, a low-density version of the RGB point cloud would be generated from another data source. Hence, this work reuses the starting RGB point cloud, whose viewpoints are calibrated during SfM. Each viewpoint c is associated with a projection matrix P_c that allows mapping 3D points into individual RGB image planes. From here, the previously computed inverse homography matrices allow translating RGB coordinates into a different Cartesian coordinate system. Note that the projection of a 3D point for a viewpoint is not always within the image boundaries; indeed, only a few points will be mapped to each image. The main shortcoming is that some points will not be successfully mapped to any image, mainly those located at the boundaries. As a result, augmented point clouds have different dimensions than the baseline due to the smaller focal length of the rest of the devices in comparison with the RGB sensor. Still, the resolution is preserved in terms of points per m^2.

7.4 Occlusion

The main drawback of the previously described projection is that spectral data from images could be projected into 3D points that were not visible from a viewpoint. Hence, this could lead to point clouds with incorrect radiometric data for additional layers. In this chapter, occlusion is handled

using z-buffers to capture the scene depth as grayscale values. Due to the aim of this work, z-buffers also store the index of the nearest point projected at each pixel. In this manner, points that are more distant from the viewpoint than the point stored at the projected pixel are discarded. From a ∞-filled image, points close to the viewpoint are projected to determine the minimum depth at each pixel, thus generating a z-buffer similar to the output of depth sensors. However, z-buffers with low resolution also lead to point clouds of lower density, as those generated by commercial software. To avoid this, the dimensionality of z-buffers is upscaled by a factor s to be determined in order to balance memory usage and the point cloud's density.

This task was solved using three different approaches. First, it was solved using OpenGL's built-in z-buffers, then, these were constructed from scratch using OpenGL's compute shaders, and finally, a similar pipeline to OpenGL's compute shaders was followed using CUDA.

7.4.1 OpenGL's rendering

Constructing z-buffers is straightforwardly solved with OpenGL as it already addresses the occlusion in the rendering pipeline. More specifically, it is the merging phase at the end of the traditional rasterization pipeline that is in charge of determining which primitive is visible at each pixel [120]. However, the desired outcome of the visibility test is a buffer with the indices of points which are visible from a camera pose. The workflow of this approach is shown in Figure 7.2.

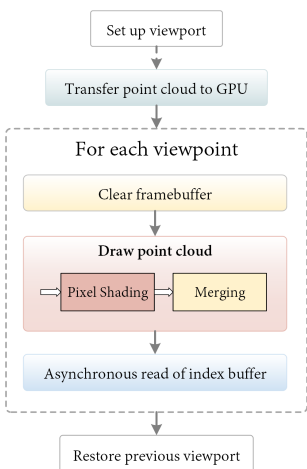


Figure 7.2: Workflow of the rendering-based methodology for a single batch of 3D points.

Points are transferred to the GPU as `vec4` values (i.e., four floating-point numbers) to fit the data alignment rules of subsequent sections. Also, a framebuffer comprising two textures of the same size is required; the first one collects point indices, whereas the second is a conventional z-buffer. Clearing the framebuffer involves emptying the depth buffer and assigning a large value (∞) to the index texture. Then, the whole point cloud is projected over the framebuffer to retrieve the nearest point visible at each pixel. Finally, the first texture is transferred to the CPU via asynchronous reading (`glReadPixels`). In contrast to the limited GPU memory, CPU storage is large enough to store a buffer of size $|V| \cdot x \cdot y$, provided that V is the image dataset, where each image has a resolution of x, y.

However, large point clouds may not fit in a single GPU buffer. An alternative workflow is depicted in Figure 7.3, with the maximum number of points depending on the maximum storage capacity. By default, the point cloud order is unknown and therefore iterations during this process are not disjoint. Consequently, the asynchronous reading involves reading the content of both depth and index buffers, which are then transferred to the GPU for the next batch of points, instead of being reset. Transferring the downloaded z-buffer content to the GPU requires another rendering stage whose core is the built-in attribute `gl_FragDepth` in the vertex shader. Also note that the framebuffer is not reset on each iteration, but only for the first batch of points. Another approach is to compute the results for

each viewpoint in a single iteration, though it implies a huge increase in data transfers from CPU to GPU. The latter requires transferring every point cloud batch once per viewpoint ($|V| \cdot n_b$), instead of n_b, with n_b being the number of point cloud subdivisions.

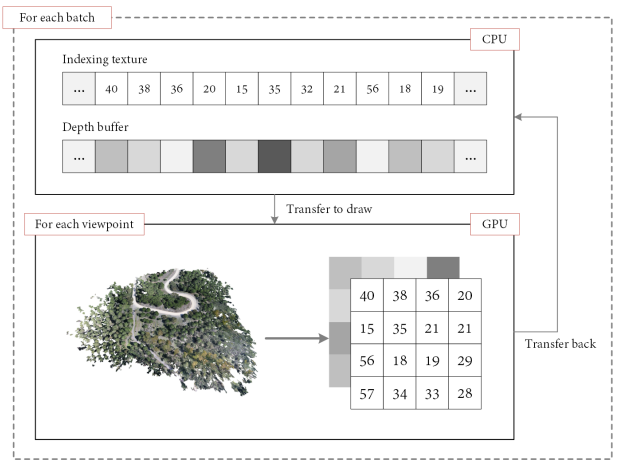


Figure 7.3: Overview of the rendering-based algorithm for multiple point cloud subdivisions.

7.4.2 Compute shaders

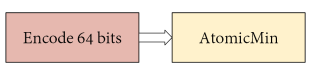
Compute shaders are general-purpose shaders for GPGPU, and therefore, they can be applied to tasks not related to rendering. Unlike OpenGL's rendering pipeline, z-buffers are not provided in this shader stage and thus must be implemented as a low-level mechanism supported by buffers known as Shader Storage Buffer Objects (SSBO). Despite the lack of integrated z-buffers, compute shaders provide a more efficient and flexible solution to read, write and perform trivial operations in buffers. Furthermore, it avoids some unnecessary stages from the rendering pipeline, including polygon rasterization and interpolations.

Regarding memory allocation, z-buffers are here modelled as integers of 64 bits (`uint64_t`) using OpenGL's modern extensions, to be equally split between distance and index. Distance is not a numeric integer, $d \in \mathbb{R}$, though its low-level representation can be interpreted as an integer using `floatBitsToUint`. For a viewpoint, xyz_c, and a 3D point (xyz_p) with index i_p, the Euclidean distance is computed, transformed into an integer and concatenated to i_p (Equation 7.1). With this representation, fewer and greater comparisons work as usual. Thus, these operators can be applied to build a z-buffer by selecting the minimum distance while also carrying the point index. To this end, the atomic min operator (`atomicMin`) is used.

$$z_{k,l} \leftarrow \|xyz_p - xyz_c\| \frown x_p \tag{7.1}$$

The procedure to project the point cloud into z-buffers is following described in Algorithm 3 if it fits in the GPU's VRAM and the maximum size of OpenGL's buffers (see Figure 7.5). Otherwise, Algorithm 4 is followed. Note that the second scenario omits the smaller procedures shown in Algorithm

Figure 7.4: Compute shader steps involved in rendering a point cloud.



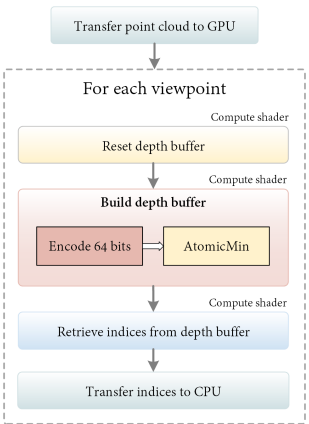


Figure 7.5: Overview of methodology based on compute shaders for a single batch of 3D points.

3. Because data transfers of z-buffers in CPU $\leftrightarrow$ GPU are notably smaller, point batches are iterated in the outer *for* and thus are only transferred once. Instead, each viewpoint's z-buffer is transferred once per point batch. Once all the points have been projected into every z-buffer, the indices indicating which points are visible are read, as proposed in Algorithm 3.

Algorithm 3 Simplest case for projecting points in z-buffers, where all the points fit in the GPU's VRAM.

Input 3D RGB points: P_{GPU}
Input Viewpoints with dimensionality $x \cdot y$: V_{CPU}
Variable $thread_{id}$ in the GPU: id
Output RGB point cloud with augmented thermal or multispectral data
Build $z\text{-buffer}_{GPU} \leftarrow uint64(x \cdot y)$ and $indices_{GPU} \leftarrow uint(x \cdot y)$
if Sort P_{GPU} **then**
    $P_{GPU} \leftarrow sortPoints(P_{GPU})$ ▷ (Algorithm 5)
end if
procedure BUILD z-BUFFERS
    for v in V_{CPU} **do**
        $M \leftarrow v_{projection}$
        procedure RESET z-BUFFER IN GPU
            $z\text{-buffer}_{GPU\,i} \leftarrow UINT64_{MAX}\ \forall i \in [0, x \cdot y[$
        end procedure
        procedure FILL z-BUFFER IN GPU(M, v_p)
            $p_{3D} \leftarrow P_{GPU}[id]$
            $p_{3D} \leftarrow \left[M \cdot \left[p_{3Dx}, p_{3Dy}, p_{3Dz}, 1 \right]^T \right]^T_{xyz}$
            $p_{2D} \leftarrow \left[p_{3Dx}/p_{3Dz}, p_{3Dy}/p_{3Dz} \right]^T$
            if p_{2D} inside image **then**
                $dist \leftarrow floatBitsToUint(length(p_{3D} - v_p))$
                $tag \leftarrow p_{index} | (dist << 32)$
                $atomicMin(z\text{-buffer}_{GPU}[id], tag)$
            end if
        end procedure
        procedure OBTAIN INDICES FROM z-BUFFER
            $index[id] \leftarrow z\text{-buffer}_{GPU}[id] \,\&\, UINT_{MAX}$
        end procedure
        $visible_points \leftarrow indices_{GPU}$
        Obtain point cloud colours in visible_points
    end for
end procedure

7.4.3 CUDA

The CUDA kernel for computing the nearest 3D point for every pixel and viewpoint is analogous to the compute shader workflow. The main advantage of the CUDA implementation is that it enables overlapping the CPU $\leftrightarrow$ GPU transfers with the GPU processing. The use of multiple CUDA streams exploits asynchronous transfer operations and kernel executions to achieve an efficient result. Hence, points from a batch are projected in the CUDA kernel while the next batch is being transferred to GPU. Similarly, results are downloaded from the CPU while a viewpoint is being processed by the current kernel.

Algorithm 4 The point vector is split if it does not fit in the GPU's VRAM during their projection in z-buffers.

Input 3D RGB points: P_{GPU}
Input Viewpoints with dimensionality $x \cdot y$: V_{CPU}
Input Maximum number of points that can be allocated in the GPU: n_{max}
Variable $thread_{id}$ in the GPU: id
Output RGB point cloud with augmented thermal or multispectral data
Build $z\text{-buffer}_{GPU}CPU \leftarrow \text{uint64}(x \cdot y \cdot |V_{CPU}|)$
$z\text{-buffer}_{GPU}CPU_i \leftarrow \text{UINT64}_{MAX}\ \forall i \in [0, x \cdot y \cdot |V_{CPU}|[$
Build $z\text{-buffer}_{GPU} \leftarrow \text{uint64}(x \cdot y)$ and $\text{indices}_{GPU} \leftarrow \text{uint}(x \cdot y)$
$n_{left} \leftarrow |P_{GPU}|$
if Sort P_{GPU} **then**
    $P_{GPU} \leftarrow \text{sortPoints}(P_{GPU})$ ▷ (Algorithm 5)
end if
while $n_{left} > 0$ **do**
    $n_{current} \leftarrow \min(n_{left}, n_{max})$
    Transfer point batch to GPU
    for v in V_{CPU} **do**
        Transfer to GPU part of $z\text{-buffer}_{GPU}CPU$ from v
        Complete $z\text{-buffer}_{GPU}$ with current point batch
        Read $z\text{-buffer}_{GPU}$ into the pointer of $z\text{-buffer}_{GPU}CPU$
    end for
    Update n_{left}
end while
for v in V_{CPU} **do**
    Transfer to GPU part of $z\text{-buffer}_{GPU}CPU$ from v
    Split indices from $z\text{-buffer}_{GPU}$ into indices_{GPU}
    visible_points $\leftarrow \text{indices}_{GPU}$
    Obtain point cloud colours in visible_points
end for

7.5 Point cloud sorting

Point clouds generated by software present an unknown order regardless of their coordinates, *xyz*. However, the outcome of previous projections is highly influenced by the sampled 3D volume; the points from a relatively small volume can be visible from a camera viewpoint while the others are discarded. Therefore, point clouds whose primitives have an unknown order also require unordered updates in memory buffers. On the other hand, organized updates are known to be more efficient due to faster memory writing. Although there is no such ordering in 3D space, there exist methods for the hierarchical clustering of geometry, such as the Z-order curve depicted in Figure 7.6. The outcome is a 1D buffer where points which have close indices also happen to be in close spatial positions.

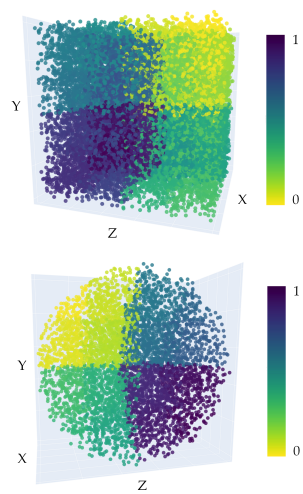


Figure 7.6: Colouring of two randomized point clouds in [0, 1] according to Morton encoding with 30 bits.

For large point clouds, changing their order presents a significant delay even for the most efficient sorting algorithms. Therefore, GPGPU programming allows sorting them in parallel to reduce the response time during the projection procedure. In this work, Radix sort is used and adapted to GPGPU by splitting it into two different steps: 1) up-sweep (reduce) and 2) down-sweep [147]. Following this approach, the buffer is sorted with a complexity of $\mathcal{O}(mn)$, with m being the number of bits and n being the number of points. Prior to sorting, *xyz* coordinates are converted to values of 30 bits (m) that encode individual coordinates (10 bits for each one).

These have been previously revised as Morton codes, though they can have a variable bit length.

Despite the enhancement derived from sorting, GPU-based algorithms are handled on the GPU through work groups. These are small groups of threads that are allowed to share data. Thus, the main drawback of globally sorting the point cloud buffer is that the workload is not uniformly shared among work groups. Instead, the ordered buffer can be shuffled according to the size of the workgroups. The algorithmic flow for the described sort configuration is shown in Algorithm 5. First, points are sorted with Radix Sort following the Morton encoding. Otherwise, a random order is used. Then, the point buffer can be reordered according to groups of size $n_{shuffle}$. The second stage is implemented as a multi-core CPU algorithm rather than in the GPU to avoid duplicating large point buffers with limited VRAM.

7.6 Alignment of heterogeneous data sources

The image-matching procedure was performed over different data sources from the same devices, e.g., visible and thermal data. With this approach, images are known to be acquired with a similar timestamp, and thus ECC converges in a reasonable time. However, RGB point clouds can be reused for different data sources whether they are represented in the same local coordinate system. Despite georeferencing providing accurate positioning, point clouds from distinct data sources may not exactly overlap. Hence, point clouds can be further aligned by finding the rigid transformation matrix with a precision below a threshold (Figure 7.7). In this chapter, it is implemented using the ICP algorithm.

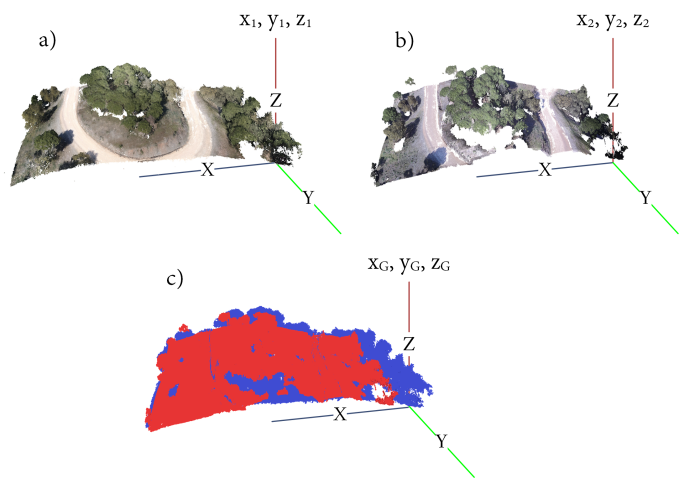


Figure 7.7: ICP registration process: a) RGB point cloud from the first dual device, b) RGB point cloud from the second dual device, and c) both point clouds aligned by minimizing their distance, with *G* referring to a global system.

Algorithm 5 Point cloud sorting.

Input Point cloud: P_{GPU}
Input Maximum number of points that can be allocated in the GPU: n_{max}
Variable $thread_{id}$ in the GPU: id
Output Sorted point cloud
$n_{left} \leftarrow |P_{GPU}|$
indices $\leftarrow$ uint $\left[|P_{GPU}|\right]$
while $n_{left} > 0$ **do**
    $n_{current} \leftarrow \min(n_{left}, n_{max})$
    Transfer the following $n_{current}$ batch to GPU
    procedure SORT IN GPU
        $codes_{GPU} \leftarrow mortonGPU(P_{GPU})$
        indices $\leftarrow$ RadixSort(P_{GPU}, $codes_{GPU}$)
        procedure RADIXSORT(P_{GPU}, $CODES_{GPU}$)
            for bit $\leftarrow 0 \rightarrow 30$ **do**
                Apply bit mask 1 << bit toP_{GPU}
                Up-sweep phase of prefix scan
                Reset last position to zero
                Down-sweep phase of prefix scan
                Reorder $P_{GPU}$with new positions
            end for
        end procedure
    end procedure
    procedure SORT IN MULTI-CORE CPU
        if Shuffle **then**
            $n_{groups} \leftarrow \lceil n_{current} / n_{shuffle} \rceil$
            $rdn_{group} \leftarrow$ RandomVector(n_{groups}, 0, 1)
            $iota_{group} \leftarrow \left[0, 1, \ldots, n_{groups} - 1\right]$
            id_{old}, $id_{new} \leftarrow$ Sort(rdn_{group}, $iota_{group}$)
            for group in (id_{old}, id_{new}) **do**
                Move point batch from id_{old} to id_{new}
            end for
        else
            Reorder points with indices
        end if
    end procedure
    Update P_{GPU} and n_{left} with $n_{current}$
end while

7.7 Results and discussion

In this section, the proposed methodology is compared with other notable software solutions for building multispectral and thermal point clouds, such as Pix4Dmapper or Agisoft Metashape. The carried tests are focused on obtaining the response time and point cloud size, although some reconstructions from these solutions also present geometrical errors. The evaluation was performed on a PC with AMD Ryzen Threadripper 3970X 3.6 GHz, 256 GB RAM, two NVIDIA RTX A6000 GPUs and Windows 10 OS. The proposed methodology was implemented in C++ using OpenGL, and besides GPU computing, CPU-based methods were accelerated using the Open Multi-Processing (OpenMP) framework.

The baseline RGB point cloud consists of 115M (multispectral) and 97M (thermal) points for the first dataset, whereas the second dataset yields point clouds with a dimensionality of 126M and 96M points. To further stress this evaluation and achieve higher point density, a single depth buffer resolution has been used, $d \longleftarrow 10$. Regarding point cloud order, points are initially shuffled to show the benefits of spatial sorting. Apart from this, two different setups are here shown: 1) sorting with Morton codes and 2) sorting and shuffling in small groups. The experimental results are obtained by repeating every test four times and averaging the results.

OpenGL and CUDA comparison

This chapter comprises the last work of this dissertation, with compute shaders being the preferred solution. Other presented approaches use OpenGL's rendering pipeline as well as the CUDA framework to perform this very same operation. However, the latter comparison was performed in a work published in the **Proceedings of the 2021 Spanish Conference in Computer Graphics**. In this work, tests were performed on a PC with Intel Core i9-9900 3.1 GHz, 48 GB RAM, RTX 2080 Ti GPU with 11 GB RAM (Turing architecture) and Windows 10 OS, using C++, CUDA and OpenGL. The dataset was the first one explained (forestry area), although the input cloud had 271 million points which were observed from 1352 viewpoints. A simplified version of this comprises 66 million points and 180 viewpoints.



Figure 7.8: Overview of the surveyed region and areas. a) The region of Jaén, b) a forest and c) an olive grove.

Regarding the configuration of commercial solutions, the steps of image alignment and reconstruction of a dense point cloud (i.e., densification) were performed with the highest quality. Since both of them are known to be able to use CUDA-enabled GPUs, they were enabled to accelerate their pipeline on the GPU. Furthermore, the latter is known to be accelerated with the multi-GPU framework CUDA and therefore, was expected to perform better since two GPUs were available.

7.7.1 Study area

The proposed method has been evaluated over two different environments located in the region of Jaén, Spain. To show its effectiveness in multiple fields, data depicting 1) a forestry area and 2) an olive grove was processed. Hence, the results will prove that the fusion of information works well in scenarios with more significant features (2) and not recognizable features (1). Figure 7.8 shows the location of both areas. The flight altitude was 454 m and 595 m above sea level, respectively. The forestry scene is composed of repetitive patterns from vegetation and human-made structures. The first area covers nearly 1.97 ha, whereas the second includes 2.44 ha.

7.7.2 Image processing

The image alignment stage is configured according to the ECC parameters: 1) the number of iterations to converge and 2) the aimed precision. The first is set to 400 iterations to sufficiently cover the number of required iterations. The precision is set to the default value, $\epsilon \leftarrow 1^{-3}$. Besides these parameters, the dimensionality of the compared images is also relevant since the lesser number of details also implies a faster convergence. However, images were not downscaled in this study to ensure that a fine-grained alignment is achieved. Finally, the hierarchical alignment of multispectral images is also studied by means of confusion matrices that show the correlation of pairs of bands. Accordingly, Figure 7.9 justifies the hierarchy shown in Figure 5.4. Note that NIR and REG layers are reciprocally the most similar. However, REG images are more similar to the root, Green, and thus NIR layers were aligned to REG images.

Regarding the sought precision, images were aligned using $\epsilon \leftarrow 10^{-5}$ for all the data sources, according to the similarity observed in Figure 7.10. The similarity was normalized to improve the readability of the chart, though the average intensity differences through the experimental results were observed to be in the order of 1^{-3}. Lower values (higher exponent) harden the convergence of the ECC alignment while also increasing the latency in the order of ms. Hence, an intermediate value that balances both quality and response time was finally used. On the other hand, the number of iterations was established to guarantee that convergence is achieved before reaching such a number. Therefore, a large value such as $it \leftarrow 400$ was enough for our case study.

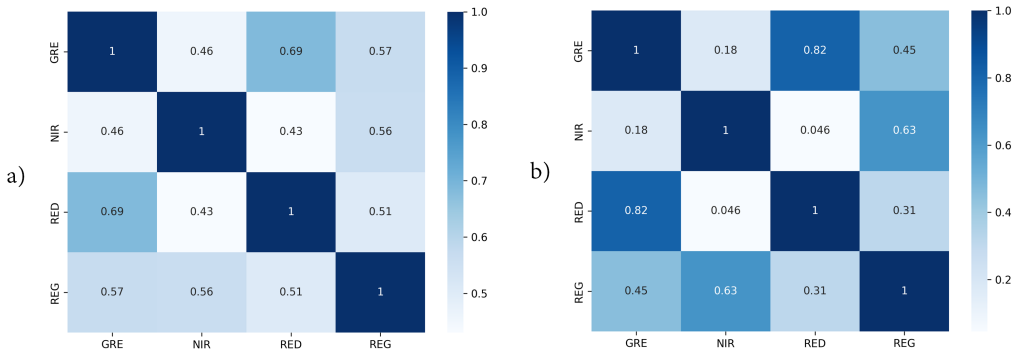


Figure 7.9: Confusion matrix to depict the similarity of multispectral images in the first and second dataset (a) and b), respectively).

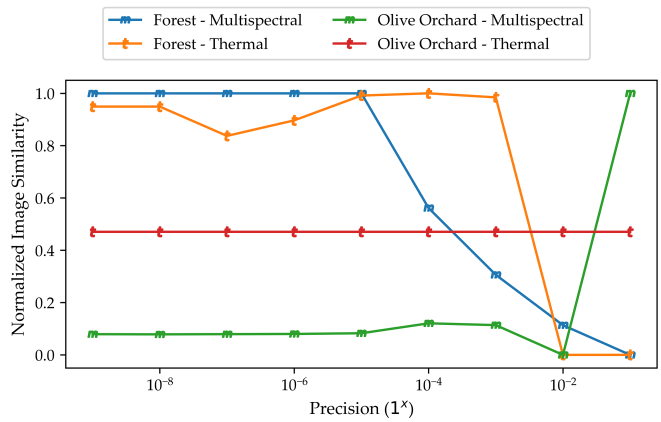


Figure 7.10: Image similarity measured after performing the ECC alignment with a precision equal to 1^x. Lower values seek the most accurate alignment. The similarity is normalized in [0, 1] for improving the visualization.

7.7.3 Response time

First, the response time of commercial solutions was compared against three configurations of our method. The response time is decomposed into three stages: 1) reading the visible point cloud, only for our method, 2) preprocessing stage, including image registration, and 3) reconstruction of the dense point cloud. For commercial software, the second step comprises the finding and matching of image features. The results are shown in Figure 7.11, and further insight is provided in Table 7.1 and 7.2. The response time of our methodology was significantly lower than in the compared commercial solutions, despite the fact that our method reconstructed larger point clouds. This difference is further exploited for GPU-based solutions and larger datasets since part of the measured latency comes from the allocation of GPU buffers that are constructed once and reused for several multispectral layers. Thus, the baseline latency from allocation does not increase with a larger number of images. Accordingly, the sorted GPU solution improves Agisoft Metashape by 77.3% on average to build thermographic point clouds. The improvement of multispectral reconstruction is 77.26% with respect to Pix4Dmapper.

The tests were performed with a z-buffer with $d \leftarrow 10$, which was shown

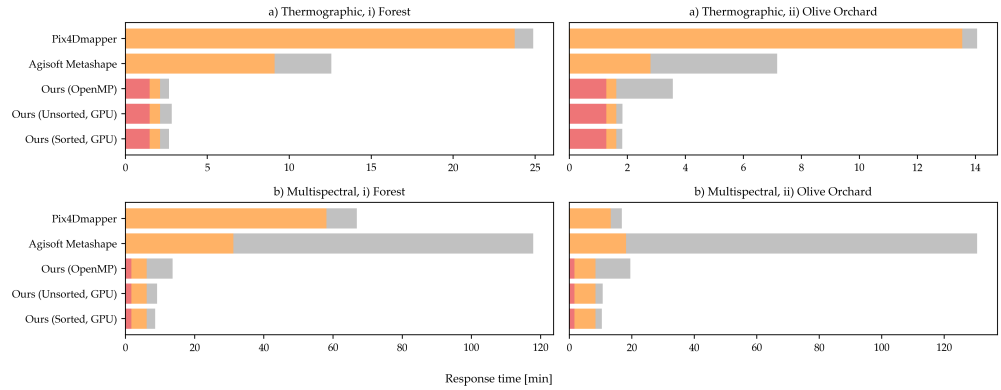


Figure 7.11: Accumulated response time for every data source (1. thermographic and 2. multispectral), study area (a. forestry and b. olive orchard) and solution, including commercial software and configurations of our method.

to balance latency and point cloud dimensionality. Larger values of d involve higher data transfers, while also risking to include background points with foreground data. Otherwise, lower values of d reduce the dimensionality of point clouds at the expense of minimizing the latency from data transfers. Based on this, $d \leftarrow 1$ would provide the baseline improvement of the ECC method with respect to the recognition of feature points in images to build a sparse point cloud. Both procedures iterate over the whole dataset, whereas the former only evaluates a set of paired images instead of surrounding images. This improvement is considerably lowered for multispectral images since ECC is performed up to five times.

Comparison of OpenGL and CUDA solutions

Table 7.3 shows the response time measured from data transfers and the occlusion-aware projection. The single batch configuration loads the whole point cloud in a single buffer if the maximum size of an SSBO is below the maximum allocatable GPU memory. Otherwise, it works as the **Multiple batches** approach does. On the other hand, the CUDA framework was able to handle the largest point cloud (271M points) in a single buffer. The reported results are the lowest timed response time from five different tests. The measured latency in Table 7.3 and Figure 7.12 show that compute shaders significantly improve the response time obtained using the traditional rendering pipeline and CUDA. Also, the rendering pipeline offered slightly better results than CUDA for the single-batch workflow. Regarding the use of multiple batches, both OpenGL-based approaches worsened their performance in comparison with previous tests. In contrast, the CUDA version obtained similar results to the previously measured latency.

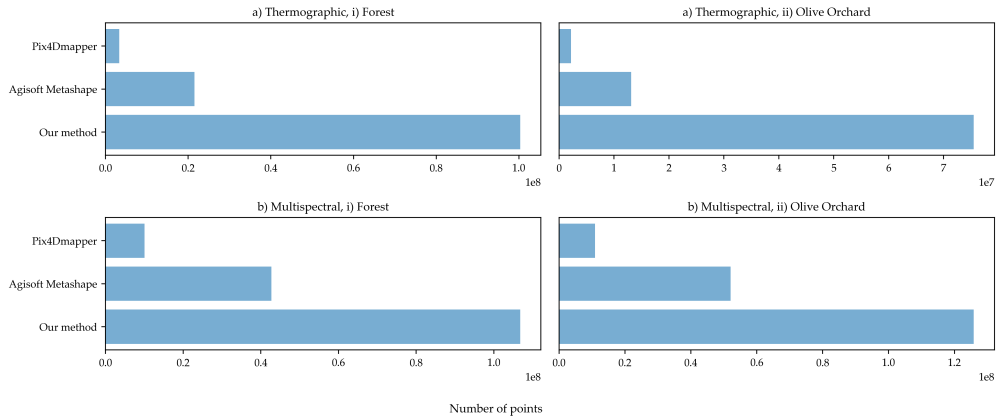


Figure 7.11a: Continuation of Figure 7.11 on the dimensionality of the reconstructed point clouds.

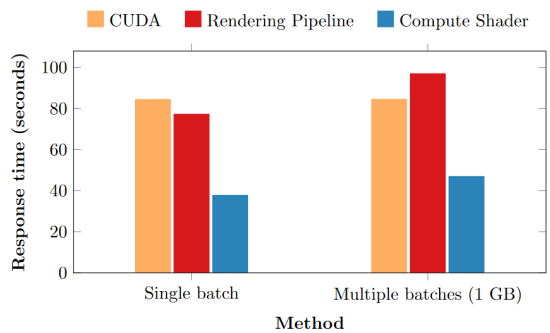


Figure 7.12: Latency in seconds for OpenGL and CUDA approaches, using a point cloud with 271M points and 1352 viewpoints).

7.7.4 Point cloud size and normalized response time

The proposed method is included as part of a pipeline where dense point clouds of RGB and alternative data sources are jointly reconstructed. In this scenario, the first point cloud is considerably larger than those obtained with imagery of lower resolution and less recognizable features. From this baseline, the number of points of the RGB point cloud is reduced according to their visibility from the whole set of camera viewpoints and the depth buffer's resolution. Still, most of them are preserved and linked to data interpolated from thermal and multispectral imagery in our case study.

Point clouds reconstructed from multispectral images are dense due to the provided variety of reflectance and the higher number of recognized features. However, the higher sparsity of thermal point clouds is observed in Figure 7.11. For the thermographic datasets, our method increases the point cloud size by 366.3% (forest) and 475.52% (olive orchard) with respect to Agisoft Metashape. In this regard, this commercial solution reconstructs the densest point clouds from the compared software, both for multispectral and thermal imagery. Despite this, it is significantly slower for larger datasets such as the multispectral. For the latter data source, the size increases by 149.89% (forest) and 141.93% (olive orchard) also in comparison with Agisoft Metashape.

Table 7.1: Numeric results obtained from commercial solutions and different configurations of our approach in two different thermal datasets. The global response time is split into reading point cloud (Stage 1), pre-processing (Stage 2) and densification (Stage 3), i.e., generation of a dense point cloud.

Configuration	Stage 1	Stage 2	Stage 3	Latency	Latency/point	#Points	Matching
Thermal Dataset 1 (Forestry)							
Pix4Dmapper	-	23.76 min	1.13 min	24.89 min	449.08 µs/point	3,325,454	98%
Agisoft Metashape	-	**9.10** min	3.47 min	12.57 min	35.05 µs/point	21,514,286	62.95%
OpenMP, No Sort	1.48 min	**0.64** min	9.77 min	11.89 min	6.90 µs/point	**100,322,449**	**99.287%**
GPU, No Sort			0.71 min	2.83 min	1.69 µs/point		
GPU, Global Sort			**0.546** min	**2.666** min	**1.594 µs/point**		
Thermal Dataset 2 (Olive orchard)							
Pix4Dmapper	-	13.55 min	0.51 min	14.06 min	388.92 µs/point	2,169,058	90%
Agisoft Metashape	-	**2.80** min	4.37 min	7.17 min	32.79 µs/point	13,117,583	**99.39%**
OpenMP, No Sort	1.28 min	**0.34** min	1.95 min	3.57 min	2.83 µs/point	**75,494,967**	89.75%
GPU, No Sort			0.217 min	1.837 min	1.459 µs/point		
GPU, Global Sort			**0.208** min	**1.828** min	**1.452 µs/point**		

Previous results based on global latency present a wider gap if they are normalized according to the point cloud's dimensionality. The normalized latency of our best method improves the results of the best commercial solution by 95.51% and 96.9% on average for thermal and multispectral images, respectively.

7.7.5 Overall comparison

The results of the four datasets and the five compared methods are summarized in Figure 7.13. The global latency and point cloud size features are normalized in $[0, 1]$. The first feature must be minimized, whereas the second one ought to be maximized to provide a better reconstruction of the target scenario. According to this figure, the three proposed methods present an outstanding performance in terms of response time and point cloud dimensionality. These are followed by Agisoft Metashape, which is considerably slower, albeit generating larger results than the last alternative, Pix4Dmapper.

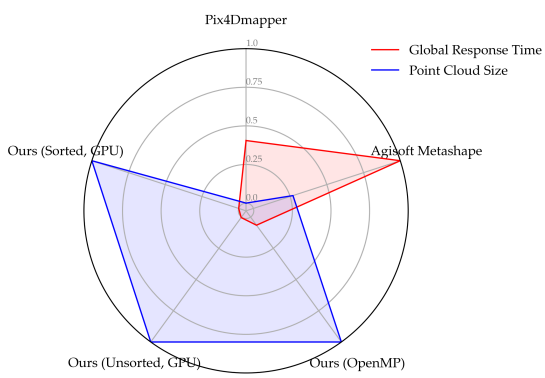


Figure 7.13: Average response time and point cloud size obtained by commercial software and three versions of the proposed method.

Table 7.2: Continuation of Table 7.1 for multispectral datasets.

Configuration	Stage 1	Stage 2	Stage 3	Latency	Latency/point	#Points	Matching
Multispectral Dataset 1 (Forestry)							
Pix4Dmapper	-	58.11 min	**8.76** min	66.87 min	398.35 s/point	10,071,939	97%
Agisoft Metashape	-	**31.22** min	86.64 min	117.86 min	165.49 µs/point	42,731,004	67.74%
OpenMP, No Sort	1.71 min	**4.45** min	7.50 min	13.66 min	7.67 µs/point	**106,780,612**	**100%**
GPU, No Sort			2.99 min	9.15 min	5.14 µs/point		
GPU, Global Sort			**2.42** min	**8.58** min	**4.82 µs/point**		
Multispectral Dataset 2 (Olive orchard)							
Pix4Dmapper	-	**13.3** min	**3.5** min	16.8 min	92.56 µs/point	10,889,523	**100%**
Agisoft Metashape	-	18.15 min	112.36 min	130.51 min	150,52 µs/point	52,021,396	**100%**
OpenMP, No Sort	1.67 min	**6.69** min	11.19 min	19.55 min	9.32 µs/point	**125,857,793**	99.934 %
GPU, No Sort			2.31 min	10.67 min	5.08 µs/point		
GPU, Global Sort			**2.08** min	**10.44** min	**4.97 µs/point**		

Table 7.3: Response time of methodologies using OpenGL and CUDA, including data transfers between CPU and GPU.

Proposed methods		Environment			
		180 cameras		1352 cameras	
		66M points	271M points	66M points	271M points
Single batch	CUDA	2.97s	10.30s	23.58s	84.45s
	Rendering pipeline	2.52s	10.38s	20.70s	77.28s
	Compute shaders	**1.43s**	**6.21s**	**8.90s**	**37.75s**
Multiple batches (1 GB)	CUDA	3.22s	11.19s	25.42s	84.50s
	Rendering pipeline	3.71s	13.35s	29.15s	96.99s
	Compute shaders	**2.57s**	**7.61s**	**18.16s**	**46.90s**

7.7.6 Data shuffling in GPU

As stated by previous work [122, 394], specific GPU patterns are used by manufacturers to distribute the workload among thread groups. This knowledge can help to further reduce the latency of GPU-based solutions, in comparison with globally sorted point clouds. Thus, the objective is to shuffle points in batches to balance the workload of every workgroup. As the outcome of a batch is expected to depend on the camera viewpoint, the data shuffling ought to help to reduce latency by distributing visible and non-visible points. To this end, a comparison of data shuffled or not after being sorted was conducted, with groups of size $g \leftarrow 32$. The results are depicted in Figure 7.14 by splitting the latency into the Sort & Shuffle and Reconstruction stages. Numeric results are also annotated in Table 7.4.

To provide a better insight into the data shuffling, the second image shows only the response time from the dense reconstruction. The shuffled ordering also involves global sorting prior to organizing the point cloud into batches. Thus, the Sort & Shuffle procedure is more time-consuming than solely sorting. Accordingly, the summed response time of the Shuffled

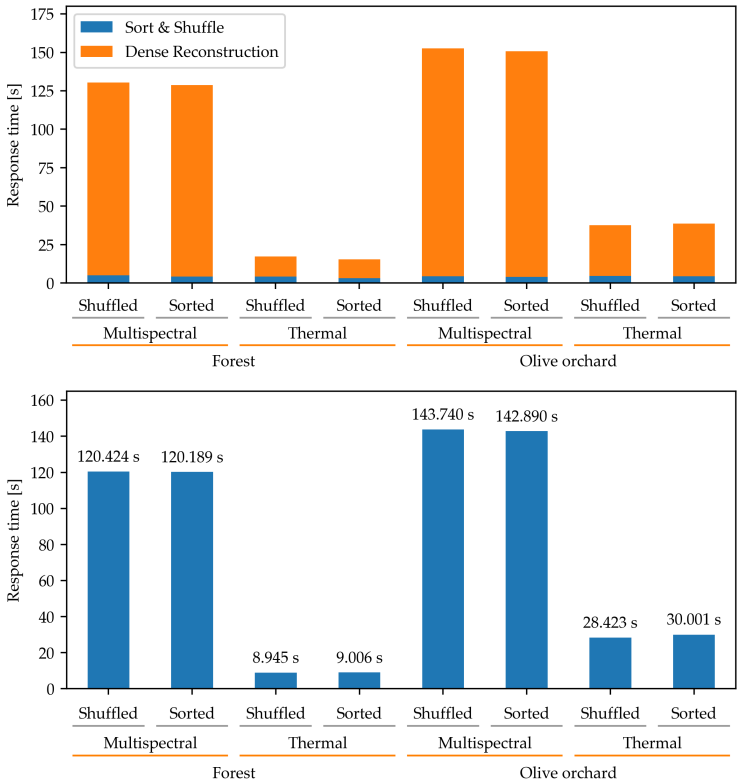


Figure 7.14: Comparison of performance of the dense reconstruction stage using sorted & shuffled data and globally sorted data. The first image compares the overall latency of the dense reconstruction, whereas the second image only compares the projection procedure.

version is higher for every configuration whether both stages are included. Otherwise, the results offer slight changes if only the reconstruction stage is considered. It is observed that data shuffling effectively helps to reduce latency in smaller point clouds (thermographic), whereas larger point clouds worsen the latency obtained by the global sort. Unlike previous studies, the proposed case study is very unbalanced for individual viewpoints. From hundreds of millions of points, only a few million are visible from a camera viewpoint. Thus, the workload balancing was not effective. Even if so, the Shuffle stage increases the latency for a one-time process, thus making it not worth it. Instead, repetitive tasks such as rendering are more likely to benefit from the use of this technique.

7.7.7 Visual results

Table 7.6 shows the rendering of multispectral and thermographic point clouds obtained by the three compared solutions. In this regard, both commercial solutions are observed to reconstruct incomplete and erroneous geometry for thermographic data. Instead, our method projects these data over an RGB point cloud that is accurately reconstructed. Multispectral

Table 7.4: Latency of the first two stages if 1) it is the first load or 2) binary data has already been stored as a result of a previous load.

	Global sort		Shuffled sort ($g \leftarrow 32$)	
Dataset	Point cloud sort	Dense reconstruction	Point cloud sort	Dense reconstruction
a) Thermal, i) Forest	**0.071** min	0.500 min	0.076 min	**0.473** min
a) Thermal, ii) Olive orchard	**0.052** min	0,150 min	0.052 min	**0.149** min
b) Multispectral, i) Forest	**0.065** min	**2.381** min	0.074 min	2.395 min
b) Multispectral, ii) Olive orchard	**0.071** min	**2.003** min	0.082 min	2.007 min

imagery presents higher dimensionality and more recognized features, thus leading to better reconstructions even for commercial software, regardless of the small point density. Hence, the reconstructed multispectral point clouds are similar among all the compared methods. The main drawback of multispectral point clouds reconstructed with photogrammetry is the existence of gaps, which are much less frequent in RGB point clouds despite vegetation being hardly reconstructed.

7.7.8 Binary data

Previous results are measured by 1) loading a Polygon File Format (PLY) point cloud, 2) reading TIFF and JPEG images and 3) reconstructing the dense point cloud from previous data. However, these are known to be time-consuming stages that can be treated differently after the first load. First, point clouds are stored in binary file format. Similarly, the image alignment results can be stored in binary files. Once aligned, RGB imagery is not loaded again. Finally, the reconstructed point cloud is stored as an additional colour to be attached to previously stored binary point clouds. Figure 7.15 compares the latency of binary and non-binary reading of the first two stages. The latency from the dense reconstruction is equal to the response time of the first stage since they load point clouds of the same size. The numeric data is shown in Table 7.5. Thus, the proposed procedure is shown to be very time-consuming, yet more efficient than current solutions, though it can be sped up after computing the results for the first time.

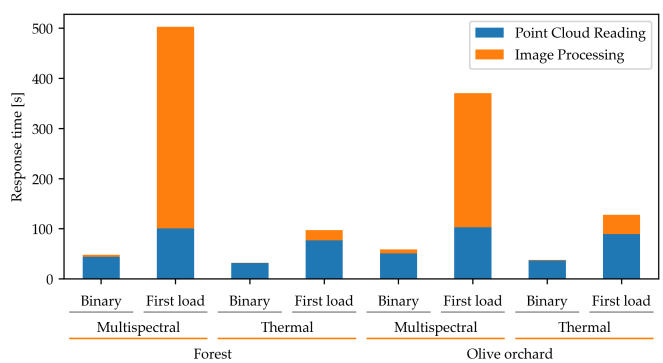


Figure 7.15: Comparison of latency measured from the first two stages of our method in the first and subsequent launches (loading calculated data from binary files).

Table 7.5: Response time of the first two stages if 1) it is the first load or 2) binary data has already been stored as a result of a previous load. Stage 1 refers to point cloud reading, whereas Stage 2 involves the image processing prior to the dense reconstruction.

	Binary		First load	
Dataset	**Stage 1**	**Stage 2**	**Stage 1**	**Stage 2**
a) Thermal, i) Forest	**0.61** min	**0.01** min	1.48 min	0.64 min
a) Thermal, ii) Olive orchard	**0.52** min	**0.01** min	1.28 min	0.34 min
b) Multispectral, i) Forest	**0.85** min	**0.12** min	1.71 min	4.45 min
b) Multispectral, ii) Olive orchard	**0.73** min	**0.06** min	1.67 min	6.69 min

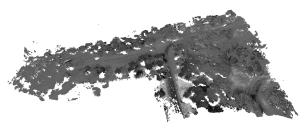
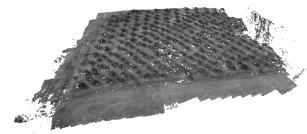
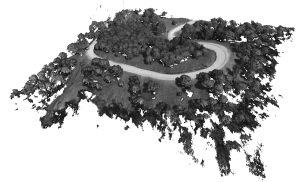
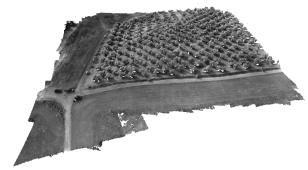
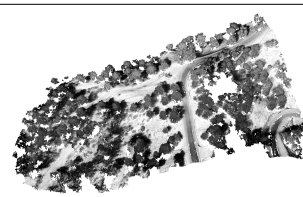
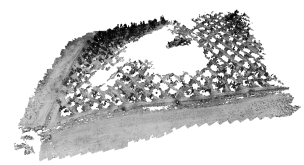
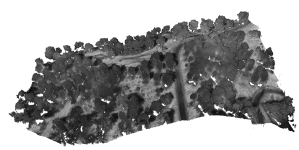
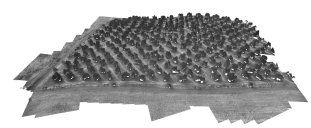
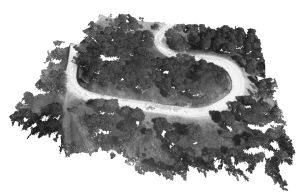
7.8 Conclusions and future work

Reconstructing multiple 3D point clouds is a common task in surveying work, and it often involves not only visible imagery but also other sensors that provide additional information about the studied field. Therefore, reconstructing 3D point clouds from multiple data sources, such as thermal and multispectral, can be accomplished by leveraging the previous RGB reconstructions. While commercial software can perform these tasks, several drawbacks were identified, including high response time, low point density, and incorrect estimations without additional knowledge, such as ground control points (GCPs). The latter issue arises because thermal and multispectral imagery have lower dimensionality and more defects than visible imagery, making it more challenging to recognize image features.

An accelerated projection-based methodology was proposed for integrating reconstructed thermal and multispectral 3D point clouds. The process began with the reconstruction of RGB point clouds using traditional software. Then, the subsequent reconstructions were performed with significantly lower latency (approximately -77%), resulting in point clouds that were more than 140% larger than the initial ones. When normalized according to point cloud size, our method improved latency results by over 95%. Despite both commercial software and our method using GPU-accelerated procedures, ours outperformed two widely-used solutions in terms of response time and size.

Although our proposed approach demonstrated improved performance in terms of latency and point cloud size, the parallelism was mainly addressed with GLSL, which provides limited flexibility and control of the execution pipeline, such as overlapping data transfers and thread logic. Additionally, it runs on a single GPU, thus restricting scalability. Therefore, a future direction is to evaluate our GPGPU solution over the CUDA framework, which offers more control over the execution pipeline and the possibility to use multiple GPUs for higher scalability. In addition, our multi-layer point cloud could be further extended by integrating other data sources, such as hyperspectral datasets. Finally, instead of performing the projection over RGB point clouds, LiDAR point clouds could be used to replace the first with a more geometrically accurate product.

Table 7.6: Graphic results of the dense reconstruction computed by 1) Agisoft Metashape, 2) Pix4Dmapper and 3) our method. Multispectral point clouds are rendered with the Green layer.

Configuration	Data source	Dataset 1	Dataset 2
Agisoft Metashape	Thermal		
	Multispectral		
Pix4Dmapper	Thermal		
	Multispectral		
Our method	Thermal		
	Multispectral		

Generation of hyperspectral point clouds

8

About this chapter

This chapter describes the efficient generation of hyperspectral point clouds as the last step in the enrichment of large RGB point clouds, using images collected by a push broom hyperspectral sensor. To this end, a GPGPU workflow for the generation, compression and rendering of dense hyperspectral point clouds is proposed. It is capable of integrating hyperspectral images into 3D models acquired by LiDAR or estimated with photogrammetry. Figure 8.1 shows the main steps of this approach.

Firstly, the input RGB point cloud is obtained with photogrammetry. Then, hyperspectral images are geometrically corrected and aligned among them with an RGB orthomosaic helping to find common features. Therefore, hyperspectral data can be subsequently projected into the 3D point cloud using the GPU to retrieve which 3D point is visible for each hyperspectral pixel. Furthermore, the rendering of large hyperspectral datasets has not been previously addressed and thus an accelerated rendering approach is proposed to tackle low performance in the visualization. Similarly, the data is compressed by partly following the work of Graciano et al. [119] to obtain a low memory footprint while also aiming at providing a compressed representation suitable for rendering. As a result, the obtained point clouds help in studying the geometrical and hyperspectral features of natural and man-made materials in the real world.

8.1 On the reconstruction of hyperspectral point clouds 151
8.2 Hyperspectral alignment and rectification . 152
8.3 Generation of hyperspectral orthomosaics . 153
8.4 Projection of hyperspectral orthomosaics 155
8.5 Hyperspectral data compression 155
8.5.1 Point cloud rendering . . 157
8.6 Results and discussion 158
8.6.1 Study area and data acquisition 158
8.6.2 Performance analysis . . . 158
8.7 Conclusions and future work 161

8.1 On the reconstruction of hyperspectral point clouds

Sensors which are typically mounted on UASs bring the opportunity to collect hyperspectral surface samples from multiple viewpoints. Thus, light and material interactions of real objects can be captured with a notable spatial resolution. However, the generation of 3D hyperspectral point clouds still poses challenges due to the high geometric distortions in hyperspectral swaths. Most of the current solutions use integrated LiDAR and hyperspectral detectors, though these present several limitations as a result of errors derived from inertial measurements and the spatial resolution of LiDAR.

The realistic modelling of material appearance has been widely studied in Computer Graphics. The exploration of hyperspectral imaging opens the possibility to study the material's response beyond the human visible range. Recent work is presented for the acquisition of hyperspectral data through a goniophotometer and modelling of BRDFs from the collected data. These advances lead us toward the generation of real-world material databases, which are highly used for rendering photorealistic scenes, virtual

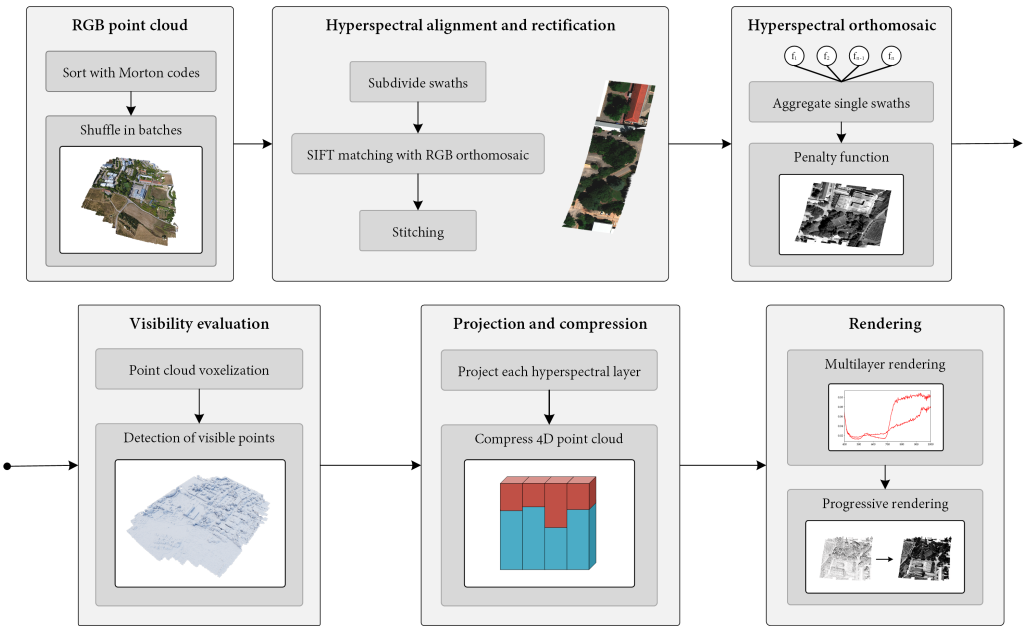


Figure 8.1: The overall approach proposed for the generation of 3D hyperspectral point clouds. The starting point cloud is first sorted and shuffled to enhance its rendering. Hyperspectral swaths are then corrected and fused to build an orthomosaic. This result is projected into the RGB point cloud, previously voxelized to obtain the visible points. The hypercube is then compressed according to a stack-based representation.

interior design in architecture, the game industry, etc. Also, the use of hyperspectral sensors mounted in UASs brings up new opportunities to explore high-resolution spectral analysis. It enables observing light-material interactions from multiple viewpoints in a large number of bands that help to characterize a wide range of real-world surfaces. Besides geometric features, studying the hyperspectral data distribution from reconstructed surfaces helps us to approach non-trivial tasks such as the modelling of specularities, interreflections, transparencies, or subsurface scattering under variable and general observation conditions.

In the last few years, novel solutions have been proposed relying on the mechanical integration of LiDAR sensors and hyperspectral cameras for the generation of 3D spectral data. Despite the progress made in the fusion of multiple UAS-based data sources, to our knowledge, there does not exist a solution capable of automatically enriching 3D point clouds with hyperspectral information.

8.2 Hyperspectral alignment and rectification

Pushbroom hyperspectral sensors provide high spectral resolution data, but the acquisition mechanism leads to the collection of multiple hyperspectral swaths that present high geometric distortions, which is one of the main challenges in hyperspectral processing. Thus, the first required step is to correct these distortions, which is a time-consuming and manual task

that can be solved with commercial software. Otherwise, Jurado et al. [69] proposed an automatized pipeline based on the iterative matching of hyperspectral swaths with an RGB orthomosaic. This algorithm was replicated in this chapter to correct the geometrical distortions. A brief description of this approach is following presented to make this a self-contained work.

The steps for the generation of hyperspectral orthomosaics by aligning hyperspectral swaths are 1) image subdivision, 2) feature detection, 3) matching and calculation of a homography matrix, 4) image transformation, and 5) validation. Firstly, each hyperspectral swath is subdivided into multiple fragments since distortions vary through the Y-axis of the images. Second, the feature detection phase is performed to collect keypoints visible in multiple hyperspectral fragments and the RGB orthophoto using the ORB feature descriptor. The likelihood of the found matches is evaluated with the Hamming distance [395]. Then, the homography matrix is estimated in order to transform every pixel from the hyperspectral fragments, thus enabling a better alignment with the reference RGB orthomosaic. A homography matrix is considered valid if the alignment error is below five pixels. To this end, the accuracy of the alignment is measured using a validation mask for each image, where control points (CP) are represented with different colours. Each one was located in meaningful places within the study area, such as corners and objects that could be easily recognized in hyperspectral and RGB imagery. Then, the method is capable of registering the visible markers within each fragment. Accordingly, the estimated homography was applied and the Euclidean distance was calculated for every projected position of the CPs and its corresponding pixel in the RGB orthomosaic. If the error was above the proposed threshold, the height of the fragment was reduced until finding an appropriate estimation or reaching the maximum number of possible subdivisions.

8.3 Generation of hyperspectral orthomosaics

Once hyperspectral swaths are corrected and aligned with RGB imagery, both datasets overlap in the UTM coordinate system. Instead of using individual swaths in later stages, these can be merged into a single hyperspectral orthomosaic that can be efficiently sampled from a point cloud. Firstly, the offset and size of each swath are represented in UTM, and thus the size of the resulting hyperspectral orthomosaic is calculated by combining the 2D AABB of every swath. Then, each hyperspectral pixel can be mapped to a pixel of the resulting orthomosaic.

Radiance observations may vary among the hyperspectral swaths and the values projected in a single pixel may differ one from another. These radiometric differences have been previously revised while generating thermographic point clouds and are derived from acquisition conditions and errors induced by sensors (e.g., the striping phenomenon) [22], among others. However, this chapter is not aimed at solving systematic errors which can be partially removed as a post-processing task. Instead, penalty functions were used as previously explained to obtain the most accurate

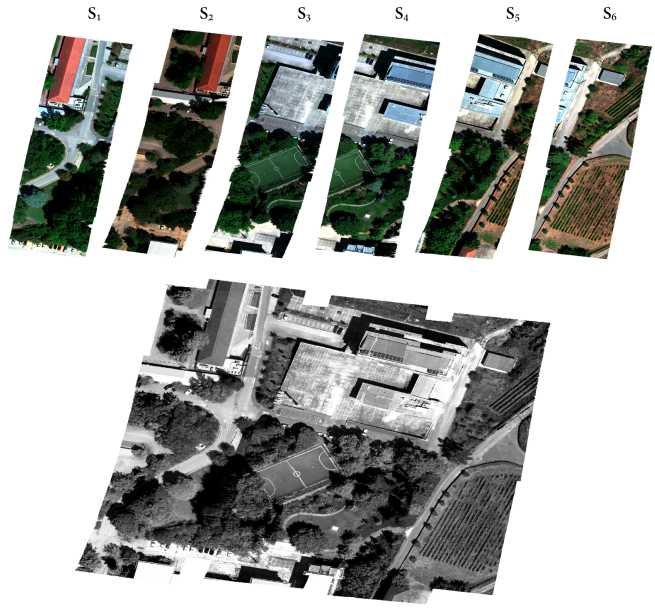


Figure 8.2: RGB hyperspectral swaths and the reconstructed orthomosaic using penalty-based functions (green wavelength).

aggregated radiance value, or at least, that which minimizes the dissimilarity from the initial samples.

The fusion of hyperspectral swaths was performed in the GPU following the workflow in Figure 8.3. Therefore, the hyperspectral orthomosaic is rapidly constructed to generate a result such as the one depicted in Figure 8.2. The workflow is split into several stages according to the definition of penalty functions, with the complete procedure being iterated for each hyperspectral layer and orthomosaic. Firstly, aggregation outputs are computed for every hyperspectral layer and stored in a SSBO. Some of these aggregations are entirely solved at the end of this stage, e.g., max and min, while mean-based aggregations require an additional stage and thus were solved at the start of the following distance-measuring stage. Then, images must be iterated again to compute the result of local penalty functions whose error is accumulated per aggregation operator, once again, storing it in an SSBO. The last stage is aimed at selecting the operator that minimizes the dissimilarity to finish the construction of the hyperspectral orthomosaic (see Figure 8.4).

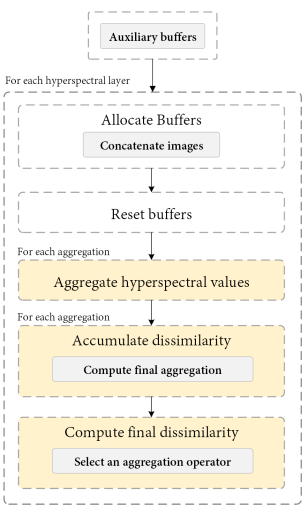


Figure 8.3: Overview of the aggregation procedure in the GPU.

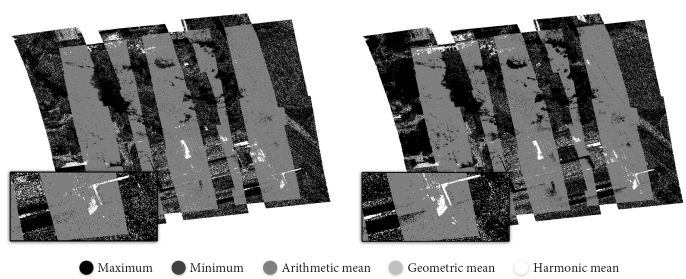


Figure 8.4: Point cloud rendered according to the selected aggregation operator for red (a) and green (b) wavelengths. The operator used in non-overlapped areas is irrelevant (indeed, it is the first one: max).

For optimization purposes, hyperspectral images were represented as `uint8_t` values that work as a rendering-based discretization of the starting radiometric resolution (in the order of 2^x). On the other hand, aggregation buffers were allocated using the `uint` data type since some operators require the sum and product of radiance samples from multiple swaths. This procedure is performed band-wise since the optimal aggregation may vary among different wavelengths. Penalty functions were implemented as subroutines, and therefore these can be easily modified in real time.

8.4 Projection of hyperspectral orthomosaics

The next stage is to project the hyperspectral orthomosaic into a 3D point cloud placed positioned in the UTM coordinate system. The input point cloud was reconstructed with SfM using RGB imagery, which led to a dense RGB point cloud. On the other hand, hyperspectral swaths were acquired with a *nadir* forward vector which, together with the limited scanning spatial resolution, leads to a slightly different projection that does not require occlusion checks. Instead, the point cloud will be generated as a heightfield, i.e., a 2.5D point cloud whose dimensionality depends on the hyperspectral resolution. Figure 8.5 depicts the heightfield of an RGB point cloud rendered with uniform colour.

Points are processed in the GPU using their height and index within a point cloud batch. The selection of the point with the highest altitude is implemented through an atomic block operating with `uint64_t` values. The first 32 bits represent the altitude (h) and the least significant bits store the point index. Although $h \in \mathbb{R}$, it can be transformed into $h \in \mathbb{N}$ with `floatBitsToInt` using its bit-level representation. Hence, the `atomicMax` operator selects the highest altitude while it also carries the index. The result is a buffer that stores the indices of those points which are visible from hyperspectral swaths. From this stage, the hyperspectral point cloud can be straightforwardly computed by transferring hyperspectral layers as textures that can be sampled in the GPU. However, note that large RGB point clouds are downsampled according to the hyperspectral resolution. For instance, a point cloud of 330M points was narrowed to 17.5M points with a hyperspectral's GSD of 0.067 m. Also, the dimensionality of the RGB point cloud is further reduced since some RGB points are not visible in hyperspectral data. This scenario occurs when the sampled texture coordinates, u, v, are not in $[0, 1]$, or invalid colours are sampled from the hyperspectral orthomosaic ($\alpha < 1$). The latter is derived from the computed orthomosaic being stored as a rectangular texture, where background pixels are marked using the alpha channel ($\alpha \leftarrow 0$).

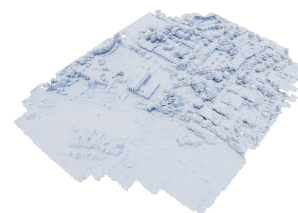


Figure 8.5: 2.5D voxelization of an RGB point cloud.

8.5 Hyperspectral data compression

The main drawback of this projection approach is the large dimensionality of hyperspectral point clouds despite being downsampled, especially for

GPU-based buffers with a very limited capacity. In this work, a 2.5D hyperspectral point cloud of 17.5M points with 270 layers was represented with `uint8_t` data type, which yields a total memory footprint of 4.4 GB. The compression and feature selection/transformation of hyperspectral datasets have been widely studied for reducing its footprint [396, 397] and extracting the most relevant features [398]. Regarding compression algorithms, there exist lossless approaches for hypercubes, though the data recovery is not as straightforward as in the representation proposed by Graciano et al. [119]. For instance, the CSDS-123 [396, 397] is a standard algorithm to reduce the size of multispectral and hyperspectral data used in onboard satellites and military drones. Although it has been accelerated in the GPU [399], it is not aimed at providing a real-time readable data representation.



Figure 8.6: The 70th layer of the Pavia University hypercube.

Despite hypercubes having a large number of changes throughout the spectral dimension, layers can be significantly compressed if their radiometric resolution is narrowed to the interval $[0, 2^8[$. Then, the hyperspectral point cloud can be compressed using a stack-based representation, as proposed by Graciano et al. [119], thus exploiting the similarity of surrounding layers in the Z-axis, i.e., the spectral signatures (Figure 8.7). With this stack-based approach, the heightfield is first composed of n voxels, determined by the number of points visible from *nadir* hyperspectral viewpoints. Then, these n voxels are compacted by fusing layers that store the radiance (1 byte) and the number of consecutive layers with such a radiance value (2 bytes, as the number of layers in one pixel may be above 2^8). Therefore, the compact layer representation requires 32 bits (`uint32_t`) split as follows: number of layers (16 bits) and grayscale radiance (rest, 16 bits).

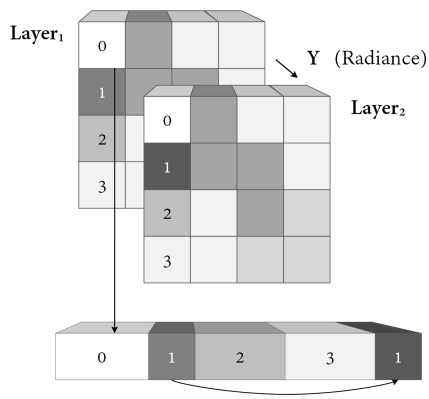


Figure 8.7: Stack-based representation for a hypercube with two layers. Only the first column is represented in the compact buffer.

Although this stack-based structure compressed the hypercube, its encoding presents a larger memory footprint (32 bits) in comparison with the initial 8 bits. Therefore, this approach is backed up by a significant compression that reduces the allocated memory albeit having a larger footprint per primitive. To prove that, this compression algorithm was tested against publicly available hypercubes [400] as well as the proposed dataset, obtaining the following compression ratio: ours (39.77%), Pavia Centre (56.74%; Figure 8.6), Pavia University (53.93%), Kennedy Space Center (97.70%), Salinas Valley (47.12%) and Cuprite (59.20%) datasets. Similarly to previous chapters, large point clouds do not fit in a single SSBO and must be split

into several batches, whose size depends on the data structure representation, the maximum number of layers and the allocatable memory. This dimensionality is calculated prior to compression as stated in Equation 8.1:

$$n_{points} = \lfloor \frac{\max_{\texttt{gpu_bytes}}}{\texttt{sizeof(uint32_t)} * \max_{\texttt{layers}}} \rfloor \tag{8.1}$$

8.5.1 Point cloud rendering

The visualization of hyperspectral point clouds is relevant as it enables visual inspections and band-wise rendering. To tackle the lack of density that may affect hyperspectral point clouds, these have been rendered using OpenGL's compute shaders. To this end, the approach of Schütz et al. [122] was followed, obtaining results such as the one depicted in Figure 8.8. The point cloud was sorted according to the 3D Morton encoding with 30 bits, and batches of small size were shuffled to avoid load unbalancing in the GPU. Nevertheless, red and green channels were discarded during the accumulation stage of this procedure; instead, only the blue (radiance) and α (number of points) channels were used.

Colours are retrieved by traversing stacks from the described compression data structure, evaluating the target layer as well as the number of repeats per stack layer. The main concern in the rendering is the frame rate for the last stacks, as the traversal time is higher and loops are very inefficient in shaders. To tackle this, images are iteratively constructed in multiple frames instead of a single one, thus reducing the latency of single frames. The user visualizes an incomplete, yet sufficient, point cloud that helps to find the better camera location until the image is fully constructed in a few frames. A noise buffer N, where $N \in \mathbf{N}^k, N_i \in [0, n_{points}[$ and n_{points} is the number of points, indicates which points are rendered in every frame. It can be shifted in subsequent frames to render every visible point and converge into a fully rendered image. The number of frames required to converge depends on the buffer length, k. Finally, Figure 8.9 shows several stages of progressive rendering.



Figure 8.8: Rendering of a point cloud with the Green wavelength, using the optimization described by Schütz et al. [122].



Figure 8.9: Iterative rendering of a hyperspectral point cloud during several frames, using 10k points per frame.

8.6 Results and discussion

The performance of the proposed method has been evaluated using point clouds and images of different sizes. The baseline RGB point cloud is generated with two different numbers of points, 150M and 330M, while other subsampled point clouds will be used for further assessing the performance. Due to the lack of previous work concerning the generation of 3D hyperspectral point clouds, this section is solely focused on the performance of individual stages, from the generation of hyperspectral orthomosaics to data compression and rendering. The reported results are obtained as the average from five different executions.

Measurements were performed on a PC with Intel Core i7-7700 3.6 GHz, 16 GB RAM, NVIDIA GTX 1070 GPU with 8 GB VRAM (Pascal architecture), and Windows 10 OS. The proposed methodology is implemented in C++ along with OpenGL. Massively parallel processes were developed in GLSL using general-purpose compute shaders, whereas CPU-based methods are accelerated using the multi-threading OpenMP library.

8.6.1 Study area and data acquisition

The study area is located at the University of Trás-os-Montes e Alto Douro campus, covering 4 ha. Its altitude is 500 m, though it varies up to 30 m in the surveyed area. GCPs were uniformly distributed to provide high positional accuracy in contrast to the UAS's GNSS receiver. Therefore, the collected images were georeferenced using five GCPs depicted as circular targets with a diameter of size 0.5 m. Besides GCPs, control points were also used to assess the quality of the results. Both GCPs and CPs were measured with a GNSS receiver based on the TM06/ETRS89 coordinate system.

Hyperspectral swaths were collected with 40% side overlap at an altitude of 100 m, thus acquiring 6 different hyperspectral images. The RGB dataset is composed of 324 RGB images acquired with a spatial resolution of 3.4 cm, 90% frontal overlap and 75% side overlap at an altitude of 80 m from the take-off position. The latter is processed with Pix4Dmapper to obtain the baseline RGB point clouds.

8.6.2 Performance analysis

The performance is assessed for every stage against a sequential approach. Firstly, hyperspectral images were aggregated to compute the hyperspectral orthomosaic. Although hyperspectral swaths have a limited resolution, the stitching procedure from Jurado et al. [102] generates a large hyperspectral orthomosaic. Hence, the response time has been assessed for different image resolutions by upscaling and downscaling the collected swaths. As shown in Figure 8.10, the response time was very reduced even for upscaled imagery (2x, 4x) despite this process being repeated once for every hyperspectral layer. Thus, the aggregation was solved in 130 ms for a single layer using the starting resolution, whereas the half-size setup achieved a significant

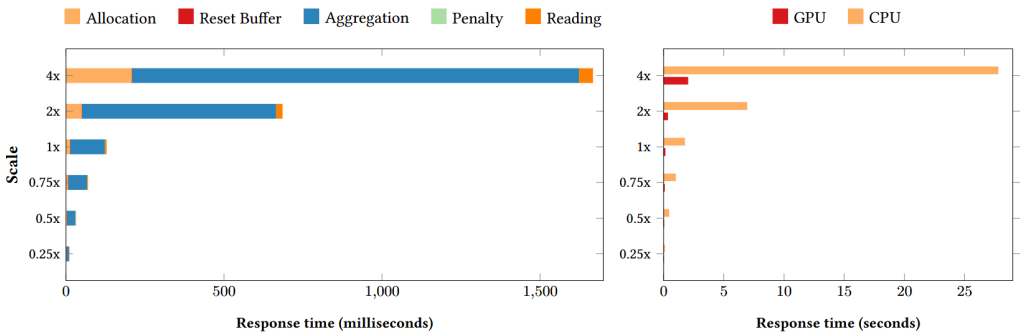


Figure 8.10: Response time in ms for the GPU-based approach, stacked per aggregation stage, whereas the chart in the right side compares the performance of both CPU and GPU-based methods.

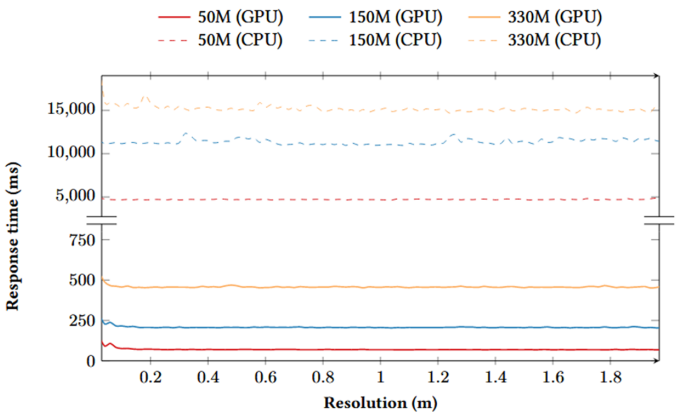


Figure 8.11: Performance of 2.5D voxelization for three different point clouds. Dotted lines show the results of the CPU-based approach, while solid lines represent the GPU-accelerated one.

speedup (74.69%). The aggregation stage is the main bottleneck since it comprises multiple operators. On the other hand, the allocation and reading phases had a higher response time as a result of transferring large buffers between CPU and GPU. Regarding the CPU versus GPU comparison, the speedup of the second approach is 92.54%.

After the orthomosaic generation, the point cloud is voxelized as a height-field to obtain which points are visible according to their height within a voxel. Hence, Figure 8.11 shows the response time for building heightfields over point clouds of different dimensionality. The resolution varies from 0.02 m to 2 m, including the default hyperspectral resolution (0.067 m). Measurements integrate both the allocation stage and the thread logic. Although input point clouds are very large, the voxelization was efficiently solved in the GPU using a single atomic operation for points projected in a voxel. It was solved in 500 ms since the processing is performed for each point regardless of the resolution. In contrast to the GPU approach, the sequential algorithm notably increased the latency for larger point clouds (+96.66% for 330M points).

Then, the projection and compression stages are performed using the visible points and the previously stitched orthomosaic. Figure 8.12 shows the response time for building a hyperspectral point cloud and compressing it.

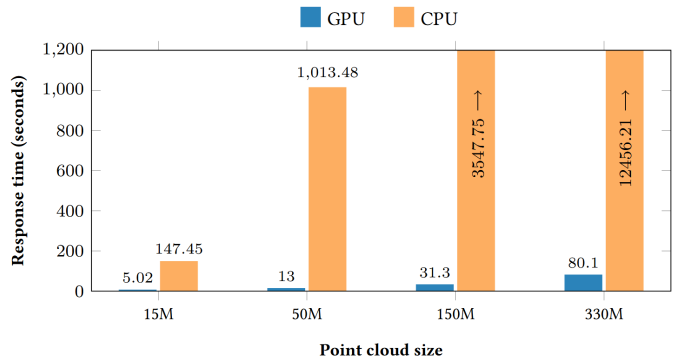


Figure 8.12: Response time (s) for building a hyperspectral point cloud with 270 spectral bands.

To this end, this procedure was evaluated using four point clouds with a size ranging from 15M to 330M, whereas the heightfield was constructed with a resolution equal to the GSD from hyperspectral data. Larger point clouds obtained a higher compression rate as they also had higher density and a larger number of voxels, with some of them showing little variance in the spectral dimension. The response time ranged from 5 s for 15M points to 80 s for 330M points. Hence, the first point cloud was compressed by iterating through a few point cloud subdivisions, whereas the last one requires over fifty subdivisions. Hyperspectral images were iteratively used for each subdivision, though its transferring time is mitigated by building a large buffer composed of the whole set of layers instead of uploading each one multiple times. Therefore, the initial delay is significantly higher, though it provides a speedup in comparison with a continuous data stream to the GPU. The speedup of GPU-based approaches in contrast to CPU is considerably higher in this stage since it involves multiple iterations. Accordingly, the improvement ranges from 96.59% (15M) to 99.35% (330M).

Finally, the frame rate for rendering the hyperspectral point cloud is shown in Figure 8.13. The target layer is iteratively increased for each new frame over two point clouds of size 150M and 330M. On the other hand, activations refer to the length of the noise buffer, thus referring to the maximum number of points rendered per frame. In this regard, two different activation values were used: 5k and 10k per frame. As depicted, the main drawback of stack-based compression is the traversal of the data structure to retrieve the original data, since deeper target layers also involve a larger number of iterations and response time. However, a lower activation size reduces the response time derived from constructing a single frame, whereas the complete scenario is generated in a reduced time. Finally, the number of point cloud subdivisions also worsens the performance since it increases the number of compute shader calls. Note that even the first layers are affected by the latter drawback.

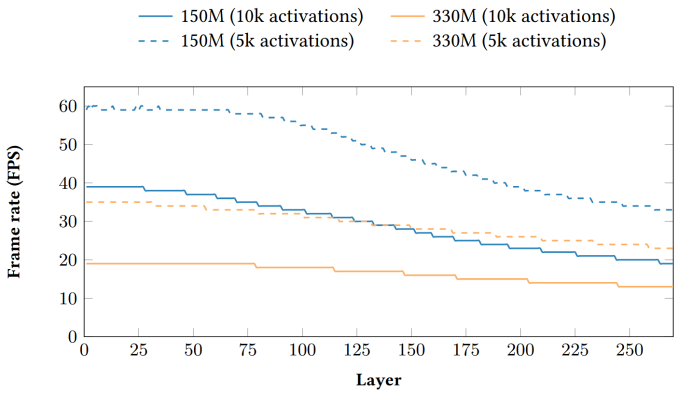


Figure 8.13: Frame rate for rendering hyperspectral point clouds with an increasing target layer. Activations refer to the number of points which were rendered in every frame.

8.7 Conclusions and future work

In this work, a method for the efficient generation of hyperspectral point clouds has been proposed. According to the limitations of hyperspectral acquisition, the collected swaths were first stitched to construct a large hyperspectral orthomosaic. Then, it was projected into the voxelization of an RGB point cloud, thereby constructing a 2.5D hyperspectral point cloud. The resulting heightfield was compressed with a stack-based representation and rendered iteratively to enhance the visualization performance. The whole pipeline was implemented in the GPU, thus showing speedups over 90% in comparison with the CPU-based stages, even for sparse point clouds (15M points).

As a future work, the rendering pipeline could be further improved together with the stack-based compressed data, mainly addressing the traversal time in compute shaders. Also, GPU buffers allocated for the hypercube could be further compacted, maybe exploring other efficient compression approaches, to reduce the number of point cloud subdivisions and the allocated GPU's memory.

Compression of spatial data from hypercubes with QuadStack

The QuadStack compression method proposed by Graciano et al. [119] also operates in the spatial dimensions, although the used datasets were further stressed in this regard and they showed a poor performance. Hence, spectral compression was proven to be possible with a lower radiometric resolution, whereas spatial compression does not work well with this approach.

Part IV

Generation of synthetic data

This part is split into two chapters. The first one covers the efficient simulation of a LiDAR sensor aimed at providing large point cloud datasets for the training of Deep Learning networks. Then, the LiDAR datasets are further enriched with intensity data by modelling the surfaces' BRDF.

Simulation of LiDAR sensors 9

About this chapter

This chapter presents a synthetic GPU-based LiDAR scanner. It is parameterized to emulate a large number of sensor models, ranging from airborne to terrestrial scanning. The approach is massively parallelized in the GPU to solve millions of intersections with a reduced response time, even simulating multiple returns for scenarios aimed at modelling forestry. This work is mainly intended for the rapid generation of large LiDAR datasets for Deep Learning. The conducted experiments show that the proposed approach outperforms a sequential approach, and the capabilities for constructing large datasets were proven to enhance previous work.

Another key factor is the use of procedural 3D environments; scenes modelled by professionals can be utilized for generating a few datasets using several viewpoints or paths, whereas those governed by generation rules help to construct a large number of datasets depicting different environments. Besides TLS, which is by far the most studied kind of LiDAR sensor, this chapter is also intended for describing how to simulate aerial surveys comprising sensor errors and surface properties. In comparison with previous work, collisions are solved using state-of-the-art ray-tracing data structures, instead of z-buffers with a limited resolution (Figure 9.1). Consequently, this piece of software is able to construct high-quality point clouds with low latency.

In comparison with previous work, the main contributions of this chapter are the generation of large labelled LiDAR datasets from procedural scenarios. These are obtained in a reduced response time following a physically-based interaction, while a sequential approach requires several minutes, even hours, for solving complete scans of the inspected scenario. The overall workflow is depicted in Figure 9.2.

9.1 On the generation of LiDAR datasets 165
9.2 Modelling of synthetic environments 166
9.2.1 Forestry environments . . 167
9.2.2 Urban environments . . . 168
9.3 Spatial indexing 170
9.4 Labelling of triangle meshes 170
9.5 LiDAR simulation 171
9.5.1 Pulse modelling 171
9.5.2 Collision solver 177
9.6 Results and discussion 182
9.6.1 LiDAR response time . . . 182
9.6.2 Generation of synthetic datasets 184
9.6.3 Visual results 185
9.7 Conclusions and future work 189

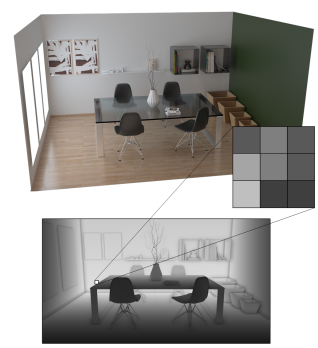


Figure 9.1: Depth buffer of a 3D scene, as proposed in previous LiDAR simulations.

9.1 On the generation of LiDAR datasets

LiDAR technology has rapidly evolved in the last two decades and therefore, it has received great attention from industrial and academic environments. This sensor enables acquiring information about a surface, object or phenomenon without physical contact. There is a wide variety of LiDAR variants according to the sensor capabilities, the platform on which they are operated (TLS, ALS, MMS, BMMLS, etc.), manufacturers and models. Unfortunately, it is not easy to get a sufficient amount of ground truth data due to time constraints and available resources. In contrast to real sensors, the LiDAR simulation enables the generation of classified data at a significantly lower cost. The LiDAR technology is prohibitive, and collecting

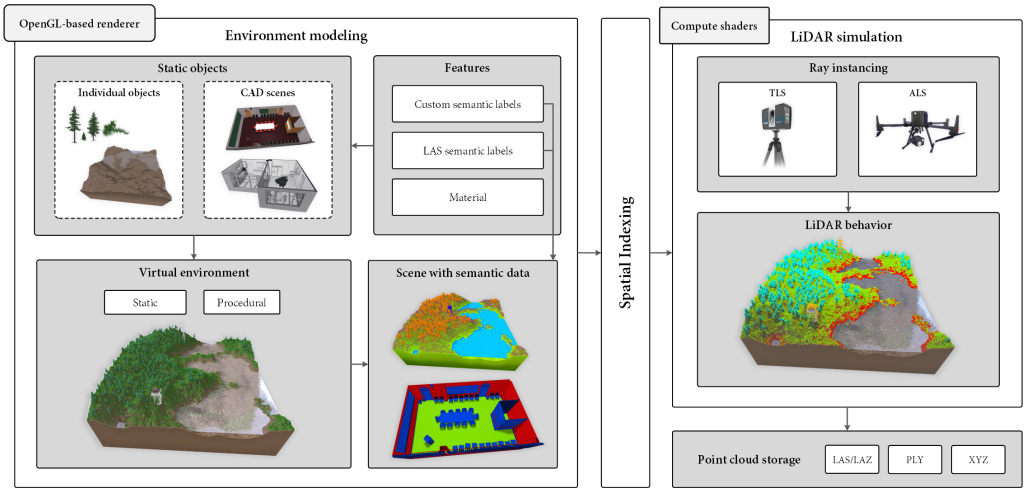


Figure 9.2: Overview of a LiDAR simulator operating over 3D virtual scenes, either procedural or static, which are previously indexed to speed up spatial queries.

large real-world datasets is time-consuming, both in the acquisition and labelling (if required).

On the other hand, simulated LiDAR data can be enriched with any kind of information. For instance, semantic tags can be accurately determined since the models in the input scenarios are named accordingly. The LOD of augmented data, including semantic tags, can be constrained by the LOD of input environments as well as by the requirements of decision-making and classification algorithms that will use these synthetic datasets. Also, simulations can go further and integrate errors and limitations of LiDAR, either systematic or random, to provide more realistic data. Nevertheless, simulating the physical interaction of LiDAR sensors with real-world surfaces is a non-trivial task that demands computationally efficient algorithms.

9.2 Modelling of synthetic environments

Synthetic scenarios must be similar to real-world environments since the scanning results are aimed at feeding DL algorithms. Furthermore, procedural scenarios are preferred over those modelled by professionals in the generation of large datasets, since the former lead to a greater variety. Following this approach, scenarios can be generated using rules that determine how will be distributed a collection of 3D models. In contrast to static scenarios, the labelling of procedural environments is more efficient as it is only performed once. Although there exists third-party software for the generation of procedural environments, such as SpeedTree®, a custom tool was implemented to meet our specific requirements and enable the labelling of every model instance.

9.2.1 Forestry environments

The procedural generation of forestry scenarios has been previously approached with ecosystems [401–403] that achieve a high level of realism. However, this work is intended to provide a simple tool for the procedural generation of natural scenarios. To this end, the procedure depicted in Figure 9.4 can be configured using a text file or a node editor in the Graphical User Interface (GUI).

The soil was modelled using a Perlin noise function, followed by hydraulic erosion over the heightfield representation of the terrain. The erosion algorithm was enhanced using the GPU to model each particle with a different thread. Accordingly, the particles move according to a weighted aggregation of slopes from their surroundings, until a move of length zero or a limit of iterations is reached. During these moves, particles deposit sediment whether the carried material exceeds their capacity; otherwise, the heightfield is eroded. The erosion is applied as a convolution with a configurable radius, whose weights, w_i, represent the distance to the kernel centre. Regardless of the particle location, the convolution has always the same weights and thus can be computed once and stored. Figure 9.3 shows several stages of this hydraulic erosion for 200k particles. The surface got smoother when the erosion run for a larger number of iterations. Hence, a limited number of iterations helps to balance realism and latency.

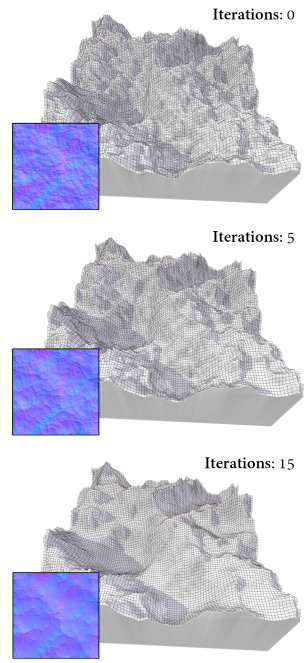


Figure 9.3: Hydraulic erosion of 200k particles over a heightfield modelled with 2D Perlin noise.

On the other hand, water is modelled as planar surfaces placed at valleys like the ones depicted in Figure 9.5. The procedure is to locate local minima defined as points whose elevation is lower than surrounding pixels. Then, local minima points are flooded until the height h_{flood} is achieved, thus obtaining which cells are part of a valley. h_{flood} is computed as shown in Equation 9.1, using a reference height, h_{water}, and a random value in $[-h_\sigma, h_\sigma]$, both obtained from the scene configuration. The set of collected valleys is sorted according to the covered area to retrieve only the top-n instances.

$$h_{flood} = h_{water} + f_{random} h_\sigma \tag{9.1}$$

where $f_{random} \in [-1, 1]$.

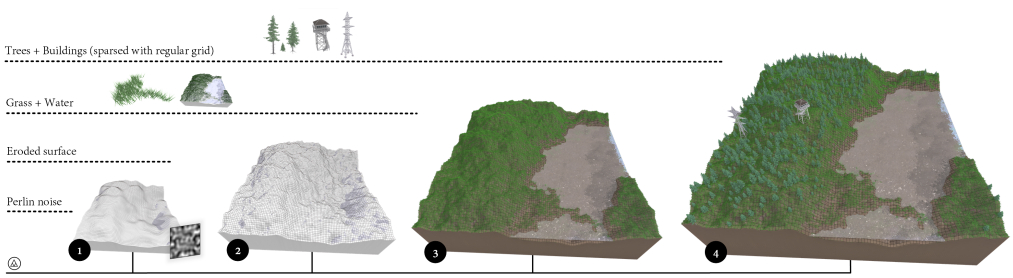


Figure 9.4: Workflow of the procedural modelling of a forestry area with water, low-vegetation, high-vegetation and buildings. First, the soil is eroded using hydraulic processes, and then, water and low vegetation are procedurally generated. Finally, trees and buildings are generated so that collisions among them are avoided.

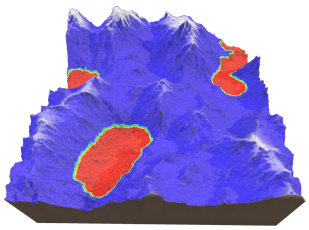


Figure 9.5: Valleys detected on an eroded terrain.

Low vegetation is represented with a large number of low-poly instances that enable the modelling of highly detailed scenes while keeping a good rendering performance. Still, OpenGL's instance rendering is required for providing a high value of FPS. Otherwise, geometry shaders could be used; however, the LiDAR simulation requires explicitly declaring the vegetation geometry in a SSBO for the spatial indexing. The distribution was computed using seeds from a random uniform distribution, with the slope and height being key factors to determine whether they are instanced or not. Thus, the instancing probability is defined as shown in Equation 9.2. The probability map is compiled in a texture that aggregates the height and normal map from previous stages. According to Equation 9.2, points that present steeper slopes and lower altitudes have a lower instancing probability.

$$p_{i,j} = 1 - w_{height} \cdot H(h, h_{min}, h_{max}) - w_{slope} \cdot (1 - \hat{n} \cdot [0, 1, 0]^\intercal) \quad (9.2)$$

where $H(h, h_{min}, h_{max})$ is an Hermite interpolation function in $[h_{min}, h_{max}]$, and $\hat{n}$ is the normal vector at (i, j), given that $i \in [0, width[$ and $j \in [0, height[$, with $(width, height)$ being the size of the eroded heightfield.

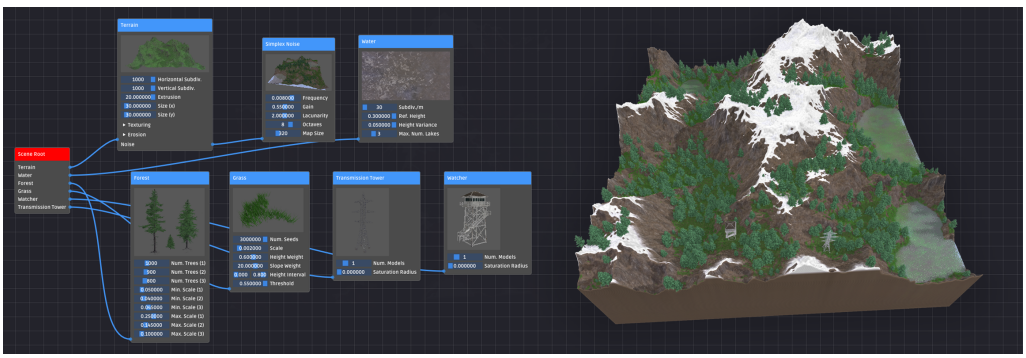


Figure 9.6: Configuration of the node editor for constructing an environment such as the one depicted on the right side.

In contrast to low vegetation, the instancing of trees also takes into account previously instanced geometry to avoid intersecting it. Instead of checking collisions among triangle meshes, it was simplified using a 2D regular grid that narrows the problem to check whether a cell is already occupied or saturated. According to this, the grid cells are saturated with a factor of $f_{saturation}$ within a radius r when a new object is instanced. On the other hand, a grid cell is considered to be saturated whether its occupation is over a threshold, $h_{saturation}$. Therefore, the density of vegetation is coarsely controlled with the grid subdivision and $h_{saturation}$. Whenever a grid cell must be occupied by a single model, it can be saturated with a value of ∞.

9.2.2 Urban environments

Urban scenarios are partly modelled as procedural environments (see Figure 9.7). The baseline is a large city and residential scenario modelled by

professionals, whereas pedestrians and vehicles are procedurally instanced. To this end, candidate seeds are sparsed over sidewalks and road models and instanced whether the linked random value is below a threshold. The instancing frequency as well as the varying scale of each model can be configured.

Firstly, roads are split into rails according to their width, and then, vehicles are rotated using the road rotation matrix. The location is computed by selecting a random road and rail, as well as a random translation within it. On the other hand, the y coordinate is computed as the collision of a ray emitted towards the road with *nadir* direction, plus half the height of the vehicle's bounding box. The main concern during the instancing of vehicles is to avoid inter-collisions. Hence, these can be detected by building the BVH of each different model and rapidly determining if there are collided polygons. However, doing this for every instance is very time-consuming; instead, BVHs are built for every model in the database, and they can be changed according to the instance's transformation matrix (Figure 9.8). Still, transforming every vertex is time-consuming, and thus, collisions between triangles can be identified by casting rays with the shape of the edges of every source model's triangle. With this approach, three rays are cast for every triangle, with its coordinates being transformed as shown in Equation 9.3. Rather than comparing the new instance with the rest, it is narrowed using a 2D regular grid whose subdivision length corresponds to the maximum xz length from any collected vehicle. Similarly, the source model from which rays are emitted is selected as the one with the lower number of polygons.



Figure 9.7: Procedural urban environment based on a) a residential area and b) a metropolis.

$$\begin{aligned} r_o &= {C_T}^{-1} \cdot C_S \cdot v_{source} \\ r_t &= {C_T}^{-1} \cdot C_S \cdot v_{target} \\ r_d &= \widehat{r_t - r_o} \end{aligned} \tag{9.3}$$

where r_o, r_t and r_d are the ray's source, target and direction, and the vertices v_{source}, v_{target} are part of a triangle edge. Composite matrices (C) with T subscript correspond to the target model, whereas S refers to the source instance, i.e., the model with a lower number of polygons.

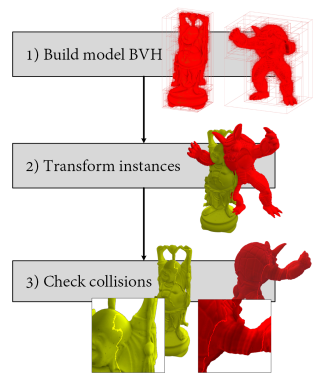


Figure 9.8: Detection of collisions between two triangle meshes.

On the other hand, translation and rotation vectors are significantly more randomized for pedestrian instancing. Pedestrian models are randomly selected, whereas their location and rotation in the y-axis are described using three random values (t_x, t_z, r_y). As proposed before, the y coordinate can be obtained by casting a ray with *nadir* direction. The detection of inter-collisions is addressed as explained, though pedestrians may also collide with buildings. Thus, it requires building the BVH of non-populated urban objects. Finally, procedural trees are also generated in some areas, and they can be upscaled and downscaled with a user-defined interval.

9.3 Spatial indexing

The objective of the synthetic LiDAR that will be following proposed is to estimate returns with high precision. To this end, these will be computed with ray-casting in 3D, rather than solving it in the image space. However, highly detailed scenarios are composed of several millions of triangles and LiDAR simulations also comprise millions of rays. Despite ray-triangle intersections being efficiently solved in the GPU, the main concern is the spatial traversal since it is considerably more time-consuming. Thus, this is a well-known time bottleneck even for optimized algorithms.

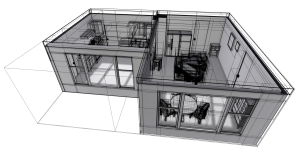


Figure 9.9: Rendering of the BVH indexing of the very same scene in 9.11. It was constructed using a buffer radius, r, of size 100.

Either static or procedural, scenarios are indexed using a BVH which can be rapidly traversed for millions of rays. The indexing was performed as proposed by Meister and Bittner [146] in the GPU. Hence, it is efficiently generated and after this, the BVH is stored in the GPU's memory to solve the subsequent scans, thus avoiding time-consuming transfers between CPU and GPU. For instance, this GPU-based approach enables building a BVH for ten million triangles with a response time below 1 s, as reported in Figure 2.26. On the other hand, the traversal time is improved with several threads working in parallel. The naïve traversal starts by the root and descends into the tree while pushing nodes into a stack. Hence, the topmost node of this stack is checked for intersections. The ray-AABB intersections are rapidly solved with the algorithm proposed by Majercik et al. [404], whereas leaf nodes are checked with the ray-triangle test described by Möller [405].

9.4 Labelling of triangle meshes

Instead of manually labelling large LiDAR point clouds, human operators are required to label objects in a 3D triangle mesh. This task is facilitated using a hierarchical segmentation that is sped up using the naming conventions of objects and materials (see Figure 9.10). During the mesh loading, object names are considered significant whether they do not collide with other instances' names. Otherwise, they are named after the material together with a unique `id`. Furthermore, models are organized either by following a custom tag or standard labels defined in the LASer (LAS) file format (see Figure 9.11). The former can be collapsed or further subdivided according to the required LOD, thus helping to provide a fine-grained segmentation. In contrast to the manual classification of real LiDAR point clouds, annotated environments help to suppress labelling errors and provide a fine-grained segmentation. Otherwise, manual labelling should be limited to a few classes to avoid overwhelming human operators in this task.

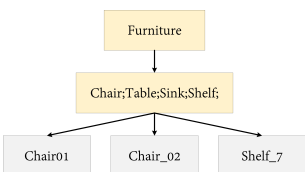


Figure 9.10: Example of hierarchical annotation, where the furniture label includes a wide range of items.

Besides semantic labels, models are also linked to materials that help in computing the LiDAR intensity. This property will be further described in the following chapter. Finally, the identifier of each instance is also stored to make the simulated datasets appropriate for instance segmentation.

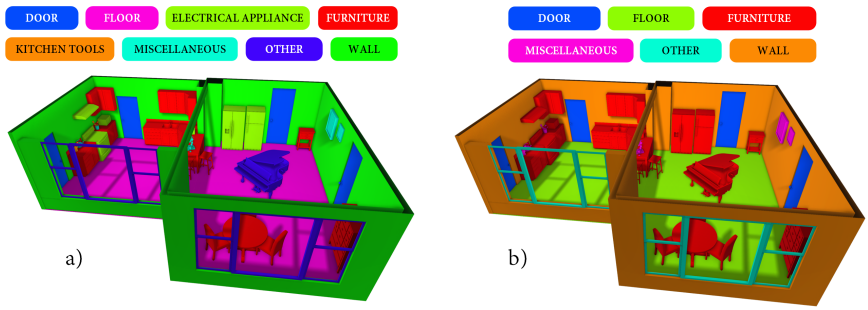


Figure 9.11: Different criteria for labelling a static scenario. First, the scene is annotated using custom labels, whereas the second classification is performed with semantic labels in the LAS 1.4 standard.

9.5 LiDAR simulation

This section is structured according to the steps followed in the GPU-based LiDAR scans. First, a comprehensible explanation of LiDAR pulse modelling is provided, followed by the interaction of LiDAR beams and synthetic surfaces. The latter stage will also help to introduce the pipeline in the GPU. The generic parameters used throughout this chapter are explained in Table 9.1.

9.5.1 Pulse modelling

LiDAR pulses are modelled as a set of rays emitted at the sensor's location that propagate towards uniformly sampled points in a unit sphere. From now on, the emitter and receiver will be assumed to be placed in the same location to simplify the conceptualization of the LiDAR mechanism. Rather than a single ray, the physical behaviour of a LiDAR pulse is better simulated with multiple rays [241] (check Figure 9.12) within a radius that can be static (collimated beams; parallel rays) or dynamic (diverging beams). Rays in the same pulse are calculated according to the orthonormal basis $(\hat{u}, \hat{v}, \hat{r}_d)$ formed by the ray direction and the $\hat{up}$ vector. $\hat{u}, \hat{v}$ are then scaled with random numbers to generate points within a circumference of radius r.

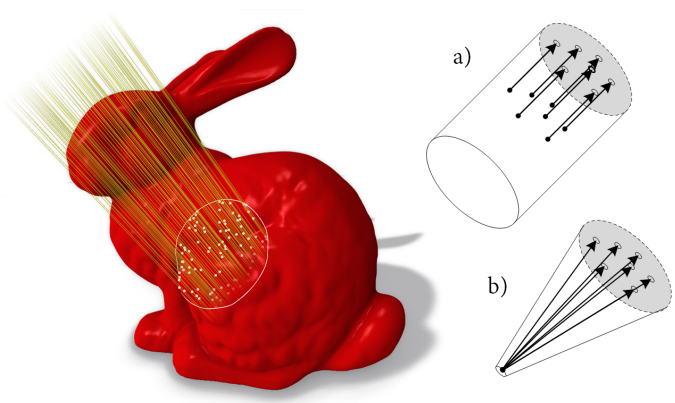


Figure 9.12: Collimated rays from the same pulse intersecting the Bunny model. The figures on the right side show the scheme of a) collimated rays and b) diverging rays.

Table 9.1: Baseline configuration parameters of LiDAR simulations. These parameters are in common to ALS and TLS.

Parameter	Description
Wavelength (λ)	532 nm for bathymetric scans, 1064 nm otherwise.
Pulse radius (r)	Radius at a distance of $t = 1$ following the ray parametric equation.
Rays per pulse (n_p)	LOD of the pulse discretization.
Energy ($\bar{I}$)	Energy linked to a LiDAR pulse, given in W.
Sensor diameter (p_l)	Diameter of the LiDAR's receiver component (m).
Maximum number of returns (n_r)	Number of maximum returns per pulse.
Maximum range (R)	Maximum range covered by a single pulse.
Fuzzy boundary around R ($R_{\delta_{min}}, R_{\delta_{max}}$)	Fuzzy boundary to avoid abrupt losses when R is reached.
Terrain-induced error ($e_{terrain}$)	Boolean value indicating if errors related to slopes are simulated.
Shiny surface error (e_{glossy})	Boolean value indicating if the 'time-walk' effect is simulated.
Outlier threshold ($t_{outlier}$)	Probability of generating outliers.
Atmospheric attenuation (a_{atm})	Atmospheric attenuation factor (see Equation 2.3).
System transmission (n_{sys})	System transmission factor (see Equation 2.3).

Table 9.2: Parameters which can be configured on the propagation of laser beams from TLS. Position-related factors are expressed in m, whereas angular parameters are expressed in radians.

Parameter	Description
Position (p)	Position of LiDAR's emitter.
Field of view ($\alpha_{fov_{xz}}, \alpha_{fov_y}$)	Horizontal and vertical aperture ($0 \leq \alpha_{fov_{xz}} \leq 2\pi$ and $0 \leq \alpha_{fov_y} \leq \pi$).
Scanning resolution (r_x)	Number of horizontal scan lines.
Horizontal reference angle ($\alpha_{centre_{xz}}$)	It helps to calculate the starting horizontal angle as $\alpha_{centre_{xz}} - \alpha_{fov_{xz}}/2$.
Vertical reference angle (α_{centre_y})	It helps to calculate the starting vertical angle as $\alpha_{centre_y} - \alpha_{fov_y}/2$.
The number of channels (n_c)	Number of simultaneous channels.
Channel spacing (d_c)	Interleaved distance between consecutive channels.
Channel angular distance (d_{α_c})	Interleaved emission angle between consecutive channels.
Axis noise ($\hat{\delta}_{noise}$)	Magnitude of the jittering in the ray's direction.
Angle noise (α_{noise})	Magnitude of the angle used for jittering the ray's target position.

Terrestrial LiDAR (TLS)

The parameters concerning TLS simulations are described in Table 9.2. Rays in TLS spread in a sphere which is discretized through the horizontal and vertical resolution, r_x and n_c. Furthermore, horizontal and vertical FOV determine which part of the sphere is covered, since these angles can be lower than 2π and π, respectively. The FOV is parameterized with $(\alpha_{centre_{xz}}, \alpha_{fov_{xz}})$ as well as $(\alpha_{centre_y}, \alpha_{fov_y})$. Therefore, the horizontal FOV covers $[\alpha_{centre_{xz}} - \frac{\alpha_{fov_{xz}}}{2}, \alpha_{centre_{xz}} + \frac{\alpha_{fov_{xz}}}{2}]$, whereas the spacing of channels is d_c (by default, it is almost zero, $\epsilon \leftarrow 1^{-16}$). However, this ideal spatial subdivision leads to grid-like renderings (Moiré effect) of LiDAR point clouds, which can be interpreted as an artefact. Instead, this unwanted visual effect can be avoided by jittering [120] the emitted rays, thus leading to an imperfect result. Furthermore, it enables emulating beam jittering and minor inaccuracies as a result of the limited sensor accuracy [406]. The jittering changes both the ray's direction ($\hat{\delta}_{noise}$) and its magnitude (α_{noise}) with random values in $[-1, 1]$.

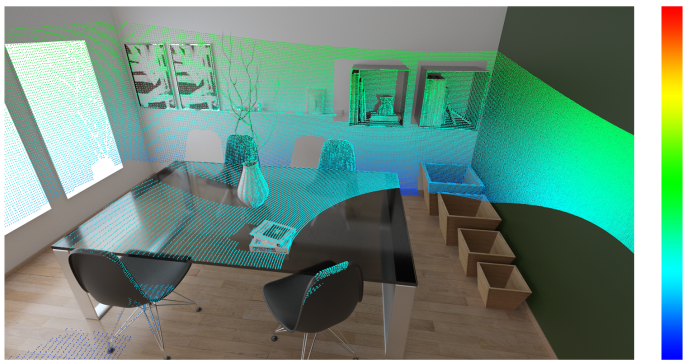


Figure 9.13: Terrestrial scan with colours encoding the point's relative height.

Equation 9.5 shows the scattering direction of a beam emitted from the TLS's emitter (Equation 9.4). Horizontal lines (α_{xz}) enable calculating the position Θ on the surface of a unit sphere with $y \leftarrow 0$, whereas the vertical lines (α_y) determine the elevation of such point. Also note that s_d and $\hat{\delta}_{xyc}$ are defined as orthogonal vectors from Equation 9.7. $r(u)$ shows the ray equation expressed through its origin, r_o, and target point, r_t.

$$r_o = p + \left[0, \frac{-d_c\,(n_c - 1)}{2} + c d_c, 0\right]^\intercal \tag{9.4}$$

$$r_t = r_o + \Omega_1 \left(\hat{\delta}_{noise} \begin{bmatrix} \lambda_x \\ \lambda_y \\ \lambda_z \end{bmatrix}, \alpha_{noise} \lambda_\alpha \right) \Omega_2 \left(\hat{\delta}_{xyc}, \alpha_y \right) \Theta \tag{9.5}$$

$$r(u) = r_o + u\widehat{(r_t - r_o)} \tag{9.6}$$

where intermediate results are calculated as follows:

$$\Theta = [\cos \alpha_{xz}, 0, -\sin \alpha_{xz}]^\intercal \tag{9.7}$$

$$\hat{\delta}_{xyc} = [\Theta_z, 0, -\Theta_x]^\intercal \tag{9.8}$$

$$\alpha_{xz} = \alpha_{centre_{xz}} - \frac{\alpha_{fov_{xz}}}{2} + \alpha_{fov_x} \left(\frac{x}{r_x} + \frac{y}{r_x n_c} - \frac{1}{2} \right) \tag{9.9}$$

$$\alpha_y = \alpha_{centre_y} - \frac{\alpha_{fov_y}}{2} + y \left(\frac{n_c \alpha_{fov_y} + \alpha_{fov_y}}{{n_c}^2} \right) \tag{9.10}$$

provided that Ω represents a rotation matrix defined by a rotation axis and scalar angle, whereas λ_x, λ_y, λ_z and λ_a are random values generated by a random uniform distribution. x and y are iterative values so that $0 \leq x < r_x$, $0 \leq y < n_c$. α_{xz} and α_y are horizontal and vertical angles, respectively. Accordingly, $0 \leq \alpha_{xz} \leq 2\pi$ and $\frac{-\pi}{2} < \alpha_y < \frac{\pi}{2}$.

Furthermore, other LiDAR models such as Pandar64 do not have uniform resolution along the vertical axis [407]. To cope with this kind of sensor, several intervals can be defined with the starting and limit angles.

Even this straightforward construction of TLS rays has a significant latency in high-resolution scans covering large areas. Thus, this construction stage was solved with a massively parallel approach in the GPU to handle

Table 9.3: Parameters which can be configured for generating rays from an ALS. Again, angles are expressed in radians.

Parameter	Description
Platform altitude (h)	Fixed elevation of the mobile platform.
Field of view (α)	Aperture of LiDAR sensor.
Speed of the platform (v)	Pace of the mobile platform in $\mathrm{m/s}$.
Number of scans (ξ)	Number of scan lines per second.
Number of pulses (ψ)	Number of pulses per second.
Spatial noise (δ_{noise})	Magnitude of distortion applied to the ray's target.
Altitude noise (h_{noise})	Magnitude of jittering in the platform altitude.
Scanning pattern	Pattern used to scan the scene.
Flattening of ellipses (e_f)	y-scale of ellipses (elliptical scan pattern).
Radius of ellipses (e_r)	Radius of ellipses in the slanted pattern.

every ray with a different thread, rather than in a sequential manner. In contrast to the CPU, one of the main drawbacks of GPGPU is the lack of noise functions required for randomizing the ray generation. Some pseudo-random generators are frequently used in GLSL, though noise was required to be uniformly distributed. This was solved by building a large buffer with uniform random values computed in the CPU, which are later used by GPU threads to randomize LiDAR simulations. Another challenge arises from the limited GPU memory since SSBOs expand to a few gigabytes at most, regardless of the GPU capacity. Therefore, the ray generation is split into several iterations that overwrite a single SSBO on which rays are stored, with each iteration being followed by the collision solver. Otherwise, CPU and GPU transfers would be larger to avoid exceeding the maximum capacity. The maximum number of allocatable rays in a single buffer is computed from the maximum capacity of an OpenGL's SSBO, the size of a single ray and the number of rays within a pulse, n_p, as shown in Equation 9.11. Note that rays within a pulse cannot be separated into different iterations since they behave synchronously.

$$n = n_p \left\lfloor \frac{\texttt{glGetIntegerv(GL_MAX_SHADER_STORAGE_BLOCK_SIZE)}}{n_p \cdot \texttt{sizeof(Ray)}} \right\rfloor \tag{9.11}$$

Aerial LiDAR (ALS)

TLS was explained so that beams were emitted from a single emitter location. However, both TLS and ALS can be configured to follow a path, thereby simulating terrestrial and aerial vehicles, respectively. However, the configuration of airborne paths is slightly more complex, though they can be automatically computed, and thus will be introduced with ALS. The parameters concerning ALS are described in Table 9.3.

In comparison with TLS, ALS pulses a spread over a plane rather than a sphere, and the sensor's emitter changes according to the path. Among other features, the FOV is typically narrowed to avoid including returns from atmospheric particles, and yet, noisy returns are very frequent. In this case study, the jittering affects the emitted rays and the height of the mobile platform, with a magnitude of δ_{noise} and h_{noise}, respectively. The resolution is parameterized by the number of scans and pulses per second. Instead of only performing parallel scans, other patterns are frequently used in ALS

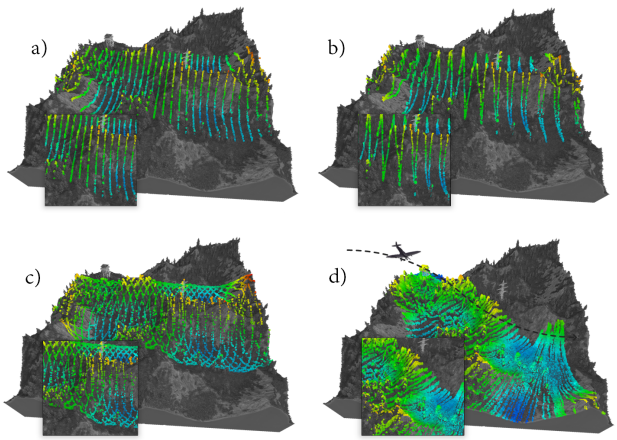


Figure 9.14: Multiple airborne scans following a) parallel, b) zigzag and c) elliptical patterns. d) illustrates a custom path defined by the user in the GUI.

to maximize the point cloud coverage. Figure 9.14 shows the aftermath of zigzag and elliptical patterns [84]. Hence, the configuration of ALS includes parameters that help to automatize the estimation of the platform path. Regardless of the scanning pattern, the airborne platform moves according to such a path, and therefore, the emitter location is calculated with a parametric value, t_i. On the other hand, the ray's source and target points, r_o and r_t, are computed according to the scanning pattern and t_i. The rays emitted with parallel and zigzag patterns are calculated as follows:

$$r_o = p_{t_i} + \begin{bmatrix} 0 \\ \lambda_h h_{noise} \\ 0 \end{bmatrix} + \frac{\xi p_{s_i}}{\psi} \vec{d} \tag{9.12}$$

$$r_t = r_o - \hat{\delta}_{sp} \sin\beta + \begin{bmatrix} 0 \\ -\cos\beta \\ 0 \end{bmatrix} + \delta_{noise} \vec{\lambda} \tag{9.13}$$

with the rotation angle β and the vector δ_{sp} being:

$$\beta = z_s \alpha \left(\frac{p_{s_i} \xi}{\psi} - \frac{1}{2} \right) \tag{9.14}$$

$$\delta_{sp} = \left[-\vec{d}_z, 0, \vec{d}_x \right]^\intercal \tag{9.15}$$

given that p_{s_i} is the index of a pulse within the i-th sweep, $\vec{d}$ is the vehicle direction, δ is a random value and $\vec{\lambda}$ is a random vector, both retrieved from a uniform distribution and finally, s is an iterative value such that $0 \leq s < \xi$. On the other hand, $z_s \in \{-1, 0, 1\}$ enables omitting the program branching for parallel scanning ($z_s \leftarrow 0$), whereas -1 and 1 are used for odd and even iterations in the zigzag pattern.

The elliptical pattern is slightly different as scans are now ellipses flattened with a factor e_f (Equation 9.16). Accordingly, β defines a circumfer-

ence flattened in the x-axis and centred in p_{t_i}.

$$r_t = r_o + \begin{bmatrix} e_f e_r \sin \vartheta \\ -h \\ e_r \cos \vartheta \end{bmatrix} + \delta_{noise} \vec{\lambda} \tag{9.16}$$

where the rotation angle, ϑ, and the ellipse radius, e_r, are calculated in Equations 9.17 and 9.18.

$$\vartheta = \frac{2p_{s_i} \pi \frac{aabb_x}{s_p \xi}}{\frac{aabb_x}{s_p \psi}} = \frac{2p_{s_i} \pi \psi}{\xi} \tag{9.17}$$

$$e_r = h \tan \frac{\alpha}{2} \tag{9.18}$$

Intuitively, e_r is computed with the tangent of $\frac{\alpha}{2}$ for any height h. The formulae were considerably simplified by momentarily assuming that $h \leftarrow 1$, and therefore, $r_{d_y} \leftarrow -h$. On the other hand, i is defined as the index of each incremental step, related to the parametric value t_i.

Scenarios can be scanned with ALS following parallel routes along the X or Z axis. Platform rotations involving changes in the *forward* direction are not simulated since these waypoints are typically discarded in post-processing. Hence, computer-aided paths simply consist of sparse lines parallel to the X-axis. The distance between them is calculated considering the dimensions of the scenario as well as the sensor's FOV (Equation 9.19), in order to exhaustively cover the scene. Otherwise, the path can be defined using a canvas in the GUI, though these require further processing. User-defined paths comprise points recorded in every frame as long as the painting cursor is being dragged, thereby generating paths with a huge number of (noisy) points. This shortcoming was tackled by simplifying the path with the Douglas-Pecker algorithm [408]: the path remains very similar, whereas the number of points is significantly lowered and the noise is mitigated. Then, points in this path are the control points of a Catmull-Rom spline which will be later sampled during the scan. Either computer-aided or defined by users, paths are sampled in the CPU and transferred to the GPU to make this routine simpler in the GPU processing.

$$n_{steps} = \lceil \frac{aabb_{min_z}}{2(h - aabb_{min_y}) \tan(\frac{\alpha}{2})} \rceil \tag{9.19}$$

where $aabb_{min}$ is the minimum point of the scene's AABB.

Finally, the emitted rays in ALS change considerably with respect to TLS; firstly, diverging laser beams within a pulse were simulated instead of collimated beams, as proposed by Zohdi [241]. Given a radius r defined at a distance $d \leftarrow 1$, rays are scattered following the Equation 9.20. For each ray, an orthonormal basis is constructed $(\hat{u}, \hat{v}, \hat{r}_d)$ using the ray direction, r_d, as well as an up vector, $\widehat{up}$, expressed as $[0, 1, 0]^\intercal$ for TLS. Accordingly, $\hat{u}, \hat{v}$ correspond to the X and Y axes in the ray coordinate system, and therefore, scaling both vectors by a random factor $\delta_r \in [-r, r]$ and adding them to the ray's target (r_t) leads to a diverging beam. For that purpose,

the point calculated in Equation 9.20, p_{d_i}, satisfies $\|p_{d_i} - r_t\| < r$, provided that i is bounded by the number of rays in a pulse, n_p, and the distance is measured by a simple Euclidean function. Consequently, LiDAR simulations are more precise with larger values of n_p, despite having a higher memory footprint.

$$\begin{aligned} p_{d_i} &= r_o + \hat{r}_d + \hat{u}\delta_r + \hat{v}\delta_r = r_t + \hat{u}\delta_r + \hat{v}\delta_r \\ \vec{u} &= \widehat{up} \times \hat{n} \\ \vec{v} &= \hat{n} \times \hat{u} \end{aligned} \tag{9.20}$$

9.5.2 Collision solver

TLS and ALS rays interact with surfaces to construct dense synthetic point clouds enhanced with semantic information. The emitted rays operate following the synchronous stages shown in Figure 9.15, regardless of the scanning platform. Particularities on TLS and ALS can be implemented through shader subroutines rather than different pipelines. The GPU-based stages are explicitly synchronized to avoid simultaneous reading and writing operations from different steps. Stages involving randomness use buffers generated with a custom random distribution and accessed with an offset equal to the thread index. The QMC (Quasi-Monte Carlo) Halton sampler was used along with the C++ built-in uniform random distribution, although other samplers could be integrated. Also, the number of rays processed during a single iteration must be a multiple of n_p.

The stages depicted in Figure 9.15 are following described:

Ray initialization

Initially, rays carry a value of energy equal to $\bar{I}/n_p$, with $\bar{I}$ being the energy of a single pulse in watts (W). The return number is set to zero and the refractive index is initially one (air medium).

Find collision in the BVH

Each ray finds the nearest intersected polygon and surface. This stage requires using the constructed BVH as well as solving ray-AABB and ray-triangle intersections. The foundational core is the time-of-flight (ToF) principle, which involves estimating the nearest intersection by timing the travel distance of the laser beam, whose velocity is known. Whether the propagated ray detects an intersection, the indices of the intersected triangle and surface are stored in what is called a collision buffer.

Ray reduction

Then, rays do not operate individually but rather as a group of rays in a pulse. Hence, the number of shader calls is p instead of $n_p p$. This stage helps to simulate multiple returns since rays which are not terminated

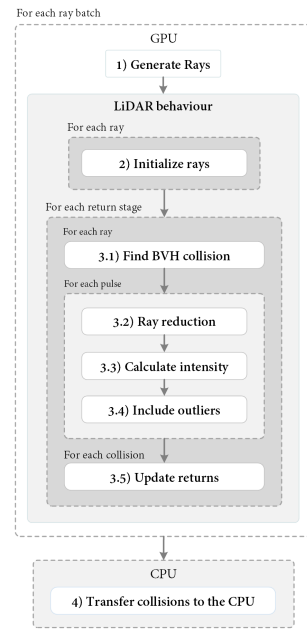


Figure 9.15: Summary of the simulation workflow as implemented in the CPU and GPU. CPU processing is minimized during this pipeline to avoid delays from data transfers.

early are able to penetrate high and low vegetation and acquire ground-labelled points. Although a pulse can detect several, disparate returns, only one is considered valid at each incremental step. Therefore, threads that intersected the nearest surface or did not collide are terminated, whereas the rest continue their path. Several impacts are considered to belong to the same chunk of a surface if multiple criteria are met.

- Collisions fall in the same triangle. This assumption cannot be applied to objects, since triangle meshes represented by several not-enclosed surfaces, such as leaves, may lead to missing some returns.
- Distance among collisions is smaller than $2 - |\hat{n} \cdot \hat{r}_d|$, thus taking into account the surface normal vector. Accordingly, this condition is expressed as shown in Equation 9.21, given that d is the distance from the collision to the LiDAR's receiver component, r is the pulse radius and $\hat{n}$ is the surface normal.

$$d_{max} = 2dr \cdot (2 - |\hat{n} \cdot \hat{r}_d|) \tag{9.21}$$

Measuring the distance solely with $2dr + \epsilon$ leads to omitting those collisions whose $\hat{n} \cdot \hat{r}_d$ highly differs from 1. Consequently, the maximum distance is corrected as shown in Equation 9.21.

- Collided triangles are adjacent and therefore they share at least two vertices. Given the limited radius of a single pulse, checks of one-level depth are sufficient under the majority of scenarios.

Rays change based on the conclusions that were extracted in this section: those that collided with a surface which was not the closest one follow their path, in contrast to those that did not collide or caused the current return. Consequently, laser emissions may iteratively find several returns, and each new intersection is assigned a return number that is later completed with the number of collisions returned per pulse at the end of the simulation.

Maximum range of ToF measurements

Given the maximum distance, R, and a noisy boundary, defined through $R_{\delta_{min}}$ and $R_{\delta_{max}}$, with $R_{\delta_{min}} \leq 0 \leq R_{\delta_{max}}$, a point is discarded if the distance to the sensor, d, satisfies Equation 9.22.

$$d > R + \delta(R_{\delta_{max}} - R_{\delta_{min}}) + R_{\delta_{min}} \tag{9.22}$$

provided that δ is a random value retrieved from a uniform random distribution ranging from 0 to 1, and R_δ denotes a value relative to R, with $R_{\delta_{min}} \leq R_{\delta_{max}}$. For instance, $R_{\delta_{min}} \leftarrow$ -1 m and $R_{\delta_{max}} \leftarrow 1$ m are valid values. Figure 9.16 shows an example of smoother boundaries in return losses.

'Time-walk' effect

This effect is simulated by translating returns according to three factors: the distance to the sensor's receiver and the indices that identify 1) the intersected surface and 2) a single collision in a unique way. Hence, index-related factors, δ_{object} and δ_{return} are intended to distort the returns according to either *object* or *return*. δ_{object} distorts every intersection within a surface

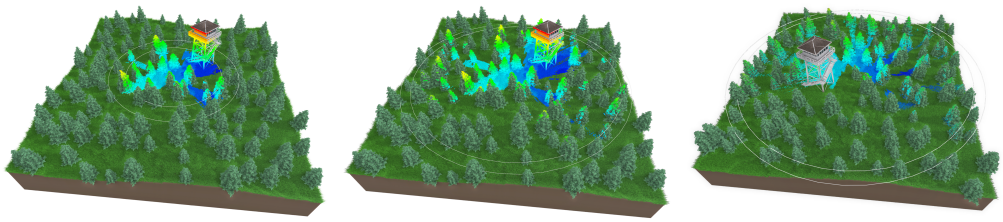


Figure 9.16: Scanning in three environments using an increasing maximum range.

with the same factor, thus preserving the shape, whereas δ_{return} changes for every return. Still, the magnitude of these errors is mainly driven by the distance factor (Equation 9.23).

$$\varrho = e_{glossy} \cdot (d \cdot \hat{r}_d k_{distance} + \hat{r}_d(\delta_{object} k_{object} + \delta_{return} k_{return})) \tag{9.23}$$

where $k_{distance}$, k_{object} and k_{return} are constant values that control the magnitude of every equation term, e_{glossy} is a Boolean value that determines if such error is applied, and d is the distance from the sensor to the intersection. Figure 9.17 shows the translation applied to highly reflective objects if the LiDAR's emitter is close to the affected surfaces.

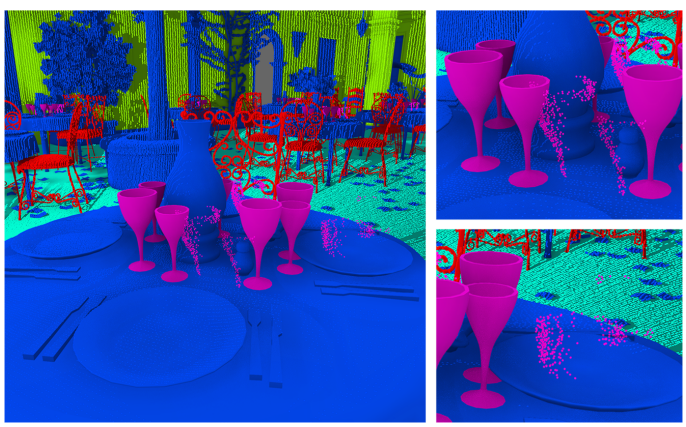


Figure 9.17: 'Time-walk' effect simulated on glass surfaces rendered according to their semantic labels.

Slope derived errors

Points are also distorted on steep surfaces according to the airborne flight altitude (vertical error) and the slope (horizontal error). The former is calculated as $\frac{h}{10^3}$ m, with h being the airborne platform altitude [409]. The altitude-based error depends on the surface slope and flight altitude [410] as expressed in Equation 9.24.

$$\begin{aligned} \varrho = e_{terrain} \cdot (&\delta_h k_{height_h} h\,[\delta_x, 0, \delta_z]^\intercal \\ &+ \delta_v (k_{height_v} h + k_\alpha \alpha)\,[0, 1, 0]^\intercal) \end{aligned} \tag{9.24}$$

provided that h is the flight altitude, k_{height_v}, k_{height_h}, k_α weight the flight altitude and slope angle for the horizontal and vertical errors, and δ are

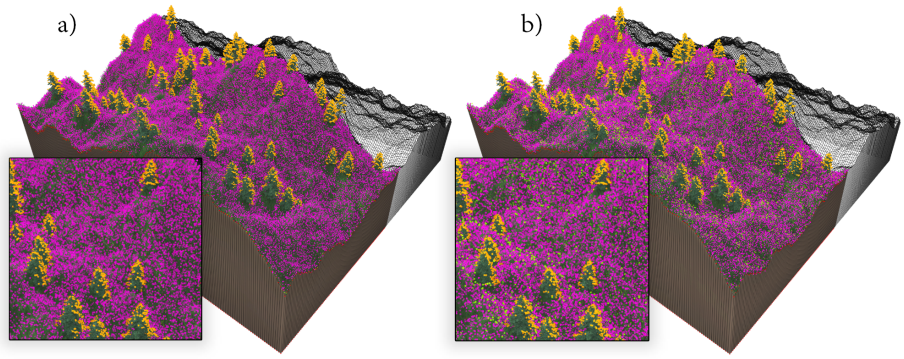


Figure 9.18: Terrain-induced errors for a procedural environment. a) depicts an environment scanned without slope-based errors, whereas b) simulates such errors, thus making these points visible in the foreground.

random values in $[0, 1]$. Figure 9.18 shows the outcome of this distortion. All these errors can be omitted by weighting them as zero.

Bathymetric LiDAR

This kind of sensor presents some relevant changes with respect to the topographic LiDAR. Bathymetric LiDAR is able to detect surfaces underwater and yet propagates causing new returns. The ray's origin is always updated as $r_o = p + \epsilon$, whereas the ray's direction after colliding is computed from the refraction vector for the incident $\hat{r}_d$, the surface normal, $\hat{n}$, and the ratio of refractive indices, r_i. In addition, r_i is given by the refractive index of the substance where the ray propagates, η_1, as well as the index of refraction from the substance of the intersected material, η_2. Moreover, η_2 depends on the wavelength of the LiDAR sensor, and therefore, the material description must include the refractivity over different wavelengths.

Return losses

The return loss over glossy surfaces is driven by an exponential function, $g(k_s)$, the coefficients a, b, c, p, a threshold t and the material's roughness, k_s, as shown in Equation 9.25. Hence, returns from glossy surfaces whose specular factor is above t are discarded whether the value retrieved from a random distribution is smaller than $g(k_s)$. Accordingly, surfaces with lower reflectivity are less likely to cause return losses. Thus, it is possible to partially or entirely discard water returns with a step-function as the one depicted in Figure 9.19.

$$g(k_s) = \begin{cases} c + a(k_s + b)^p & k_s > t \\ c & k_s \leq t \end{cases} \tag{9.25}$$

where c allows introducing some randomness even for diffuse materials. However, c is defined as zero by default.

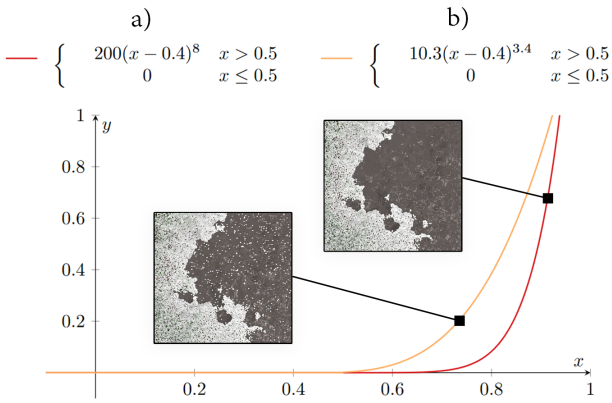


Figure 9.19: a) A loss function that slightly tolerates surfaces as glossy as water, and b) another loss function that discards every return from glossy surfaces.

Intensity computation

This stage simulates radiometric information acquired by the virtual LiDAR, which will be the matter of the following chapter. Note that radiometric data may also be relevant for DL networks and therefore should be accurately modelled. The unitless returned intensity values are equally influenced by the surface material and the LiDAR specifications, including the operating wavelength. The formulae describing the received optical power were already explained in Chapter 2, and therefore, we refer the reader to this section for further insight into the involved parameters.

Simulating outliers

Outlier points are included to simulate errors caused by atmospheric particles such as dust or steam [411]. To this end, a variable amount ($t_{outlier} \in [0, 1]$) of returns is translated along the ray's direction with a magnitude of $t \in [0, 1]$, which is modelled with a pseudo-random generation. In this case, the distribution from which noise is retrieved is particularly relevant to emulate the spatial distribution of outliers. The value t refers to the parametric value in $p = r_o + r_d \cdot t$. Figure 9.20 shows some outliers simulated at the Conference scenario.



Figure 9.20: Outliers in a TLS simulation, with $h \leftarrow 0.95$.

Update of return numbers

Pulses are updated once all the rays are terminated or achieve the maximum number of returns. Returns are then changed to save the overall number of collisions per pulse. The return number provides valuable information since it helps to compute a factor k_τ defined as $\frac{\tau}{n_\tau}$, i.e., the return number, τ, divided by the overall number of returns. Returns obtained in later iterations are more likely to belong to the ground; however, filtering by τ does not help to retrieve every ground point and filter out the rest. Instead, the last return of every pulse satisfies that $k_\tau > 1 - \epsilon$, and these are more appropriate to filter out vegetation and render only points close to the ground. For example, LiDAR technology has been widely adopted

in archaeology to detect remains which are not visible (either buried or occluded by the canopy) using the DEM reconstructed with points that belong to the ground.

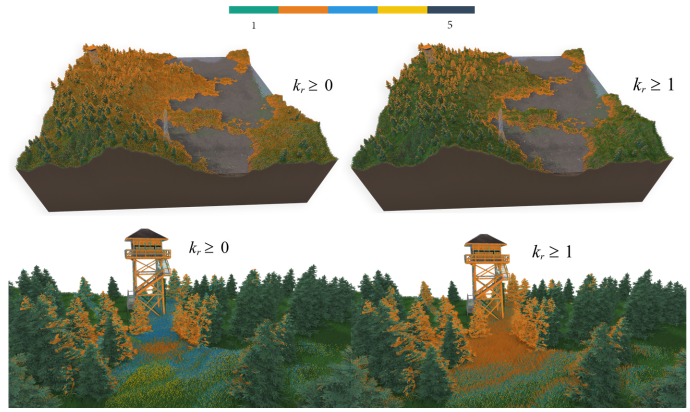


Figure 9.21: ALS and TLS simulations over a procedural environment. First, the complete point cloud is rendered ($k_r \geq 0$), whereas the second image renders only the last return of each pulse.

Figure 9.21 depicts two different point clouds filtered by means of k_τ to visualize the last return of each pulse. The forest canopy was significantly dense and many pulses did not penetrate vegetation, whereas low vegetation was easily avoided. Accordingly, it shows a much less dense point cloud at ground level when $k_\tau \geq 1$ by filtering out vegetation.

9.6 Results and discussion

The proposed LiDAR simulator has been tested against static and procedural scenes with high polygonal complexity (from 4.5M triangles to 11M). Also, the performance was checked by considering both response time and resulting point clouds. Therefore, the GPU- and CPU-based approaches were compared. Then, the proposed simulation was compared against several state-of-the-art LiDAR solutions by addressing both quantitative metrics, such as the facilities to generate dense semantic datasets, and qualitative factors, such as the similarity of the point clouds with respect to real-world LiDAR scans.

Measurements were performed using a PC with Intel Core i9-9900 3.1 GHz, 48 GB RAM, RTX 2080 Ti GPU with 11 GB RAM (Turing architecture). The proposed methodology was implemented in C++ along with OpenGL. Massively parallel processes were either implemented in the GPU, using GLSL and general-purpose compute shaders, or in the CPU with OpenMP.

9.6.1 LiDAR response time

A comparison of the GPU-based solution against its analogous multi-core CPU approach will be first carried out to show the efficiency of the proposed simulation. Synchronism was established among the pipeline stages to await the end of different GPU calls, thus allowing us to measure the stage-wise latency in the GPU. Note that the response time from ray instancing is

shared for the same LiDAR configuration, whereas the rest of the stages are very influenced by the scenario. Figure 9.22 summarizes the results obtained for TLS and ALS surveys, using scenes with an increasing number of polygons: Suburb (4.1M), San Miguel (4.5M), City (10.1M) and Forest (11M). Simulations with multiple returns were bounded to five returns. Further insight into the results is provided in Table 9.4.

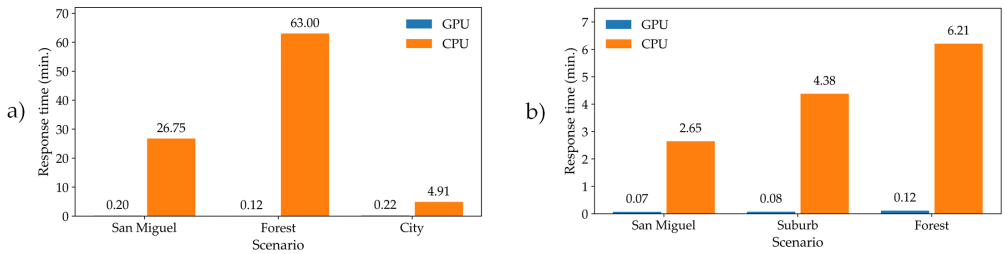


Figure 9.22: Comparison of response time between the GPU and multi-core CPU approaches using a) TLS and b) ALS with 10M pulses, 10 rays per pulse and multiple returns.

Ray build time. TLS beams were straightforwardly computed, and thus the instancing phase shows lower latency. On the other hand, ALS beams require a larger number of parameters and calculations involving angles and the platform path. The response time for 10M pulses grows linearly with respect to 5M. Despite CPU-based ray-instancing not being as time-consuming as the LiDAR core, the speedup ranged from 90.9% (ALS, 10M pulses) to 99.12% (TLS, 10M pulses).

LiDAR behaviour. The ToF solver operates for every batch of rays, with the batch dimensionality being limited by the maximum allocatable memory for an SSBO. In this stage, the response time is also proved to grow linearly according to the number of pulses. However, ALS surveys show lower latency both in CPU and GPU since the scene geometry is less dense in the vertical axis, and thus, the BVH clusters are rapidly discarded. Although SSBOs are very limited in size, reducing it beyond the limit queried from OpenGL helped to reduce the latency, apparently from a lower GPU overload. This factor was also parameterized to be tuned during these tests.

Although TLS scans are notably slower, the reported speedup with respect to the sequential approach is impressive: San Miguel (99.27%), Forest (99.81%) and City (95.06%). ALS scans, on the other hand, achieve the following speedups: San Miguel (98.62%), Forest (98.71%) and Suburb (99.15%).

Stage occupancy. Besides the overall response time, the individual latency of every stage was also measured. To this end, stages were explicitly synchronized in the CPU instead of allowing OpenGL to handle parallelism and memory access. Figure 9.23 shows the relative latency for high-performance tests using environments from lower (5M) to higher complexity (10M) scanned with 5M pulses. Apart from the phases depicted in Figure 9.15, reading transfers from GPU to CPU were also included as they were required to perform further processing in the CPU and store the results as part of the simulated point cloud. Instead of constructing the

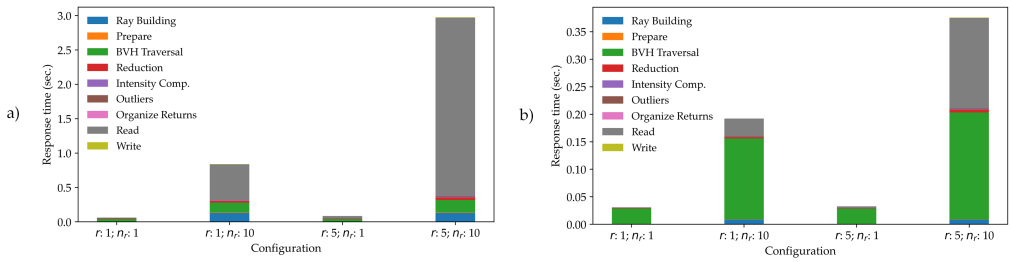


Figure 9.23: Absolute (left) and relative (right) response time of LiDAR stages. a) The first chart row compares the performance of TLS in a procedural forest, using a different number of allowed returns (r) and rays within a pulse (n_r). Similarly to TLS, the second chart row presents the results for an ALS sensor.

whole ray buffer at once, rays were iteratively constructed to avoid CPU and GPU data transfers.

As reported, transferring collisions to the CPU takes a substantial portion of time, whereas BVH traversals are notably more time-consuming for highly populated environments. However, traversing a scene is significantly slower for TLS scans as they cannot discard clusters as rapidly. The reduction stage is also relevant in this chart due to the pulse-wise operations; note that in this stage, every ray within a pulse is processed by the same thread.

9.6.2 Generation of synthetic datasets

Another main objective of this work is to provide a framework for generating large semantic datasets. Thus, several tests were conducted to evaluate both the dimensionality and the number of labels in the collected datasets. Comparisons were established with previously cited work [246, 252, 254, 268, 412], either based on real or synthetic datasets, by taking into account the number of scans, the overall number of points, as well as the number of semantic annotations. To provide a fair evaluation, the tests were performed using the configuration of a Velodyne HDL-64E [271], as proposed in most of the urban datasets in the literature [246, 254, 268]. Aerial surveys are, on the other hand, not frequent in previous work. Accordingly, other simulations have been carried out using the configuration of an airborne DJI Zenmuse L1 sensor [413]. Remark that some of the sensor parameters were not provided by the manufacturers, and therefore were adjusted to generate dense point clouds (Table 9.6). Also, the variable height of urban environments required solving the surveys with multiple scans at different altitudes.

The results were collected by scanning two virtual urban scenes. The first has 32 semantic classes, whereas the second environment presents 45 semantic annotations. Furthermore, these scenes were populated with procedural pedestrians and vehicles. After scanning both environments, an average frame size of nearly 623k points was reported from 488 scans, either from TLS or ALS, which greatly improves the current average size of synthetic LiDAR simulators. Despite real-world datasets acquiring more

dense point clouds, e.g., Semantic3D [255], the main contribution of virtual scans is their low response time and efficiency both in acquiring data and annotating each point. Furthermore, the conducted tests were only performed over procedural environments generated with a single seed. Otherwise, a large number of differently populated environments could be modelled to further collect synthetic LiDAR datasets.

Figure 9.25 illustrates the semantic labels of the input scenarios as well as the point clouds collected from TLS and ALS scans, coloured with their height. The number of points annotated with different categories is depicted in Figure 9.24, by reporting points returned from TLS and the fusion of both TLS and ALS (Table 9.5). Despite the contribution of ALS scans being relevant for augmenting the dataset labels, the weight on the bar chart is very reduced in comparison with TLS scans due to the lower number of pulses.

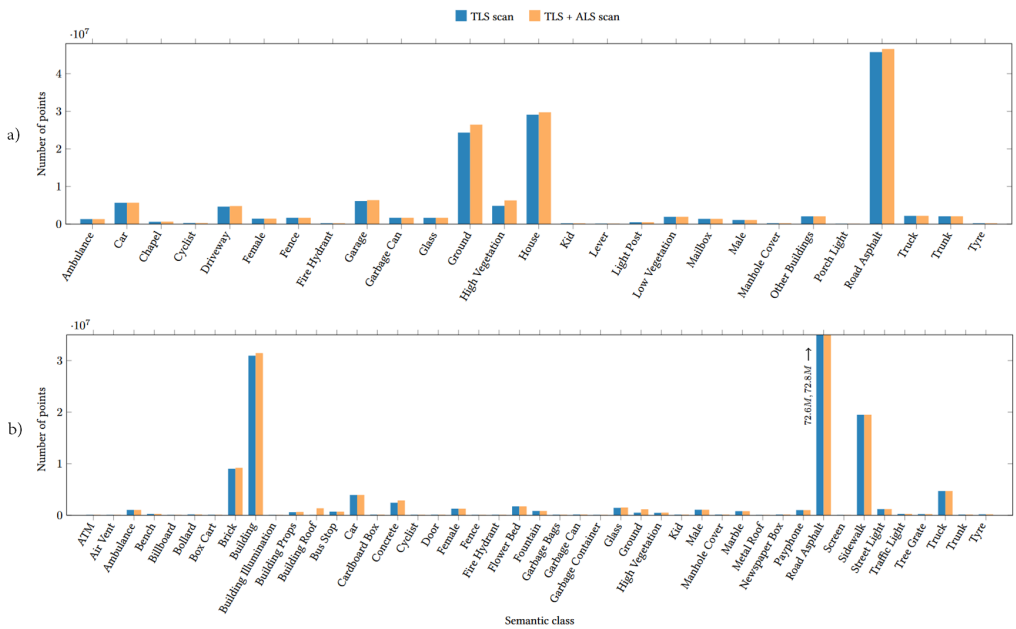


Figure 9.24: Number of LiDAR points per semantic label in two different urban environments. The first bar reports the results of TLS scans, whereas the second bar shows the results combining both ALS and TLS. a) Distribution of 139M and 145M points for TLS and TLS + ALS scans, respectively, whereas b) shows the profile generated by 165M and 170M points.

9.6.3 Visual results

Besides quantitative tests, comparisons were also established by means of visual results using real-world LiDAR point clouds as the baseline result. Furthermore, these tests were carried out over CAD models instead of polygonal meshes extracted from previous scans, as the latter introduces geometrical errors. Vehicles were selected as the target since their CAD models were easier to collect and they are also very frequently scanned in publicly available TLS urban datasets. The compared dataset was obtained

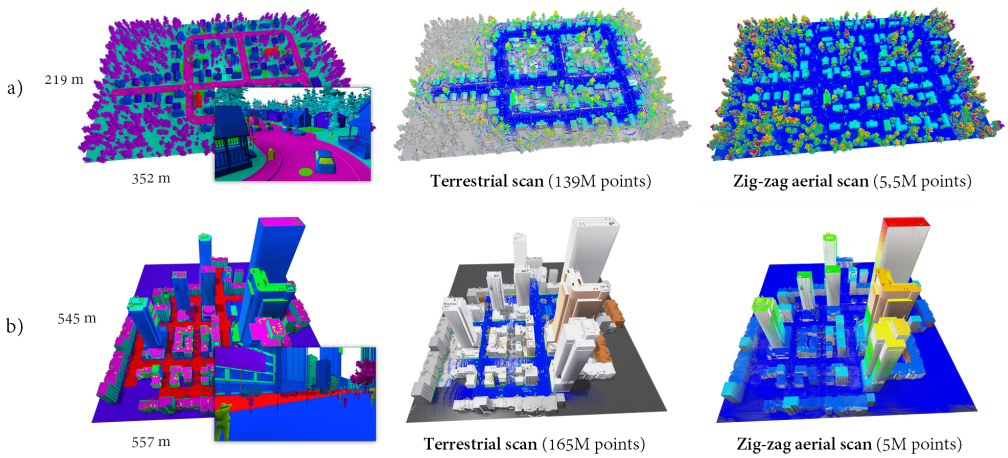


Figure 9.25: Point clouds simulated over urban environments. The input triangle meshes are first depicted, and then, environments are scanned either with TLS or ALS.

from a Pandar64 sensor [407] that provided RGB images and individual TLS scans. This way, individual scans were compared instead of point clouds obtained from several viewpoints.

Regarding other virtual simulators, most of the revised work provides datasets rather than the simulator itself, operates only in pre-defined environments [242], or does not integrate surface materials [236, 238, 268, 271, 412]. Others lack non-uniform scanning [244] and noise/loss behaviour [414]. Consequently, comparisons have been performed against the widespread blender plugin Blensor [257]: it is an open-source project that also implements a naïve simulation of noise and errors derived from reflective surfaces. The scenario was configured so that some car materials were modelled as highly reflective (e.g., glass), whereas the LiDAR location was configured according to the real LiDAR placement. In addition, the Pandar64 LiDAR presents different spatial resolutions along the vertical axis. In contrast to previous work, the proposed solution also supports this kind of sensor. Finally, a loss function was defined for glossy surfaces, over which reflection errors were emulated. Nevertheless, note that the effects derived from 'time-walk' are harder to notice due to the reduced distance between the sensor and the target surfaces.

Figure 9.26 shows that glossy surfaces are omitted by Blensor, whereas the point clouds obtained from our simulator have some gaps as a result of the loss function. On the other hand, Blensor includes a noise function defined by its amplitude and distribution to include some variability in the final point cloud. Consequently, both virtual LiDAR sensors generate results following patterns similar to the real LiDAR, including noisy point clouds; however, reflection errors were better emulated by the proposed solution in a lower response time.

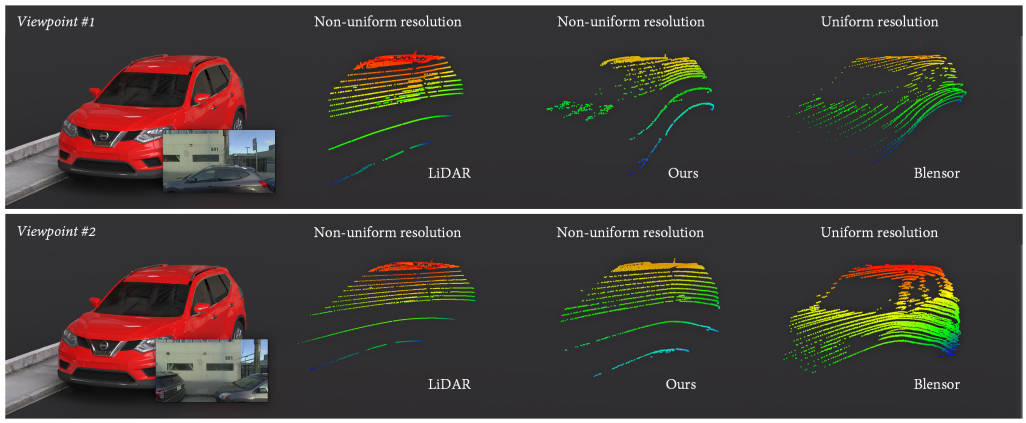


Figure 9.26: Comparison of real and synthetic point clouds. The first two images depict the input CAD model and its corresponding LiDAR point cloud as captured from a Pandar64 sensor [407]. The third image illustrates a synthetic point cloud from our simulator, and finally, the fourth image shows the result from the Blensor plugin with a uniform vertical resolution.

Table 9.4: Comparison of CPU and GPU response time for TLS and ALS scans. The results were obtained using static and procedural scenes with a significant number of triangles.

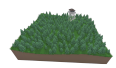
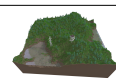
Terrestrial LiDAR								
Configuration				 San Miguel #triangles 5.6M		 Forest #triangles 11M		City #triangles 10.1M
n_p	Method	n_r	Ray inst.	ToF Solver		ToF Solver		ToF Solver
Single return								
5M	**GPU**	10	**0.141s**	**2.747s**		**0.755s**		**3.664s**
	CPU		15.601s	193.790s		1,854.337		128.900s
10M	**GPU**	10	**0,279s**	**5.254s**		**1.510s**		**7.745s**
	CPU		31.740s	1,571.758s		3,743.797s		261.091s
Multiple returns (5)								
5M	**GPU**	10	**0,141s**	**5.637s**		**3.323s**		**6.597s**
	CPU		15,601s	194,533s		1.900,348s		129,725s
10M	**GPU**	10	**0.279s**	**11.481s**		**6.906s**		**12.949s**
	CPU		31.740s	1,573.349s		3,748.547s		262.640s
Aerial LiDAR								
Configuration				 San Miguel #triangles 5.6M		 Forest #triangles 11M		Suburb #triangles 4.1M
n_p	Method	n_r	Ray inst.	ToF Solver		ToF Solver		ToF Solver
Multiple returns (5)								
5M	**GPU**	10	**1.240s**	**0.923s**		**2.348s**		**0.995s**
	CPU		12.586s	66.889s		102.070s		119.946s
10M	**GPU**	10	**2.510s**	**1.806s**		**4.4294s**		**1.996s**
	CPU		27.583s	131.253s		344.882s		235.174s

Table 9.5: Overview of outdoor LiDAR datasets with semantic annotations, regarding their average size (*Points*/*Scan*) and number of labels.

Dataset	Annotation	Number of scans	Average size	Labels	Synthetic
SynLiDAR [268]	Point-wise	198,396	98.197k points	32	✓
GTA-LiDAR [412]	Pixel-wise	121,087	-	2	✓
Semantic3D [255]	Point-wise	30	**133.3M** points	8	✗
SemanticKITTI [246]	Point-wise	43,552	105.79k points	25	✗
SemanticPOSS [252]	Point-wise	2,988	72.289k points	14	✗
nuScenes [254]	Point-wise	40,000	35k points	32	✗
Ours	Point-wise	488	622.95k points	**53**	✓

Table 9.6: Specifications of sensors simulated during the scanning of urban environments.

Attributes	Sensors			
	Velodyne HDL-64E [271]		**Zenmuse L1** [413]	
	Environment 1	Environment 2	Environment 1	Environment 2
Field of view	360°× 26.9°		70.4°× 4.5°	
Resolution	0.08°× 0.4°		0.1437°× 0.00918°	
Number of channels	64		1	
Vertical FOV origin	-11.45°		0°	
Maximum range	120 m		190 m	
Maximum number of returns	2		3	
Scan altitude* (h_l)	2 m		45 m	115 m / 290 m
Platform speed* (s_p)	–		2 m/s	0.5 m/s
Rays within a pulse* (n_r)	5		5	
Pulses per scan*	302,625		240,000	
Generated rays	342M	30M	390M	72M
Resulting points	139M	5.5M	165M	5M

* These parameters were not provided by the manufacturer and thus were adjusted with the objective of generating a dense point cloud.

9.7 Conclusions and future work

A massively parallel and parameterized LiDAR was proposed for generating dense semantic point clouds over any environment defined as a triangle mesh, aimed at collecting large datasets for DL. However, procedural scenes were constructed to provide a time-efficient alternative that alleviates the manual annotation task. Procedural environments are labelled once and produce large datasets, whereas static environments only contribute to building a few point clouds at most. Spatial queries were rapidly solved with state-of-the-art data structures proposed for ray tracing, whereas the ToF solver includes multiple returns and well-known LiDAR errors. To this end, LiDAR's pulses were discretized as several rays that strike into the scene surfaces, instead of solving the simulation in the image space. Thus, the described algorithm is suitable both for real-time simulations and the generation of large LiDAR semantic point clouds with low latency. Accordingly, LiDAR scans can be solved iteratively or in a single batch. Also, the sensor parameterization helps in simulating a wide range of LiDAR sensors and platforms, including ALS.

This solution was assessed through its performance and capabilities in the generation of large datasets. The workflow was implemented in GLSL and compared with its analogous multi-core CPU approach by conducting much more dense scans than actual LiDAR sensors. These tests showed that the GPU approach improves the sequential procedure with speedups above 90%. Hence, it was demonstrated that this simulator is appropriate for generating large semantic LiDAR datasets with low latency from highly detailed scenes. Regarding scanning capabilities, a significant enhancement both in average scan size and the number of semantic labels was observed. Note that this is not solely affected by the LiDAR scans, but also as a result of the facilities given on the annotation task. Finally, point clouds were visually compared against the results obtained using the Blensor plugin, proving that some errors, such as those derived from highly reflective surfaces, were better simulated by our solution.

In future work, the proposed pipeline could be further accelerated in multi-GPU systems. Furthermore, this framework could be extended to cover full-waveform and single-photon LiDAR [415] and other nonincluded systems. Finally, a deeper study ought to be conducted to show the benefits of synthetic LiDAR point clouds in the training of DL networks, thus checking whether datasets can be partially or completely exchanged by synthetic data.

Estimating radiometric data from synthetic LiDAR

10

About this chapter

This chapter estimates intensity data over the previously explained LiDAR simulator. Semantic labels have been previously mentioned as additional data that provide further insight into the observed materials. However, there are other properties which are estimated during flight and also help to characterize materials, such as the return intensity. Hence, surfaces can be modelled with BRDFs to estimate intensity, multiple returns, return losses, etc. At least two approaches exist for modelling BRDFs. The trivial one is to consider the traditional Computer Graphics BRDFs, whereas a more up-to-date solution consists of utilizing BRDF databases measured from real-world surfaces with a gonio-photometer. Another alternative is to transfer LiDAR data into DL models to estimate the intensity from the observed depth. However, the latter approach requires solving the LiDAR simulation in the image space. Other approaches are based on the voxelization of point clouds, thus losing part of the details of the scene, or using, for instance, graph-based representations for point clouds. Nevertheless, DL-based alternatives are out of the scope of this dissertation and thus only the first two explained approaches will be evaluated in this section.

Comparisons with previous work are not trivial to establish, nor against real-world scans, since the latter requires having the model of a real-world scene that should be scanned with real and synthetic LiDAR. As a consequence, the discussion of this chapter is merely focused on the modelling of scenarios with distinct materials. This way, the intensity histograms can be evaluated to check whether both approaches achieve similar results as well as the differences between TLS and ALS. For instance, the intensity distribution is expected to be correlated to the viewing angle and thus on the kind of scan, either terrestrial or airborne. Similarly, some materials are expected to return higher intensity values, in comparison with others with a dull appearance.

This chapter explains the two mentioned approaches for estimating LiDAR intensity, based on the work of Chapter 9. Both approaches, together with the required parameters, are depicted in Figure 10.1.

10.1 On the estimation of LiDAR intensity 191
10.2 Material modelling . . 193
10.3 Analytical BRDFs . . . 194
10.4 BRDF database 197
10.5 Results and discussion 198
10.5.1 Analytical BRDFs 198
10.5.2 BRDF database 199
10.6 Conclusions and future work 200

10.1 On the estimation of LiDAR intensity

The most straightforward approach for simulating LiDAR intensity is to use traditional shading functions, also called analytical BRDFs. The number of possible BRDFs is very limited, or at least, these are not enough to emulate any real-world material. Furthermore, these are typically constrained to estimating red, green and blue data with rendering-based material properties,

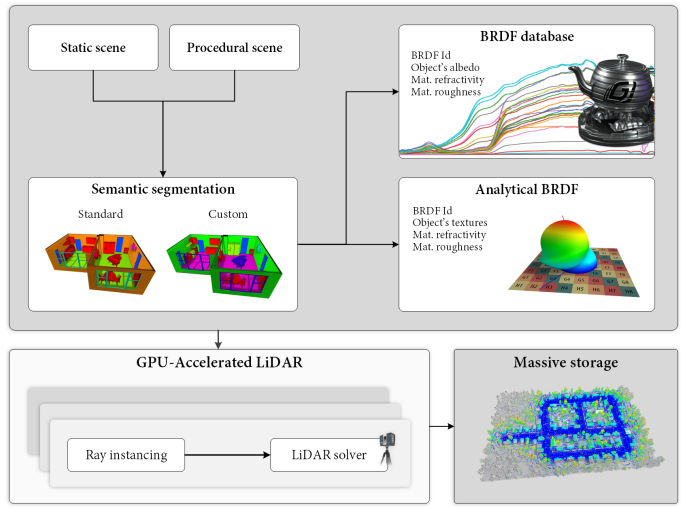


Figure 10.1: Overview of the proposed LiDAR simulation. First, the input scenarios are attached to semantic annotations and materials. Then, the previously explained LiDAR simulator solves the scan and the resulting point cloud is stored in a standard file format.

including the albedo colour and specular factors. Other BRDFs have a larger number of configurable parameters such as the material roughness and reflectivity. However, it is not trivial to determine which BRDF is the most similar to emulate one specific material in a synthetic scenario. For instance, both Oren-Nayar and Lambertian BRDFs have a dull appearance.

More recently, Dupuy and Jakob [95] published a database of spectral signatures from isotropic and anisotropic materials, comprising the interval ranging from 358.628 nm to 1001.89 nm with a sampling frequency of ~3.341 nm. On the other hand, previous databases were limited to red, blue, and green wavelengths as they were designed for rendering applications [100]. From these very first databases, other studies were able to parameterize the collected materials to build a larger database [101]. However, RGB databases are not appropriate for simulating most of nowadays sensors as these may operate in a wavelength interval not included in the RGB spectra. For instance, infrared devices cannot be simulated with RGB data. Still, state-of-the-art hyperspectral databases cannot help to simulate every kind of LiDAR. A frequent wavelength in LiDAR technology is 1550 nm, which is not covered by the signatures collected by Dupuy and Jakob [95]. Hence, BRDF databases are improving by covering wider spectra, including the infrared, though are still not sufficient for other kinds of sensors such as LiDAR. Still, this approach is more physically accurate than the first one, despite being limited to wavelengths below 1000 nm.

A brief insight into the literature on the simulation of radiometric data shows the confusion between ray-tracing and ray-casting concepts. To the best of our knowledge, none of the previous LiDAR simulators work as a ray-tracing solver. Instead, rays are cast and the collision intensity is estimated [93, 258, 259]. However, this fusion of ToF and radiometry estimations has led previous work to erroneously refer to this as ray-tracing. Besides this, the operating wavelength has been barely taken into account in the simulation [241, 257, 263]. Other works use widespread BRDFs such

as Oren-Nayar, Lambert and Blinn-Phong [263]. More physically-based approaches use material properties such as diffuse, specular and transmissive factors provided by third-party vendors [260]. Finally, other state-of-the-art methods use images co-acquired with LiDAR information to estimate intensity [268, 270]. These are typically affected by environmental conditions such as lighting and neither is guaranteed that the material database is large enough. However, the former drawback can be tackled using libraries that help to estimate several environmental and lighting scenarios from a single image [416]. Similarly, sensor defects and limitations could be simulated by training a DL network [269].

10.2 Material modelling

Static and procedural models are linked to semantic annotations and materials that help to calculate the LiDAR intensity. As previously proposed, materials and any other kind of data can be attached to models following a hierarchical search based on the object's name and its material name. The latter is used if the object's name is not unique, i.e., it collides with the name of other objects. Rather than defining the material of every surface from scratch, a material database is provided to facilitate the modelling of surfaces, covering some widespread materials such as stone, fabric, plastic, iron, etc. Each one of these materials is parameterized with at least three properties, as can be observed in Figure 10.2. Using the BRDFs collected by a gonio-photometer fastens the calculations as it already provides the reflectance, whereas the shading BRDFs use the albedo and specular colours, among other factors, as part of the formulae. Still, the values obtained from the BRDF database are scaled according to the object's albedo. Also, the refractivity (η) is required to account for changes in the traversed medium, whereas the roughness ($\alpha \in [0, 1]$) is utilized for modelling return losses. The refractivity is not provided as a value, but rather as the Id of a material in a database of refractive indices.

The refractive indices were obtained from publicly available signatures collected from a large number of peer-reviewed works [417]. In comparison with the sampled BRDFs, refractive indices are provided as sparse values in a variable wavelength range (it varies from one work to another). Therefore,

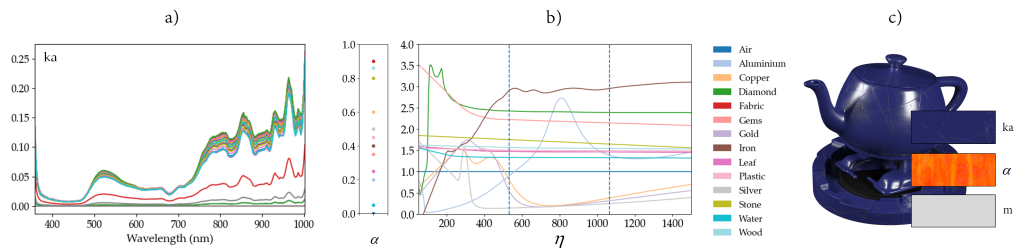


Figure 10.2: Modelling of scene surfaces to estimate the returned intensity. a) Reflectance from a BRDF parameterized with the input and output vector (w_i and w_o) [95] and b) model-agnostic material specifications, including refractivity and roughness exponent. Finally, c) previous material properties are scaled according to the model textures (albedo, metallic and roughness).

the refractivity signatures are first used to construct a Catmull-Rom spline which can be later sampled for the LiDAR wavelength. This property is especially relevant for bathymetric LiDAR since the refracted vector is computed according to the refractivity of two different media.

BRDFs help to estimate the ratio between incoming and outcoming radiance for a specific wavelength by considering the incidence angle of LiDAR beams, regardless of using analytical BRDFs or collected BRDFs. In this stage, the emitter and the receiver are assumed to have the same location, and therefore, incident ($\vec{w}_i$) and reflected ($\vec{w}_o$) vectors follow the same path. The latter assumption helps to considerably reduce the memory footprint of the second approach in the GPU. Omitting this would lead to having n^2 samples instead of n, with n being the sampling resolution of the collected signatures in a hemisphere.

10.3 Analytical BRDFs

Analytical BRDFs require some properties that are typically provided as PBR textures. However, transferring these textures during the simulation either implies solving this piece-wise or building a texture atlas, with the latter having a very limited dimensionality in OpenGL. Instead, the albedo, normal, roughness and metallic texture were sampled per vertex before solving the simulations and subsequently stored together with the vertex data. The object-wise texture sampling was efficiently solved using OpenGL's compute shaders.

The returned reflectance was part of the LiDAR equations presented in Chapter 2, whereas the overall formula helps to account for other atmospheric and viewing conditions. The focus of this chapter is simply on the returned reflectance, although the intensity reported in later experiments corresponds to the formulae explained in Chapter 2. The first approach is to estimate the reflectance with analytical functions, $f_r(\vec{w})$, that calculate the amount of light received back to the scanner, according to the incoming energy. In this manner, a BRDF can be used for each type of surface, or so to

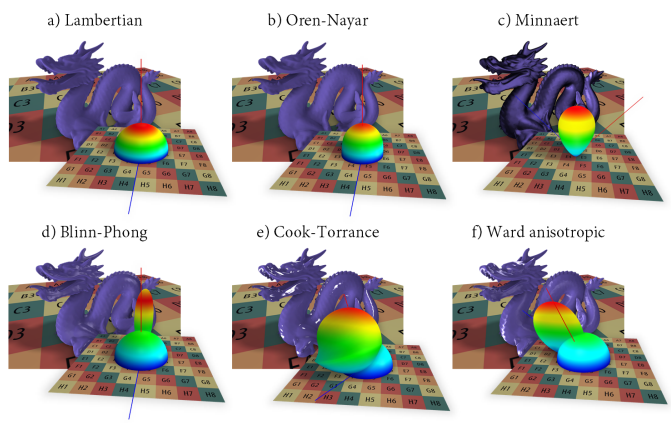


Figure 10.3: BRDFs depicted 1) by shading a 3D model and 2) by distorting a semi-sphere with the estimated intensity. $f_r(\vec{w})$ is represented by the distance from each vertex of the distorted semi-sphere to the origin, [0, 0, 0]. From left to right: a) Lambertian, b) Oren-Nayar with α_m = 0.5, c) Minnaert with a darkening factor (k) of 1.47, whereas ρ_d is increased by a factor of A = 2.6, d) Blinn-Phong with α = 11, e) Cook-Torrance with α = 0.685 and F_0 = 0.40, and f) Ward anisotropic with α_x = 0.15 and α_y = 0.75. ρ_d = 1 for all the illustrations.

speak, for each material. In order to cover most of the materials represented in our scenarios, six different BRDFs, depicted in Figure 10.3, were included. The rendering of the Dragon model shows the aftermath of sampling BRDFs with many viewing (outgoing) and normal vectors and a single incoming vector from the light. As in the LiDAR formulae, the calculated intensity is conditioned by, at least, the dot product between the incoming vector and the surface normal. On the other hand, the hemisphere is discretized with a variable resolution, and its vertices were scaled according to the reflectance sampled for a single incoming and outgoing vector, conditioned by the vertex's normal vector. Accordingly, the Lambertian BRDF spreads the same amount of energy in every direction regardless of the surface normal. For the latter representations, reflectance is typically normalized with $1/\pi$ to account for energy spreading.

However, these BRDFs can be simplified whether incident and reflected vectors are assumed to be equal, $w_i = w_o$. That is, the outgoing LiDAR beam and the incoming energy, which reaches the sensor's receiver, share the same path. The subsequent formulae show the backscattering form of the BRDF models depicted in Figure 10.3 and described in [96, 103], where ρ_d and ρ_s are diffuse and specular colors of the material, and h is the halfway vector ($\hat{l} + \hat{v} = 2w_o$, where $\hat{l}$ and $\hat{v}$ are light and view vectors, respectively).

- Lambertian:

$$f_r(\vec{w}) = \frac{\rho_d}{\pi} \tag{10.1}$$

- Oren-Nayar:

$$\begin{aligned} f_r(\vec{w}) &= \frac{\rho_d}{\pi}(A + B)\sin\theta_w \tan\theta_w \\ A &= 1 - 0.5\frac{\alpha_m^2}{{\alpha_m}^2 + 0.33} \\ B &= 0.45\frac{\alpha_m^2}{{\alpha_m}^2 + 0.09} \end{aligned} \tag{10.2}$$

- Minnaert:

$$f_r(\vec{w}) = \frac{\rho_d}{\pi}(\hat{n} \cdot \hat{w})^{2(k-1)} \tag{10.3}$$

- Blinn-Phong:

$$f_r(\vec{w}) = \rho_s(\hat{n} \cdot \hat{h})^k = \rho_s(\hat{n} \cdot 2\hat{w})^k \tag{10.4}$$

- Cook-Torrance:

$$
\begin{aligned}
f_r(\vec{w}) &= \frac{F(\beta)D(h)G(\vec{w})}{\pi(\hat{n}\cdot\hat{w})^2} \\
F(\beta) &= F_0 + (1-F_0)(1-\hat{n}\cdot\hat{w})^5 \\
D(h) &= \frac{1}{\alpha_m^2(\hat{h}\cdot\hat{n})^4}\exp\frac{(\hat{h}\cdot\hat{n})^2-1}{\alpha_m^2(\hat{h}\cdot\hat{n})^4} \\
G(w) &= \min\left(1, \frac{4(\hat{n}\cdot\hat{w})^2}{\hat{h}\cdot\hat{w}}\right)
\end{aligned}
\tag{10.5}
$$

  where $F(\beta)$ is the Schlick approximation [120], $D(h)$ is the Beckmann distribution [96] and $G(h)$ is the geometric attenuation factor. F_0 is the external reflection under normal incidence casuistic ($\vec{w} = \vec{n}$), whereas the rest of the values are approximated through interpolation between F_0 and 1.
- Ward anisotropic:

$$
f_r(\vec{w}) = \frac{\rho_s \exp\left(-\frac{\left(\frac{h\cdot x}{\alpha_x}\right)^2+\left(\frac{h\cdot y}{\alpha_y}\right)^2}{(\hat{h}\cdot\hat{n})^2}\right)}{4\pi\alpha_x\alpha_y(\hat{n}\cdot\hat{w})}
\tag{10.6}
$$

An example of intensity estimation with analytical BRDFs is shown in Figure 10.4, with the vehicle path highlighted with yellow lines.

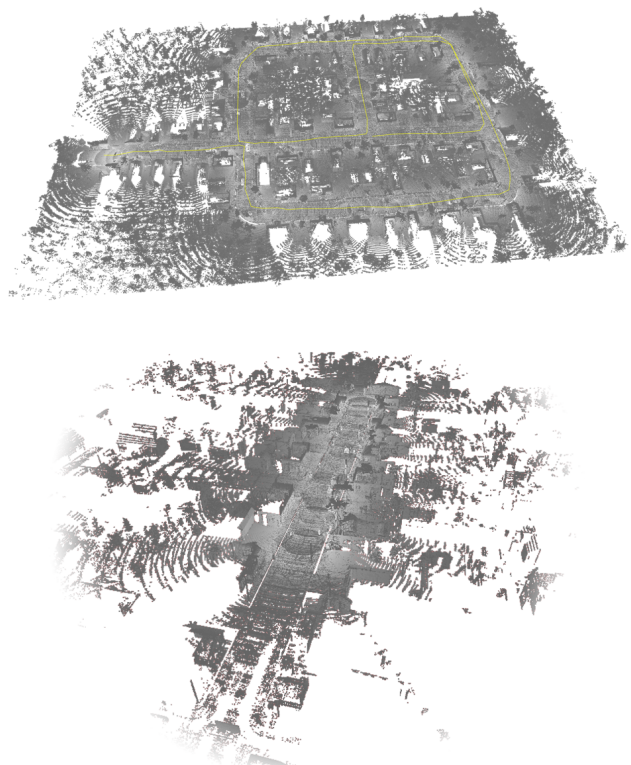


Figure 10.4: Intensity simulated using the analytical BRDFs in the scanning of an urban environment.

10.4 BRDF database

The main concern when using databases collecting BRDF signatures is to store the discretized representation and to sample it from the LiDAR simulation. From the database provided by Dupuy and Jakob [95], a single hemisphere sampling was discretized with $360 \times 90 \times 195$ values from spatial and spectral dimensions. These values were queried from the interface published together with the collected materials, which evaluates the database with the incoming and outgoing vectors, w_i and w_o, albeit both being equal in our solution. Also, the LiDAR wavelength must be linked to the index of the most similar wavelength in the database. Before the simulation, every material is subsampled to store $x \times y$ samples instead of $x \times y \times w$, with x and y are the length of azimuth and elevation discretizations, respectively, and w referring to the spectral dimensionality. Examples of sampled signatures are depicted in Figure 10.6, using random w_i and w_o vectors.

During the simulation, the data transferred to the GPU was queried using $-\widehat{r_d}$, where r_d is the ray direction. From here, the azimuth and elevation were calculated as shown in Equation 10.7. The reflectance was then accessed considering that data was stored in elevation-major order, with an index calculated as $m_{Id} \cdot 360 \cdot 90 + \phi \cdot 90 + \theta$, with m_{Id} being the Id of the intersected material.

$$\begin{aligned} \phi &= \lfloor \left| \arctan\left(-\widehat{r}_{d_y} / -\widehat{r}_{d_x}\right) \right| \cdot 180 \cdot \frac{1}{\pi} \rfloor \\ \theta &= \lfloor \arccos\left(-\widehat{r}_{d_z}\right) \cdot 180 \cdot \frac{1}{\pi} \rfloor \end{aligned} \tag{10.7}$$

Then, the obtained reflectance was integrated into the formulae of LiDAR. The starting spectral signatures, depicted in Figure 10.6, are weighted according to the dot product between the incident vector and the normal vector. An example of the intensity estimated during the scanning of an urban environment is illustrated in Figure 10.5.

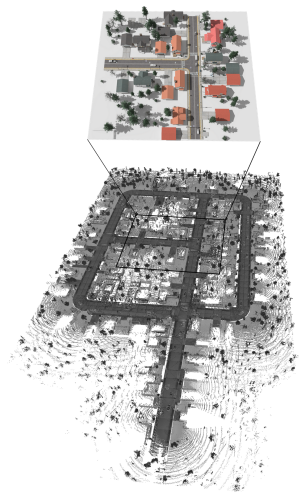


Figure 10.5: LiDAR intensity estimated over an urban scenario using the BRDFs collected with a goniphotometer.

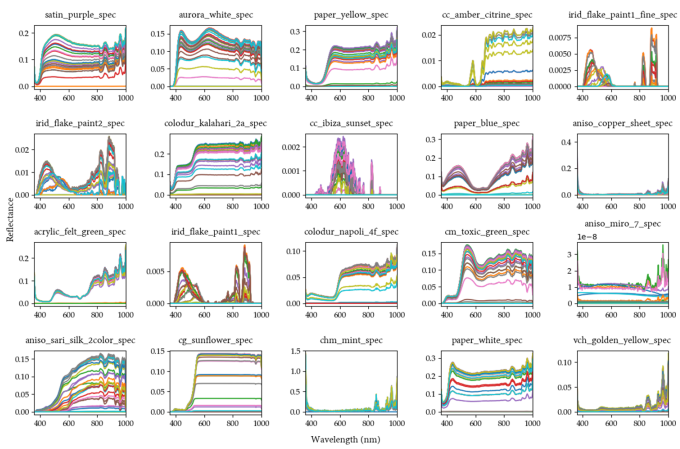


Figure 10.6: Signatures sampled from the material database published by Dupuy and Jakob [95]. Each cell represents a different material, and the depicted signatures are different samples for randomized input and output vectors.

10.5 Results and discussion

The proposed simulation was evaluated using the specifications of commercial LiDAR sensors. Procedural scenes with a complexity ranging from 5M triangles to 11M were used as the input. The generated point clouds will be following analyzed to provide better insight into the estimation of intensity by changing the scene materials. During these experiments, the LiDAR sensors were configured to return a single point per beam, whereas the rest of the properties were set according to the datasheets from Velodyne HDL-64E (TLS) and DJI Zenmuse L1 (ALS) sensors. Still, the resolution and the vehicle speed were adjusted to have a considerable number of points. Finally, LiDAR pulses were discretized using ten rays in every experiment.

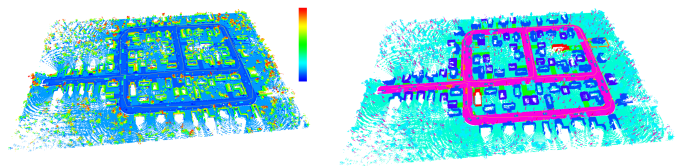


Figure 10.7: Point clouds obtained with a TLS simulation over an urban scenario. The first is coloured according to the relative height, and the second is shaded with an unique color linked to the semantic label.

10.5.1 Analytical BRDFs

The simulation of distinct surfaces is the main objective pursued by the proposed intensity estimation. Three different experiments were conducted to evaluate the outcome of analytical BRDFs. First, surfaces were modelled with the most appropriate material available in the database. According to this, the following materials and BRDFs were linked: `Aluminium` ← `Cook-Torrance`, `Glass` ← `Blinn-Phong`, `Fabric` ← `Minnaert`, `Iron` ← `Cook-Torrance`, `Leaf` ← `Oren-Nayar`, `Plastic` ← `Cook-Torrance`, `Stone` ← `Minnaert`, `Wood` ← `Ward Anisotropic`. Then, every surface was modelled with the Lambertian BRDF. The returned intensity is depicted in Figure 10.8. Finally, the first setup was replicated for TLS and ALS to illustrate in Figure 10.9 the changes obtained using different scanning methods. The first two experiments were conducted over a procedural forestry environment, whereas the last one was performed over an urban scenario as the one shown in Figure 10.7.

Surfaces with a dull appearance, such as leaves and fabric, returned lower intensity values compared to other surfaces. Additionally, a strong correlation was found between the shape of the intensity histograms and the type of scan. Airborne scans exhibit a Gaussian-like distribution due to the wider variety of surfaces, normal vectors, and incidence angles covered. It is worth noting that the FOV in ALS is typically larger than in TLS, which contributes to a greater variety of incidence angles and therefore achieves a histogram with a different shape. In contrast, terrestrial surveys have a narrower FOV and a lower variety of incidence angles, leading to a histogram with most values gathered around zero instead of being uniformly distributed.

It was also observed that changing the BRDFs to Lambertian resulted in more uniform intensity distributions. Specifically, surfaces that were initially modelled with Minnaert BRDF showed a significant increase in returned intensity after this change, while the opposite occurred for BRDFs that initially returned higher intensity values.

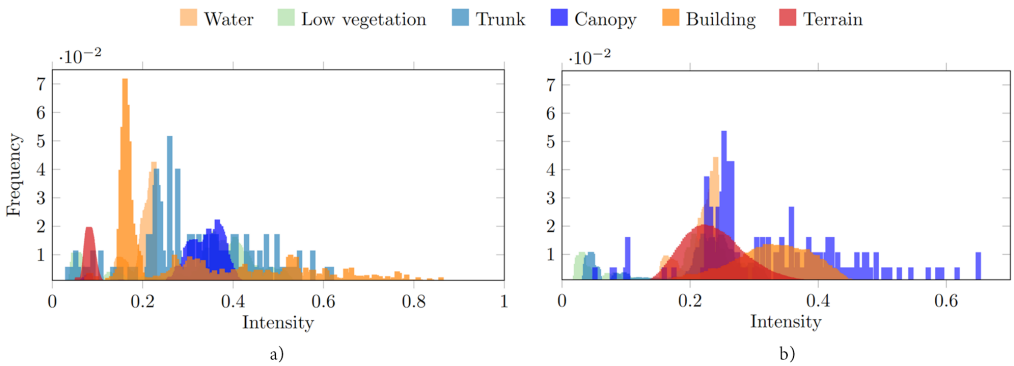


Figure 10.8: Comparison of the intensity estimated in the aerial scanning of a procedural forestry environment. a) Objects were modelled with a BRDF similar to the expected appearance, whereas b) uses a Lambertian BRDF for every surface.

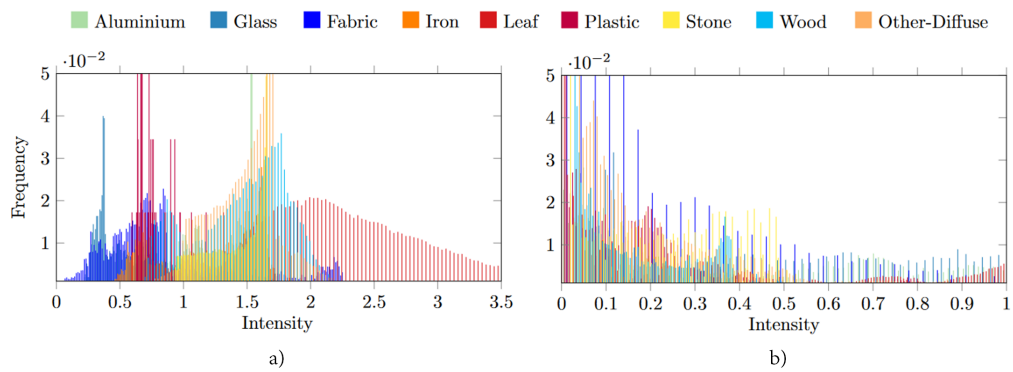


Figure 10.9: Comparison of intensity estimated from a) airborne and b) terrestrial scans.

10.5.2 BRDF database

Using this approach, the experiments were conducted by changing 1) the sensor wavelength from 903 nm to 532 nm, thus expecting a lower intensity in the latter configuration, and 2) the surface materials. First, all the surfaces were modelled with BRDFs similar to the expected appearance, ranging from acrylic to metallic and anisotropic profiles, and then, all of them were modelled with the BRDF of a maple leaf, which shows a more Lambertian-like appearance. The outcome of the three conducted experiments is illustrated in Figure 10.10.

The intensity histograms of our TLS scans are strongly influenced by both the wavelength of the LiDAR and the materials in the scene. Because our scans were conducted with a narrow FOV, most of the returned intensity

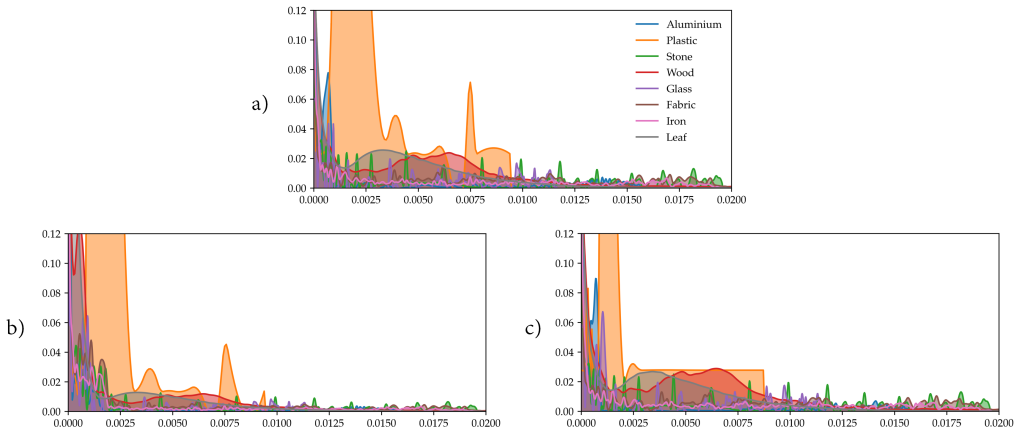


Figure 10.10: a) Intensity from the simulation of an HDL-64E LiDAR operating at 903 nm over surfaces with realistic materials, b) idem but working at 532 nm and c) the first experiment replicated with Lambertian surfaces.

values are close to zero, since the incident vector and the majority of surfaces have different orientations. This is in contrast to scans performed from a terrestrial vehicle, which reach a large number of surfaces oriented such that $\hat{n} = -\hat{r_d}$ (where n is the surface normal and r_d is the ray direction). However, the long-range capabilities of our LiDAR allow us to detect very distant surfaces with normal vectors that differ significantly from r_d, which in turn diminishes the returned intensity. Interestingly, we found that changing the LiDAR wavelength from 532 nm to 903 nm increased the returned intensity. This result was expected by observing Figure 10.6 since the majority of materials have signatures that grow as the wavelength also increases. Furthermore, using a single BRDF intended to emulate a Lambertian response led to overall brighter intensity values. However, it is important to note that this configuration does not necessarily imply upscaling of intensity values. Instead, this approach offers a mid-range intensity that is lower than the signature of some materials but brighter than materials with a dull appearance.

10.6 Conclusions and future work

Two different methodologies were evaluated for estimating the LiDAR intensity during scans solved in the GPU. The first one uses traditional shading methods, where BRDFs were expressed using formulae intended to emulate the appearance of any material. However, only six analytical BRDFs were implemented, thereby evidencing that it is not possible to cover every material found in nature. Additionally, not all of these BRDFs are physically based, and selecting one over another can be challenging. For instance, there are several BRDFs that emulate anisotropic materials. Another approach is to obtain BRDFs by sampling real-world materials using a gonio-photometer [95]. This method is based on the sampling of published BRDF databases which provide more reliable signatures and

extend the spectra from RGB to a wider wavelength range. Also, using collected BRDFs eliminates parameters used to estimate the reflectance from scratch (e.g., specular colour). Therefore, this approach was proven to be much more convenient, whereas the obtained results did not show any notable changes compared to the first approach. Indeed, the conducted TLS scans had a very similar intensity distribution.

Building on the previous chapter's work, it is essential to evaluate the simulated point clouds, along with the estimated density, using DL. This evaluation will help determine the degree of applicability of the simulated data and enable any necessary adjustments to the process. Additionally, integrating intensity data to other kinds of sensors, such as full-waveform LiDAR [415], would be highly beneficial to expand the scope of this research.

Part V

Applications of remotely sensed data

This chapter comprises several case studies where previously processed and generated data are applied to the planning of acquisition processes as well as to the detection and classification of features. First, the synthetic LiDAR data will be used for the planning of scans, in order to optimize and minimize the locations required to monitor buildings. Then, thermographic point clouds are applied to the detection of landscape features that could be related to buried remains of an archaeological site. Finally, the hyperspectral data of a vineyard plot is corrected to help in the classification of different grapevine varieties using Deep Learning.

Optimization of LiDAR scans 11

About this chapter

This chapter presents a pipeline for optimizing TLS scans in indoor environments using the previously implemented LiDAR simulation. To solve this problem, metaheuristics such as local search (LS), tabu search and genetic algorithms (GA) operate over a set of positions, whose fitness is measured after their synthetic LiDAR point cloud. The expected outcome is a set of optimal locations that help to scan a real-world scenario by ensuring proper density, coverage and quality. In comparison with previous research, buildings are represented by 3D models, thus enabling to account for sensors with adaptive height. The literature of Planning 4 Scanning (P4S) has utilized four metrics to evaluate plannings: LOA (accuracy), LOD (detail), LOO (overlap) and LOC (coverage). This work comprises the previous four metrics into three functions that reduce the number of steps in this multi-objective optimization. Following the approach of Chapter 9, the LiDAR scans as well as part of the optimizations were implemented in the GPU or as multi-core CPU. Rather than selecting among different metaheuristic algorithms, these will be chained to narrow the search space. For instance, LS helps to provide a rapid improvement of initial positions, whereas GA narrows this search space by generating a solution that encodes which positions ought to be used without worsening the metrics. That is, **which is the minimum number of points (and which points) required to solve the optimization while still guaranteeing some minimal requirements?** This problem has been previously referred to as the NP-complete set-coverage problem [282–284] if the number of solutions, n, is known. However, this case study is even more cumbersome since the number of required solutions is not known unless defined by the end user.

Case study of TLS optimization: Building Information Modelling

TLS scans are very time-consuming as they are operated by humans whose labour is to place the LiDAR sensor on a limited number of locations that may contribute, with better or worse results, to generate a composite point cloud. However, temporal repeatability must be guaranteed in some environments, thus requiring deterministic planning. A case study where these requirements are easier to understand is on the monitoring of buildings in the construction industry. Periodic scans help to track the conditions of a building, which can be saved in a Building Information Modeling (BIM) [277]. This data representation encodes the features of a building, including 3D design drawings, materials, costs and safety specifications, and provides an interface for the management of 4D applications. Together with TLS, it enables the monitoring of continuously evolving buildings to preserve cultural heritage,

11.1 On the optimization of TLS scans 206
11.2 Synthetic scenarios . . 207
11.3 Solution encoding . . . 207
11.4 Metrics 208
11.4.1 Level of accuracy, coverage and resolution 208
11.4.2 Level of overlap 210
11.4.3 Point sampling 212
11.4.4 Solution initialization . . 212
11.5 Spatial search 213
11.5.1 Solution neighbourhood 213
11.5.2 Greedy algorithm 214
11.5.3 Simulated annealing . . 214
11.5.4 Tabu search 214
11.6 Genetic algorithm . . . 215
11.7 Interactive tools 216
11.7.1 Picking of scannable building levels 216
11.7.2 Pipeline design 217
11.8 Results and discussion 217
11.8.1 Performance of local searches 218
11.8.2 Pipeline performance . . 221
11.8.3 Level of overlap 222
11.8.4 Response time 223
11.8.5 Variable scan height . . . 224
11.9 Conclusions and future work 225

Although this optimization provides a minimum number of optimal scan positions, these are still required to be performed by humans. Therefore, positions must be optimized but also narrowed to make this task less time-consuming.

track its current state and maintain repair records [89, 273, 279, 280]. However, monitoring buildings over time is time-consuming, especially in dynamic environments. Also, TLS scans are given by multiple point clouds that must be later fused, and for this task, there must be some notable overlapping among them, or at least allow us to distinguish visible markers [281]. In this case study, the development of tools for planning TLS surveys plays a key role.

The main contribution of this work is the implementation of a massively parallel optimization solver using smooth objective functions to measure the fitness of individual solutions, instead of being supervised by thresholds as in previous work. Starting solutions are locally enhanced; otherwise, a huge number of grid partitions with a large memory footprint are required. Then, the optimal subset of locations is estimated with genetic algorithms (GA), with an objective function that implicitly narrows the number of scans. The outcome of GA is further refined to guarantee the overlapping of individual scans using a Greedy approach. The input models used during the experimentation are large CAD scenarios from BIM projects authored by Autodesk Revit®. As a result, this work is able to provide a near-optimal set of locations aimed at optimizing the monitoring of buildings. Furthermore, these optimizations are performed using a parameterized LiDAR that can be adjusted according to the specifications of commercial sensors.

11.1 On the optimization of TLS scans

3D imaging technology is widespread in the construction industry for tracking building progress. It enables capturing the environment geometry in a precise and highly detailed way. Rather than collecting their geometry as polygonal meshes, these are scanned to generate 3D point clouds. In this regard, TLS has been increasingly being used to collect large datasets from buildings [85] and cultural heritage remains [89, 272, 273], among others. Some of the main shortcomings of TLS technology are the occlusion and range limitations [276]. Hence, planning TLS scans may help to mitigate this by 1) minimizing the number of scans, 2) generating a point cloud with uniform density, and 3) reducing the occlusion from scene objects. All these three objectives are equally affected by the location and configuration of the scanner. Other LiDAR technologies have arisen as suitable alternatives to TLS for addressing the aforementioned drawbacks. However, these are still recent, and despite solving some of these drawbacks, others arise. For instance, LiDARs mounted on backpacks reduce the acquisition time and cover large areas [418], but the path should be carefully planned to avoid occlusion and guarantee coverage. Also, their point density is lower [419].

Regarding the optimization of TLS scans in buildings, this field of research is frequently referred to as Planning for Scanning (P4S). Most of the studies narrow 3D environments to 2D cross-sections, or simplify the scene primitives into others whose collisions are easier to solve, even wrapping a set of primitives (e.g., voxelizations [288]). Also, it often requires previous knowledge of the scenario. For instance, OOBBs have been utilized over en-

vironments in which AABBs did not fit well to rotated, non-planar surfaces [231]. On the other hand, DSMs have also been explored in 2.5D optimizations [291]. However, most of these works are based on the iterative find of the Next Best View (NBV), while others apply heuristics such as Greedy [286], SA [294], GA [295], Particle Swarm Optimization (PSO) [295] and Integer Programming [288].

The main drawbacks found in previous P4S work are 1) the widespread use of greedy algorithms to determine optimal plannings, 2) the simplification of input scenarios to speed up solutions, 3) the use of objective functions relying on user-defined thresholds (range, angle, etc.) and 4) the lack of massively parallel algorithms that help in reducing the latency.

11.2 Synthetic scenarios

Input scenarios are triangle meshes depicting buildings with one or multiple levels, extracted from publicly available BIM projects. These are not constrained to any standard, and therefore, the number and size of the polygons vary. Typically, polygons in walls, floor and ceiling surfaces are larger than those used to represent furniture. However, TLS scans must reach as many polygons as possible. Consequently, operating with surfaces of similar dimensions would greatly benefit a non-weighted planning algorithm. To this end, triangle meshes can be subdivided to reach a nearly uniform triangle size. Note that it is hard to achieve since items such as furniture have a high LOD that cannot be reached by other items, at least not without a large memory footprint. From here, the best choice is to recursively subdivide polygons until their area is below a threshold. Following, scenarios are indexed in a BVH to be rapidly traversed.

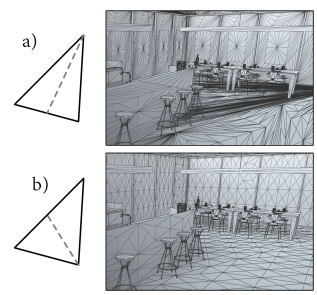


Figure 11.1: Two different triangle subdivisions over the same scene. a) Triangles subdivided by iterating through the edge to be split, and b) triangles subdivided with the proposed method.

Several approaches are effective to subdivide polygons, although some of them yield aesthetic results (see Figure 11.1). Whether the edge to be split is selected randomly, the triangle can be subdivided into large triangles that degenerate into segment-like polygons. This hardens the detection of collisions in later ray-triangle intersections, together with the loss of precision from using floating point data. Instead, the longest edge within each triangle can be split, thereby generating triangles with a uniform edge length.

11.3 Solution encoding

Candidate solutions in GA are encoded as a buffer of binary values indicating which minimal set of LiDAR positions will be used for scanning. Each binary value is defined as an activation indicating whether this location will be used or not in the final planning. Firstly, positions are either uniformly or randomly sampled along the selected building level. Then, these are optimized with small changes whose impact will be measured by means of multiple metrics. Finally, GAs are intended to minimize the number of TLS scans while maximizing the coverage of a LiDAR point cloud.

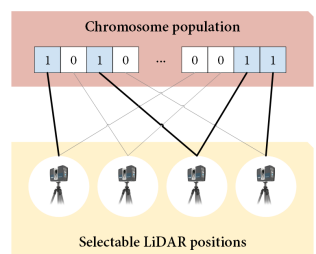


Figure 11.2: Binary encoding of active LiDAR solutions for a genetic algorithm.

With this sequence in mind, initial solutions are stored as xyz coordinates, whereas the solutions of GA are encoded as vectors of binary values $[b_0, b_1, b_2, ..., b_{k-1}]$, where $b_i \in \{0, 1\}$ and k is the number of solutions generated on the previous spatial search. Although activating k solutions provide the most complete scan, the GA is intended to minimize the time required by human operators for scanning. Figure 11.2 shows the proposed solution encoding for GAs. Binary values can be represented by Boolean data in the CPU; however, this data type is not uniformly represented in CPU and GPU hardware. Instead, Boolean values can be expressed using a minimal integer encoding in the GPU hardware with the `uint8_t` data type. In the GPU, it requires using the recent GLSL's `GL_NV_gpu_shader5` extension. Representations with a lower memory footprint are indeed possible with bit-level encoding, at the expense of more intricate operations in accesses and modifications. Nevertheless, the experiments that are later presented allocate a few MBs at most.

11.4 Metrics

Fitness functions evaluate the quality of candidate solutions to help in their optimization and determine which are the best candidates. In this section, a solution is represented as a set of n different scans in the same building level, which are evaluated using four metrics. First, the number of scanned polygons provides a trivial measure of the coverage of the scenario (F_1: LOC). Also, the uniformity of the scanned points in polygons helps to account for LOD, LOA and LOC (F_2). Finally, TLS scans require a minimum overlapping factor among them to guarantee that they can be later fused in post-processing (F_3: LOO).

11.4.1 Level of accuracy, coverage and resolution

The LOC metric is trivially solved by calculating the number of scanned polygons. However, measuring the uniformity of a point cloud is not as trivial. Instead of providing the required resolution, the optimal value can be computed during pre-processing. Therefore, a nearly optimal resolution is computed polygon-wise by measuring the average distance between uniformly sparsed points. These points were obtained by retrieving two values (u and v) from a uniform distribution and used together with the parametric form of a triangle. Therefore, polygons with a mean distance above the reference mean are accordingly penalized on the overall fitness of a planning solution. As a result, point clouds covering a polygon only with a few points within a triangle are heavily penalized (see Figure 11.3). As in previous chapters, QMC samplers approximate better what is expected from a randomized, yet uniform, scan.

F_1 and F_2 metrics allow enhancing starting positions by translating them towards better locations. They can be either applied to one or multiple locations. In terms of preferences, the point cloud's uniformity and accuracy are typically chosen over coverage for a single scan. Despite this, both F_1

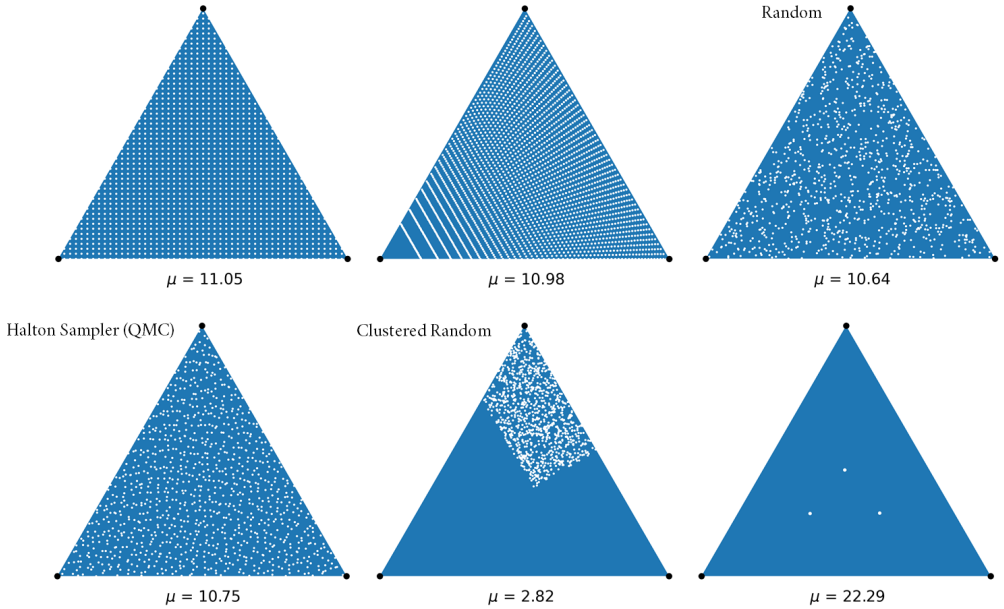


Figure 11.3: Average distance (μ) of every sampled point with the rest. From left to right, and from top to bottom: uniform grid distribution, uniform random sampling of u and v, random distribution, QMC random distribution, random distribution only for a part of the triangle and three points uniformly scattered.

and F_2 are evaluated during optimization, with F_1 working as a secondary metric that helps to untie scenarios where multiple solutions have the same F_2 value. F_1 and F_2 are defined in Equation 11.1 and 11.2 for evaluating a set of solutions, K. Individual solutions have a buffer of collided triangles, T, with each one being intersected by a set of points, P. The set of collided triangle indices is denoted by L, with each index appearing only once. Finally, the whole set of triangles is S.

$$F_1 = \sum_{s=1}^{|S|} 1[t_s \in \{L_1, L_2, ..., L_k\}] \tag{11.1}$$

$$F_2 = \sum_{k=1}^{|K|} \left(\sum_{s=1}^{|S|} d_{optimal_s} - \sum_{l=1}^{|L_k|} d_{optimal_l} + \right.$$
$$\left. + \sum_{t=1}^{|T_k|} \left(\frac{\sum_{i=1}^{|P_t|} \frac{\sum_{j=1}^{|P_t|} d^2(p_i, p_j)}{\max(|P_t|-1,1)} \cdot \left(2 - \left| \widehat{n}_{t_k} \cdot \widehat{(r_o - p_i)} \right| \right)}{\max\left(|P_t|, 1\right)} - d_{optimal_t} \right) \right) \tag{11.2}$$

where F_1 represents the number of different intersected polygons and F_2 estimates the accuracy by measuring the distance from the optimal uniformity to the computed one. d represents a distance function, e.g., the Euclidean distance, t_s is a triangle index in S, $\hat{n}$ is the normal vector of a triangle, and r_o is the location of the sensor's emitter. Instead of only considering the average dissimilarity from the expected uniformity in intersected primitives, F_2 also sums the dissimilarity from the rest of, the non-collided,

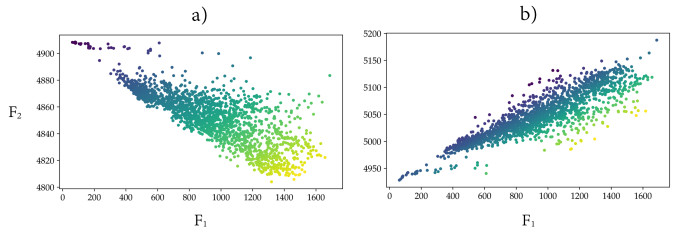


Figure 11.4: F_1 and F_2 results from uniformly sampling a 3D environment. The first image depicts how F_2 behave with the proposed formula (F_2 must be minimized, whereas F_1 is intended to be maximized). The second plot does not remove the optimal average distance of collided polygons.

polygons. However, the summed dissimilarity from intersected polygons must be subtracted to avoid penalizing solutions with a higher number of points, as illustrated in Figure 11.4. Remark that F_2 intends to minimize the measured dissimilarity to the estimated optimal inter-point distance. Therefore, the number of reached polygons, F_1, is implicitly considered in this expression by penalizing scans that only reach a small number of polygons.

Also, note the sensor's range is frequently integrated into the LOA metric by discarding returns whose distance is above the maximum allowed distance. However, these criteria can be integrated during the LiDAR simulation, rather than worsening the efficiency of F_2 estimations. The approach followed considers a distance much lower than the typical range of real-world sensors, thus improving the LOA of the simulated planning.

11.4.2 Level of overlap

The F_2 metric is better suited for achieving local improvements, whereas F_1 fits better on the selection of the minimum number of locations providing better scene coverage. However, none of these metrics considers the overlapping of individual scans, despite it being required to guarantee they can be combined in post-processing. To this end, features visible in one scan should also appear in other close scans. With this objective, the F_3 metric defined in Equation 11.3 measures the overlapping area between two LiDAR scans, indexed as i and j, and depicted as circumferences with a fixed radius, r.

$$
\begin{aligned}
A_{i,j} &= \left(r_i^2 \cos^{-1}\left(\frac{d_i}{r_i}\right) - d_i \sqrt{r_i^2 - d_i^2} + \right. \\
&\quad \left. + r_j^2 \cos^{-1}\left(\frac{d_j}{r_j}\right) - d_j \sqrt{r_j^2 - d_j^2} \right) \\
A_{i,j,r_i=r_j} &= 2\left(r^2 \cos^{-1}\left(\frac{d}{2r}\right) - \frac{d}{2}\sqrt{r^2 - \left(\frac{d}{2}\right)^2} \right) \\
F_{3_{i,j}^{\theta}} &= \frac{A_{i,j}}{\pi r_i^2} \mid \quad i,j = 1,2,\ldots,|P_c|
\end{aligned}
\tag{11.3}
$$

where d is the distance from two locations ($d = d_i + d_j$) and r is the LiDAR maximum range, which was previously clamped to account for the accuracy

loss from distance. Note that r is the same for every scan ($d_i = d_j$).

The main drawback of Equation 11.3 is that it does not account for the overlapping of previously added locations. To handle this, the occupancy of each LiDAR scan is represented using a 2D grid that is filled with points sampled from other scans. Hence, the estimated overlap is given by the number of voxels occupied divided by the overall number. If $d \geq d_1 + d_2$, the sampling can be omitted since both locations do not overlap. Yet, Equation 11.3 helps to establish a ranking of candidate solutions, thus measuring how helpful is every scan to achieve the required overlapping. However, ranking locations by their degree of overlap leads to selecting those with the highest overlap instead of the required one. Accordingly, Equation 11.4 ranks locations by their distance to the required overlap. With this approach, solutions that offer a higher overlap than the required one are preferred over those below. To guarantee this, the most significant bit of preferred locations is set to one. Due to the sampling strategy, the accuracy of the overlapping estimations depends on the grid subdivision as well as on the number of sampled points belonging to another TLS scan, as depicted in Figure 11.5.

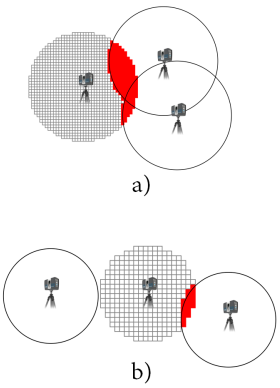


Figure 11.5: Grid occupancy in two different configurations. a) Grid with higher resolution and three overlapping circumferences, and b) sparser grid with only two overlapping circumferences.

$$F_{3_{i,j}} = \begin{cases} \left(r - \left|\psi - F_{3^{\theta}_{i,j}}\right|\right) \mid 1 \ll 32 & F_{3^{\theta}_{i,j}} \geq \psi \\ r - \left|\psi - F_{3^{\theta}_{i,j}}\right| & \text{otherwise} \end{cases} \tag{11.4}$$

with $\psi \in [0, 1]$ being the required overlapping.

Following this approach, the required overlapping is guaranteed for every solution. Still, another shortcoming is the existence of disjoint sets as a result of selecting the best n solutions with a greedy algorithm. Similarly, these flaws can be detected using a disjoint set. New solutions, which belong at least to a disjoint set, are selected according to their distance to another disjoint set and the value of F_3, as proposed in Equation 11.5. The stop criteria are conditioned by the number of disjoint sets, which should be one, and the number of scans, which limits the number of available solutions. The procedure to combine disjoint sets is the following:

- First, the closest disjoint sets are selected, as well as the two closest solutions within both of them, p_o and p_d.
- Then, solutions overlapped with the first location are sorted according to the expression in Equation 11.5, which must be maximized.
- The new LiDAR location is linked to the first solution, which is known to be overlapped, whereas the second solution is only linked whether the overlapping is higher than zero.
- New solutions are also checked to ensure the required overlapping.

$$g(p_o, p_d, p_i) = \left(\frac{d(p_o, p_d) - d(p_i, p_d)}{d(p_o, p_d)} F_{3_{o,i}}\right) \tag{11.5}$$

11.4.3 Point sampling

During pre-processing, polygons are iteratively sampled to generate uniformly distributed point clouds. Then, the average distance of every point in comparison with the rest is calculated. The overall average distance for a polygon is defined in Equation 11.6. The Euclidean distance is used by default, although it can be exchanged by any other distance function. Note that $|P| - 1$ is used instead of $|P|$ to not take into account the distance of points with themselves in an efficient way.

$$d_\mu = \frac{\sum_{i=1}^{|P_t|} \frac{\sum_{j=1}^{|P_t|} d^2(p_i - p_j)}{\max(|P|-1,1)}}{\max\left(|P|, 1\right)} \tag{11.6}$$

where P_t represents a set of points in a triangle t and the inner summation computes the Euclidean distance for 3D points.

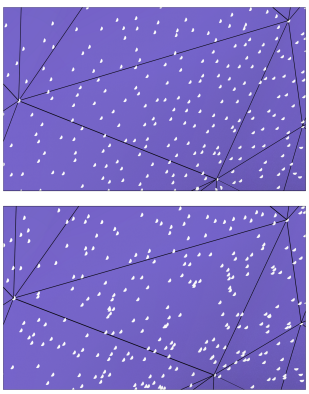


Figure 11.6: Comparison of polygons sampled using the Halton sequence and the C++ built-in random uniform distribution.

Although built-in random uniform distributions are available in most language libraries, these are mainly based on pseudo-random sequences such as those used for Classic Monte Carlo integration (CMC). They provide good quality estimations, but they require a large number of samples and have a higher response time [420]. The latter drawback is irrelevant in this case study since only a few points will be sampled from each triangle, with the number being conditioned by the polygon area. However, these distributions generate point clouds far from the expected uniformity. Instead, the QMC method is based on well-distributed deterministic sampling patterns that provide better results for our objective. More specifically, the Halton sequence was used to calculate the quasi-optimal mean distance within polygons [421]. Figure 11.6 shows the outcome of sampling points with the C++ built-in random uniform distribution and the proposed QMC sequence.

Either from pseudo-random or QMC sequences, two parametric values (u, v) are generated to sample a triangle surface. Equation 11.7 shows the formula for generating a point within a triangle defined through its three vertices (p_1, p_2, p_3) with $u, v \in [0, 1]$.

$$p_s = (1 - \sqrt{u})p_1 + (\sqrt{u}(1 - v))p_2 + (v\sqrt{u})p_3 \tag{11.7}$$

Note that a significant drawback of this approach is that sparse scans with all their points gathered in a small area are also evaluated as appropriately uniform. To solve this, the triangle vertices were included as part of the point set, $|P|$, thus penalizing scans biased towards certain parts of polygons.

11.4.4 Solution initialization

The initial search space must be partitioned to obtain the first solutions (N) over polygons that are interactively marked as ground. From the collected ground planes, the 2D AABB is estimated to restrict the search space. However, an AABB does not adjust well to most of the possible shapes,

and may still lead to instancing solutions out of the floor model. Therefore, candidate locations must be evaluated to ensure that:

- **The location is over a ground polygon**. To verify this, a ray is cast towards -Y in the BVH. Ground polygons are transferred to the GPU and their Id is compared against the nearest collision.
- **The sensor is not located over non-ground planes**, e.g., furniture placed between the floor and the LiDAR sensor. m points are sampled around the candidate location, using a radius r and casting m rays with $-\vec{Y}$ direction. The location is discarded whether any of the m rays have their nearest collision in non-ground-labelled items. Both r and m can be configured.
- **The sensor is not located close to building walls and furniture**. It cannot be placed over items, nor next to them, thus guaranteeing a safety distance. Hence, four additional rays are cast using {X, -X, Z, -Z} vectors.

At least two ways to initialize solutions were integrated: random and uniform sampling. The uniform sampling utilizes a 3D matrix bounded by the ground AABB, whereas the sensor height is limited by the LiDAR's maximum and minimum height. The LOD of the partitioning is parameterized with the voxel size; a lower size provides a better LOD and a higher number of solutions. Otherwise, solutions can be randomly distributed to provide a less exhaustive initialization. The latter approach explores fewer solutions, scattered throughout the scene using a random uniform distribution. Note that in this approach uniformity is not a must; instead, it is avoided to provide an alternative to the uniform, grid-like, sampling. Then, the set N is either enhanced (spatial search), subsampled (greedy algorithm) or narrowed with bit sets (GA).

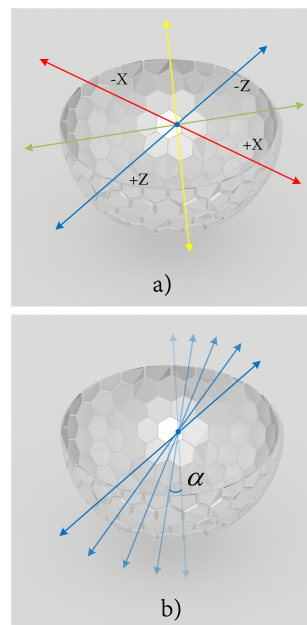


Figure 11.7: Overview of gradients for a 2D neighbour search. a) A fixed approach generates eight vectors, whereas b) the subsampling of a circumference can be performed with a variable resolution.

11.5 Spatial search

This section explains how the initial solutions, N, change and improve to provide a wider space exploration, instead of simply relying on the starting locations. However, metrics aimed at improving scans do not have any knowledge concerning overlapping. The following searches work according to the F_2 metric from Equation 11.2.

11.5.1 Solution neighbourhood

Neighbourhood exploration is aimed at locally enhancing candidate scans. The neighbourhood of a scan is a set of 3D points close to the sensor location. Discretization poses several challenges since there is an infinite number of possible translations. The first approach is to downscale the neighbourhood to 14 unitary vectors (six faces and eight corners of a cube). Otherwise, it can be discretized by uniformly subsampling a unit sphere, thus obtaining a buffer of gradients whose length depends on the user-defined sampling resolution. Furthermore, unit vectors can be down- and up-scaled with a step length: larger changes may omit good candidates, and the contrary

may be too exhaustive. Figure 11.7 illustrates the gradient vectors that help to explore the contiguous space in 2D: the first scenario has fixed partitions, whereas the second depends on the user-defined resolution.

11.5.2 Greedy algorithm

The traditional local search (LS) is here referred to as a Greedy algorithm and helps to rapidly survey the neighbourhood. The surrounding points are evaluated and sorted according to the results of the metrics. Solutions are iteratively improved by adopting the location of the best neighbour if any improves the current location. Otherwise, this approach gets stuck on local minima and terminates. The termination criteria also contemplate reaching a maximum number of iterations. Re-visiting previously explored locations is avoided with the dot product $(\cdot)$ between the gradient being evaluated and the previous one ($f(\hat{g}_1, \hat{g}_2) = \hat{g}_1 \cdot \hat{g}_2$). Hence, a gradient is considered to be already explored whether $f(\hat{g}_1, \hat{g}_2) < \epsilon - 1$, with ϵ being a small value such as 2^{-16} that avoids errors derived from comparing floating-point data. Note that this test is very efficient and copes with immediate loops; however, loops with a deep higher than one are possible.

11.5.3 Simulated annealing

SA, in contrast to the greedy approach, avoids getting stuck in local optima. To this end, neighbours with worse fitness than the current solution are accepted under probabilistic criteria. The acceptance ratio is modelled as a temperature value, T, which is first initialized and downscaled iteratively according to a cooling factor [422]. In this work, it is configured using the traditional exponential formula defined in Equation 11.8:

$$p(f') = \begin{cases} \exp(\frac{f'-f}{T}) & f' - f \geq 0 \\ 1 & f' - f < 0 \end{cases} \tag{11.8}$$

given that f' and f are two fitness values from a new solution and the current one, respectively, in a minimization problem. Consequently, a random uniform distribution is also applied to determine whether a worse solution is accepted. The stop criterion depends on a threshold temperature and a maximum number of iterations.

11.5.4 Tabu search

A trivial improvement of LS is given by a tabu search (TS), where the already processed locations are avoided with the so-called tabu moves. The explored solutions cannot be re-explored and therefore, all the neighbours may be tabu. In this case, the solution space can be re-initialized. The very same scenario is achieved if every neighbour worsens the fitness of the current location, or a maximum number of iterations is reached without any improvement. The main shortcoming of 3D explorations is how to

store tabu moves, which was approached using a 3D regular grid of variable resolution. Hence, a neighbour is considered tabu when the target point falls in a tabu voxel. Given the step size of neighbour explorations, n_l, an appropriate voxel size is given by $f \cdot n_l$, with $1 < f < 2$.

11.6 Genetic algorithm

Genetic algorithms operate over previously enhanced solutions to find an optimal subset. This sort of algorithm is inspired by nature on how to handle populations that evolve and improve over time. Accordingly, it combines exploration and exploitation phases [423]. Regarding our case study, this metaheuristic helps to reduce the setup complexity while still providing dense coverage. In comparison with memetic algorithms, the first stage aims to improve individual locations guided by an exploration of surrounding areas, without evaluating the scene coverage. Yet, the proposed F_2 metric weights non-scanned polygons. After this, a GA finds the optimal solution regarding the scene coverage. To this end, a slight variation of the previously proposed F_1 metric is here proposed. The complete procedure is depicted in Figure 11.8, whose stages are following detailed:

Initialization of population. The solution space is first populated with p chromosomes of size n, each one with a random number of activations (LiDAR locations). Hence, variety is ensured by avoiding populations with a fixed number of activated bits.

Compute template rays. Template rays are intended to describe how rays traverse the space from a default LiDAR location, which is known to be at 0^3. These rays are generated according to the specifications of a commercial LiDAR sensor. Then, the LiDAR point clouds are simulated from this set of rays. Template rays are only transferred once to the GPU, rather than calculating and transferring them for every location to be checked. The GPU-based LiDAR simulation proposed in previous chapters was simplified to provide only the first collision to favour both replicability and performance.

Parent selection. The aim of this stage is to select the s most promising individuals for the subsequent crossover. Two different selections are here proposed. First, the 2-by-2 tournament allows selecting good solutions, although they may not be in the top-most s. This approach favours the variability of later populations. On the contrary, the elitist selection chooses the best individuals from the current population.

Crossover. Previously selected parents are combined to generate the next population. Parents can be mixed by selecting a crossing point which split each chromosome into two parts which are combined to build offspring. Another approach is to randomize the crossover by utilizing random values which determine the parent from which each gen is obtained (p_{1_i} if $r_i \leq 0.5$, p_{2_i} otherwise). There is only a non-valid solution, given by the zero chromosome (0^n), which is easily handled by randomly altering a bit.

Mutation. Mutations include minor variations in new generations. This

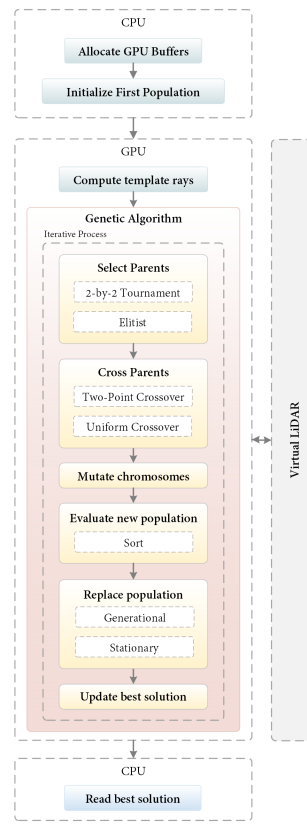


Figure 11.8: Overview of GAs on the evaluation of the best combination of spatial LiDAR setups.

stage is parameterized by the number of mutable individuals and genes within each one. No restrictions are considered regarding the number of activations, and therefore, mutations simply activate or deactivate some random bits.

Evaluation of new population. The fitness of every new solution is estimated based on the computed returns. Accordingly, solutions covering more polygons are more likely to be selected as parents in the following iterations. However, this approach favours activating every bit in every chromosome. Instead, the previously proposed F_1 metric is modified according to Equation 11.9 by also accounting for the number of activated bits within a chromosome (c_g). With this new metric, the genetic algorithm is intended to reach the maximum number of polygons with the minimum number of scans.

$$F_1 = \frac{\sum_{s=1}^{|S|} 1[t_s \in \{L_1, ..., L_k\}]}{\sum_{g=1}^{n} c_g} \tag{11.9}$$

Replace population. New and previous populations are then mixed following the generational or stationary approach. The generational approach replaces parents with their offspring, whereas the stationary algorithm replaces the worst individuals with the new population whether they improve current solutions.

Update best solution. It is noteworthy that even this minor operation must be performed in the GPU, as the reading transfer would lead to a huge delay in the GA response time.

11.7 Interactive tools

This section presents some tools intended to facilitate the configuration of the scenario and the algorithm.

11.7.1 Picking of scannable building levels

The described P4S method is not restricted to buildings with a single level. Indeed, it has been evaluated in realistic buildings with several of them. However, the planning must be addressed individually. The selection of floor-labelled polygons is interactively performed with ray-casting, thus obtaining the intersected primitive in real time. The result is computed by casting a ray whose direction is given by the user's viewpoint and a 3D position, which is computed by unprojecting the camera matrix to a 2D point within the application canvas. The first collided model is obtained traversing the BVH and pushed into the ground list. With this regard, Equations 11.10 and 11.11 show how to unproject a 2D canvas point to 3D.

$$p_{3D} = (P \cdot V)^{-1} \cdot \left[\frac{p_{2D_x}}{c_x} 2 - 1, \frac{p_{2D_y}}{c_y} 2 - 1, 0, 1 \right]^T \tag{11.10}$$

$$p_{3D} = \frac{p_{3D}}{p_{3D_w}} \tag{11.11}$$

where (p_{2D_x}, p_{2D_y}) is the canvas point, P, V are the camera projection and view matrices, respectively, (c_x, c_y) is the canvas size, and p_{3D} is a 4-tuple describing the projection coordinates.

11.7.2 Pipeline design

Spatial searches and GAs can be concatenated to iteratively improve the starting solutions. Hence, this pipeline is not solely limited to applying LS + GAs. Instead, a flexible pipeline was implemented to omit stages if necessary and even concatenate several LS. The pipeline design was facilitated using a node editor as the one depicted in Figure 11.9. Greedy, SA, TS and GA nodes can be chained one after another, and even the spatial searches were parameterized to support the narrowing of solutions according to their fitness and the required number. Note that, by default, the maximum number of solutions is not required; instead, it is intended to be implicitly computed in the GA.

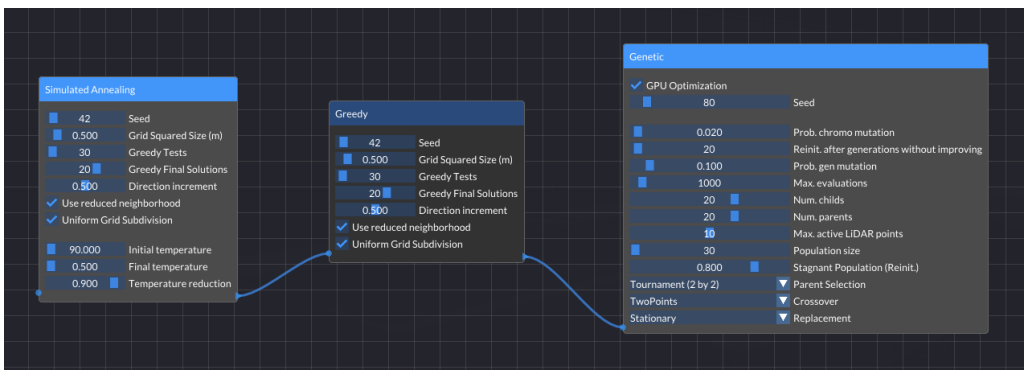


Figure 11.9: Showcase of a node editor chaining SA, Greedy and GA nodes.

11.8 Results and discussion

The proposed optimization has been tested against three different scenes from commercial software (see Figure 11.10), with a size ranging from 130k to 2.7M polygons. Although there exist a few studies concerning P4S in 3D, their implementations are not available. Consequently, the carried out experiments are focused on evaluating the results of metrics and the performance speedup in comparison with a multi-core CPU-based approach. The latter version is implemented using OpenMP when possible; however, the LiDAR simulations are very time-consuming and they were solved in the GPU in both approaches. The configuration of local searches (LS from now on) is shown in Table 11.1, whereas LiDAR specifications are detailed in Table 11.2. All measurements were performed on a PC with AMD Ryzen Threadripper 3970X 3.6 GHz, 256 GB RAM, two NVIDIA RTX A6000 GPU and Windows 10 OS.

This section is organized as follows. First, the performance of different

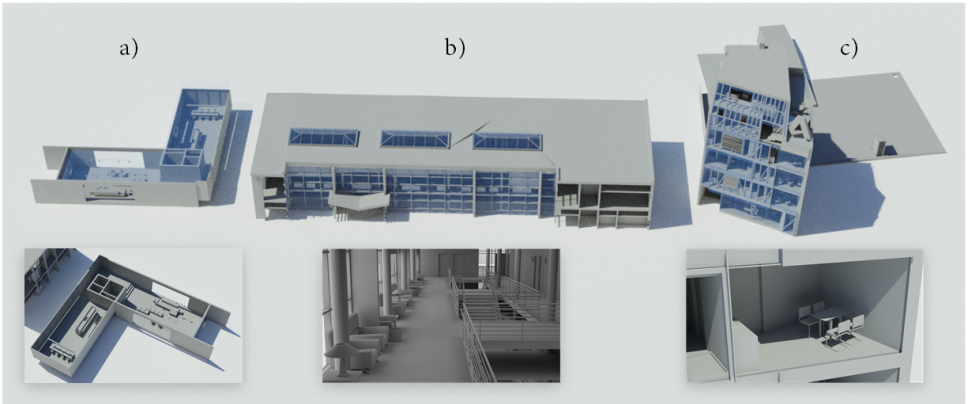


Figure 11.10: Three different environments authored by Autodesk Revit ® . a) A basement with 130k triangles ($20 \times 4 \times 21$m), b) a school with 500K triangles ($44 \times 24 \times 32$m) and c), an office building with 2.7M triangles ($16 \times 8 \times 49$m).

LS algorithms was evaluated regarding F_1 and F_2 metrics. Then, LS and GA algorithms were combined to find the best pipeline, also in terms of metrics. CPU and GPU approaches were assessed according to the measured response time, and finally, an experiment was carried out to show the benefits of considering LiDAR sensors with variable height.

11.8.1 Performance of local searches

This section shows how local searches improved the F_2 results using greedy, LS, SA and TS over the basement scene. The evaluation of the F_2 metric for each initial solution is depicted regardless of the best solution observed until such iteration. Although LS achieved huge improvements, the importance of GA selection is evidenced by Figure 11.11. It shows the narrowing phase of a greedy algorithm that selects the best $k \leftarrow 30$ scans (red) in a scenario with areas that have disparate geometrical complexity, guided by the minimization of the F_2 metric. Hence, most of the selected locations were surrounded by dense geometry, and yet, it managed to cover most of the scene. Similarly, small enclosed rooms, as the zoom-in of Figure 11.11, are not scanned due to their smaller number of polygons.

Table 11.1: Configuration of parameters concerning greedy, SA and TS algorithms.

Attributes	Value
Starting solutions	Grid sampling
Number of final solutions	Without restrictions
Voxel size	0.5 m × 0.4 m
Max. number of iterations	60
Max. number of iterations without improvements	10
Neighbourhood	Discrete (16)
Length of neighbourhood steps (Δ)	0.05 m
Initial temperature (T_0)	450°C
Temperature decrease (T_z)	0.8 T_{z-1}

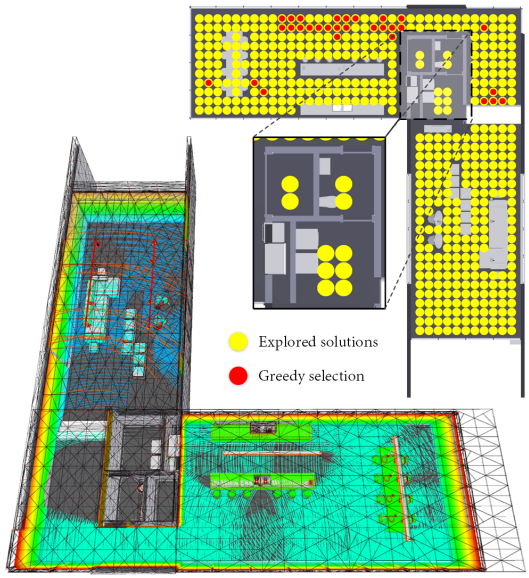


Figure 11.11: Selection of the best thirty LiDAR scans with a greedy approach in the basement scene.

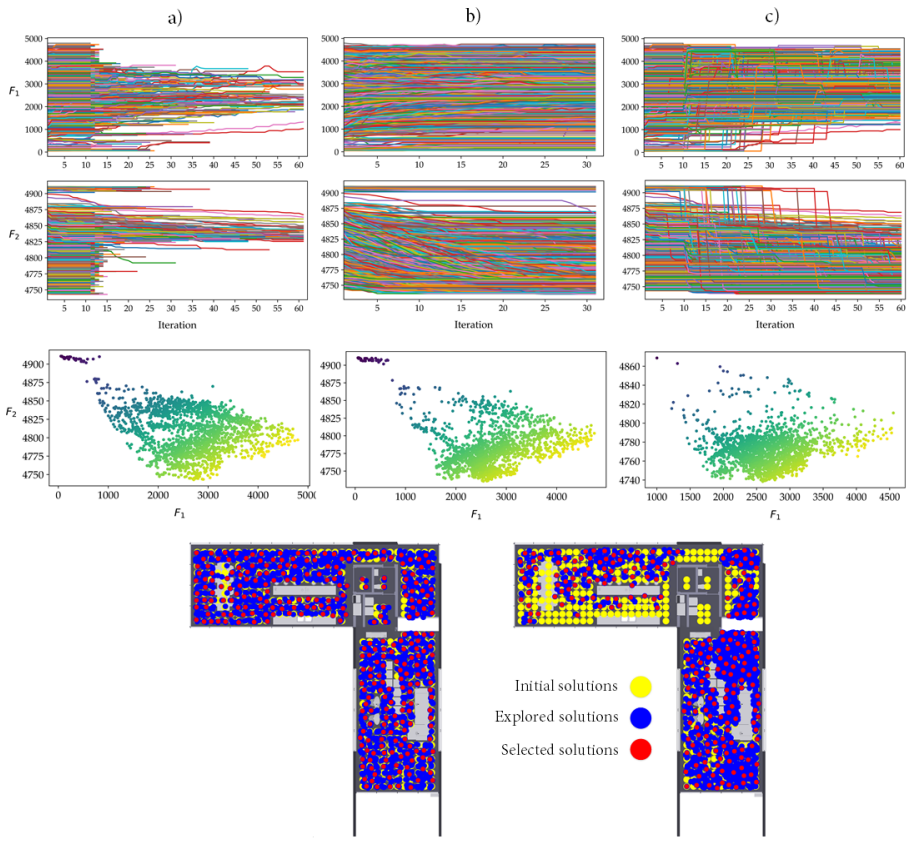


Figure 11.12: Optimization obtained using a) naive LS, b) SA and c) TS, showing both F_1 and F_2 results in a timeline as well as the final distribution. The bottom images show the locations optimized with SA and TS, organized into a regular grid and filtered by selecting only one solution per subdivision.

Table 11.2: Specifications of LiDAR sensor during optimization, following the commercial device HDL-64E.

[a] It has been adapted to reduce the optimization latency.
[b] It has been adapted to improve the accuracy of the results.

Attributes	Value
Resolution	4500×64 beams
Maximum number of returns	1^a
Maximum range	$5\ \mathrm{m}^b$
Coverage	360°x 26.9° (-24.9°-2°)
Height coverage	$[0.5, 2]\ \mathrm{m}$
Minimum xz distance to items	$0.4\ \mathrm{m}$
Vertical tests	$\mathrm{r} \leftarrow 0.2\ \mathrm{m}$, $\mathrm{n} \leftarrow 16$ rays

From this baseline, Figure 11.12 depicts the improvements of initial solutions with different LS. For that purpose, the F_1 and F_2 results achieved by individual solutions are depicted during a fixed number of iterations. However, some explorations, such as SA, finished earlier since the temperature also converged sooner or no better solutions were found. Optimized locations are not narrowed using GA, and therefore, those depicted are filtered according to a regular grid that allows selecting a single solution per partition.

Local search. As expected from the description of LS, the explorations of initial solutions finished early, especially for locations that start with low F_2 values. Worse solutions than the current one were not accepted, thus rapidly achieving the maximum number of iterations. Also, changes were controlled with a small factor; otherwise, the grid-like sampling turn into a random sampling. Due to the naïve mechanism of LS, most locations converged into locations very close to walls, as regarded by Soudarissanane and Lindenberg [276], since they maximize F_1 (number of collisions) with a low incidence angle (better LOA).

Simulated annealing. As opposed to LS, SA was able to temporarily worsen the metric results and helps to explore a wider area in xz and y. Hence, most of the initial locations reached lower values of F_2 than LS, as shown in the scatter plot. Implicitly, initial solutions moved to locations that covered a larger number of polygons since F_1 is integrated into the F_2 formula. However, early-finished explorations were not as frequent as in LS. Despite LOC being included in F_2, the line graph shows iterations that reached a lower number of collided polygons than previous solutions, i.e., worsened the LOC metric. It is not correlated to SA accepting worse solutions, since LOC is not a primary metric at this stage; instead, it is the consequence of changes that improve their accuracy at the expense of worsening the coverage. For that reason, the subsequent GA stage is designed to provide better coverage.

Tabu search. By avoiding previous moves and re-initializing when getting stuck, initial solutions were slightly more optimal in terms of F_2 (not for F_1) than those obtained by previous algorithms. However, they were not as uniformly sparsed since 1) the algorithm showed a preference for areas with higher geometrical density and 2) random re-initialization favoured this. Therefore, the local improvement of SA is preferred over TS as it offers a wider range of possibilities to the later GA, whereas TS offers immediate results. Also, note that the steeper metric signatures are the

result of re-initialization by selecting a new set of random points. Without re-initialization, TS turns into a more strict LS.

11.8.2 Pipeline performance

One of the main challenges of planning LiDAR set-ups is to cover the whole building level, including enclosed rooms. However, previous approaches optimize local metrics, where the sorting and narrowing of locations according to F_2 only ensures a dense coverage of the surrounding area. A better selection is provided by GAs guided by a global metric measuring the polygon coverage. These were evaluated using the configuration in Table 11.3 by combining them with a starting LS. Hence, different pipelines were checked, including LS + GA, SA + GA and TS + GA. Besides coverage, the implicit selection of LiDAR locations is also observed during these tests.

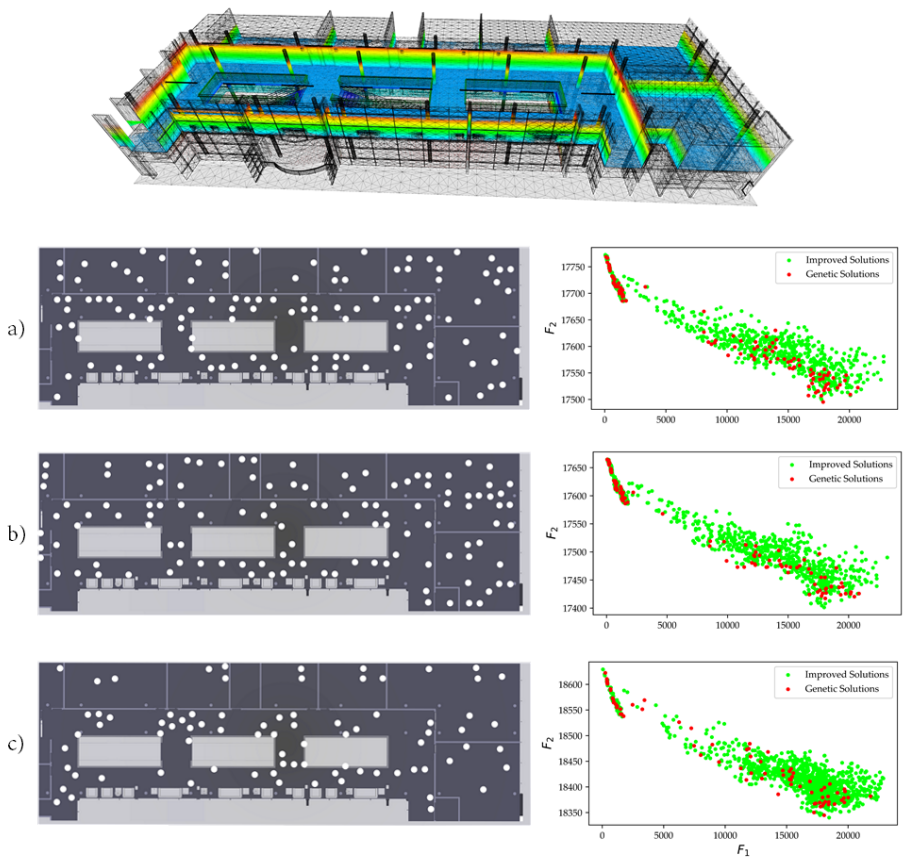


Figure 11.13: Results obtained by combining local searches and genetic algorithms. The left images depict the final LiDAR distribution, whereas the right images show the improvement of intermediate results with regard to the proposed metric ($Y \leftarrow$ Minimization, $X \leftarrow$ Maximization). a), b) and c) refer to LS + GA, SA + GA and TS + GA combinations.

The results from three different pipelines are depicted in Figure 11.13, using a single level of the School scenario. This scenario includes multiple

Table 11.3: Configuration of the evaluated genetic algorithm.

Attributes	Value
Maximum number of evaluations	50,000
Population size	500
Number of parents	250
$\mathscr{P}$(Chromosome mutation)	0.02
$\mathscr{P}$(Gen mutation)	0.1
Stagnant population (re-initialization)	80%
Stuck at local minima (re-initialization)	30 it. without improving
Crossover	Two points
Selection	2-tuple tournament
Replacement	Stationary

enclosed rooms that cannot be scanned from central areas, as well as gaps in the centre that lead to scanning lower building levels. In this regard, the tabu search (c) obtained a lower number of points, though most of them were gathered in the main room, similar to locally improved solutions being translated to areas with higher geometrical complexity. In contrast to TS, LS and SA provided a higher variety of locations, thereby helping to select a higher number of locations with better coverage of enclosed rooms. Also, note that the LiDAR range was limited to $r \leftarrow 5$ m to account for the accuracy loss. Despite LS and SA providing similar results, the pipeline SA + GA managed to scan the small rooms on the left and right sides due to its wider exploration. Nevertheless, the performance in terms of F_1 and F_2 was similar for every spatial search, as shown in the scatter plots. It can be concluded that the subsequent GA is significantly conditioned by the previous local search.

11.8.3 Level of overlap

Previous LiDAR locations were improved and narrowed according to F_1 and F_2 metrics to fit LOC, LOD and LOA requirements. However, LOO was not considered during these optimizations. Instead, it is considered after GA selection by including new locations aimed at improving the overlapping rather than providing an optimal solution. In this section, several tests were carried out to evaluate the LOO using a greedy approach to 1) sample the scenario and 2) narrow it to a few solutions scattered by controlling the spatial density. Accordingly, Figure 11.14 shows the initial overlapping as well as the improvement after requiring an overlap of 20%, 40% and 60%. On the right side of the image, the LiDAR locations were obtained by limiting the sensor range to 1.5 m and 1 m, respectively. Initial solutions were not improved by means of local searches, thus displaying a grid-like pattern. The last experiment pushes even further the overlapping requirements by scattering solutions with a minimum distance of (6 - ϵ) m. Therefore, the minimum number of solutions to reach one another is 2, though it varies according to their distance.

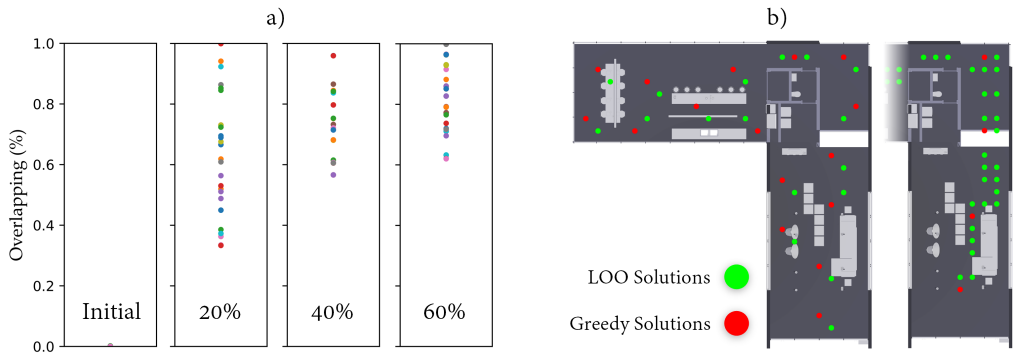


Figure 11.14: a) Initial overlap and results obtained after requiring an overlap of 20%, 40% and 60%. b) Rendering of LiDAR scans selected by a Greedy approach as well as those selected by the proposed method to enhance the overlap.

11.8.4 Response time

The main drawback of local searches is the sequential evaluation of LiDAR scans. Our proposal is conditioned by GPU-based LiDAR evaluations that cannot be solved in parallel. The performance of the GA algorithm was improved by solving the whole simulation of a chromosome at once since it is composed of several locations. However, changes in locations must be addressed individually during a local search. On the other hand, improvements based on pre-calculations are more feasible at the expense of higher CPU-memory usage. Therefore, some optimizations and implementation details are here discussed to provide an efficient solution.

The current LS approach checks the validity and computes the metric values of every new solution, regardless of the possibility that it may have been previously checked. It seldom occurs with random initialization, though it is more likely to happen with uniformly sampled locations due to the constant step length. Therefore, the response time can be decreased by pre-calculating both the validity and metric results in grid voxels whose length depends on the step length of neighbourhood translations. Still, the latency remains the same for a greedy approach.

Figure 11.15 shows the average and global response time for the three LS approaches. The office building scene was used to conduct this evaluation using the configuration in Table 11.1. The first chart shows the average response time per LiDAR location. Hence, latency varies depending on the number of iterations. The global latency comprises the response time from the search and pre-calculation stages. Accordingly, it can be concluded that this approach significantly accelerates the search, except in those algorithms that get easily stuck in local optima or cannot reinitialize.

During the GA, different chromosomes are evaluated synchronously. Parallelism is, however, possible in other stages: initialization, parent selection, crossover, etc. Hence, it was evaluated whether parallelism was effective enough to get some advantage with the pipeline on the GPU. Initially, it was implemented as a multi-threading solution on the CPU. Figure 11.15 shows

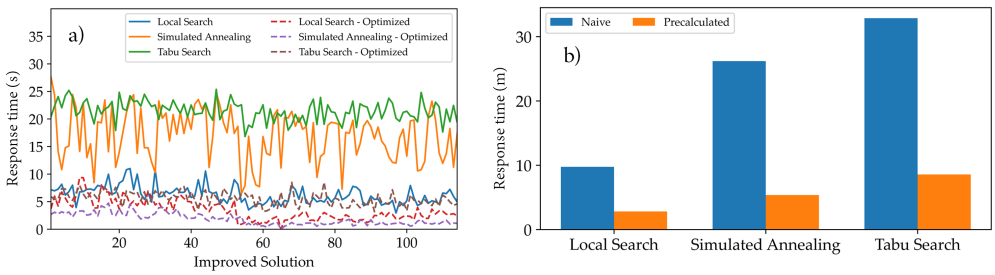


Figure 11.15: a) The average response time per scan for LS algorithms, implemented by following two approaches: naïve and optimized, and b) summation of response time from previous configurations.

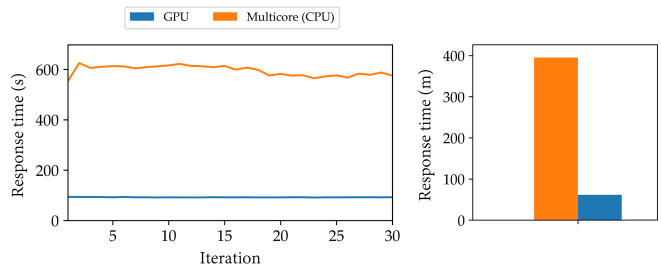


Figure 11.16: Response time of GAs implemented as CPU (multi-core) and GPU solutions. First, the response time is shown per GA population, and finally, the response time is accumulated for each optimization.

that the GPU-based solution is significantly faster since data is processed in the GPU for the whole GA pipeline. Otherwise, data would be transferred to the GPU, evaluated and subsequently downloaded to the CPU again (see Figure 11.16).

11.8.5 Variable scan height

Another contribution of this work is the estimation of the most appropriate scanning height. Figure 11.17 shows the results of F_1 and F_2 metrics by using 1) fixed height or 2) variable height within the range $[1\ \mathrm{m}, 2\ \mathrm{m}]$. In both cases, the scenario was uniformly subsampled every $0.1\ \mathrm{m}$. The first search was performed over a 2D grid, regardless of being applied over a 3D scene, whereas the second one is 3D. The results are obtained using an experimental environment composed of four outer walls and four inner walls. The latter have gaps placed at different heights, as depicted in Figure 11.17. The LiDAR was first configured as a Velodyne HDL-64E, and then, as a Pandar64 with a wider vertical FOV and non-uniform resolution. Instead of LS, a Greedy approach was used to evaluate a fine-grained subdivision of the space that is subsequently narrowed to the top-ten locations. As observed in Figure 11.17, the scans selected by the Greedy approach are placed next to wall holes, thereby allowing to reach more polygons despite being guided by the F_2 metric. Therefore, this shows the inclusion of F_1 in F_2 as well as the capabilities of the proposed pipeline to adapt to different heights.

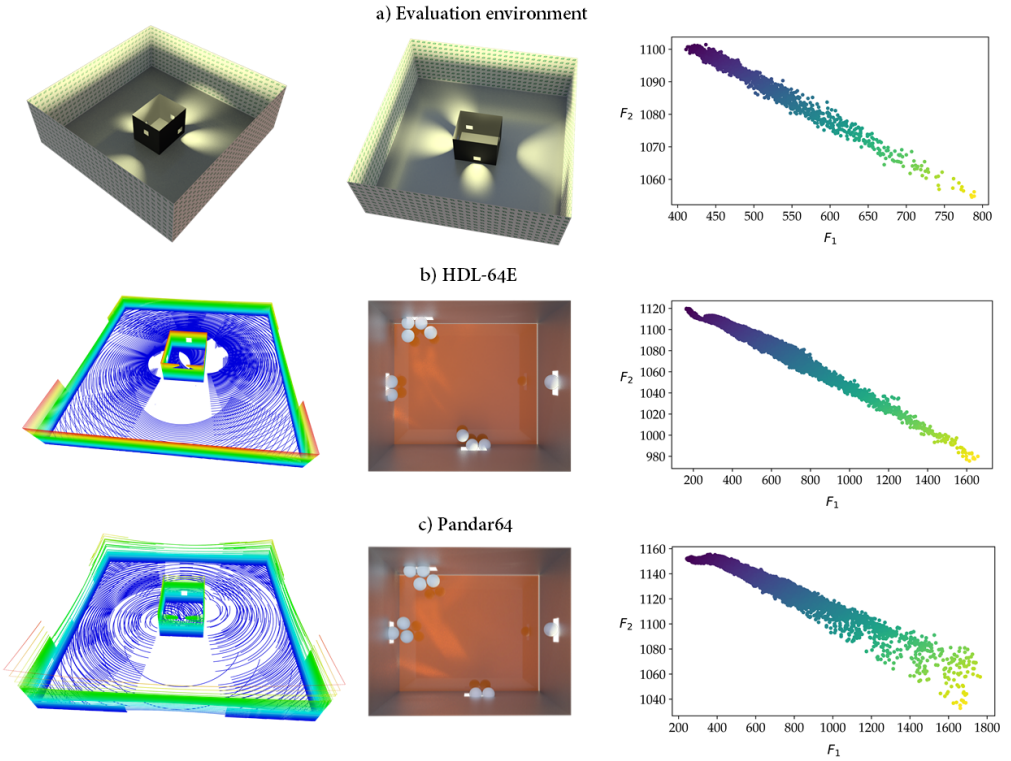


Figure 11.17: a) An experimental environment designed for testing the benefits of variable height, and b, c) locations selected by a Greedy algorithm using HDL-64E and Pandar64 LiDAR sensors. The scatter plots show the values of F_1 and F_2 for each initial solution. The first one shows the results of a LiDAR with a fixed height.

11.9 Conclusions and future work

In this chapter, a P4S methodology was described to select the best n scanning locations in complex 3D environments, taking advantage of GPU-based spatial queries and LiDAR scans. Instead of applying a single metaheuristic algorithm, as in most of the revised work, the pipeline was split into two different phases: local optimizations and selection. The proposed metrics do not rely on user-defined values and were implemented as efficient objective functions. Furthermore, some of these metrics, such as the LOA, were improved by considering several criteria (and metrics) at once. Also, the optimal number of required scans is implicitly obtained with a GA guided by the LOC metric. Finally, algorithms were accelerated with multi-threading in the CPU and GPU (the latter only for GAs). The results showed that the proposed method is capable of constructing a LiDAR planning to scan environments with significant occlusion.

Despite being accelerated in CPU and GPU, the proposed method is time-consuming due to the dimensionality of the input scenarios and the high resolution of LiDAR scans. Therefore, pre-calculating the metric results in uniformly scattered locations ought to be further explored, at the expense

of a larger memory footprint. In this regard, compression can also help to reduce the cited footprint by exploring the spatial similarity of close scans. Finally, TLS planning requires discrete locations, whereas other kinds of sensors, such as mobile or aerial scanning, operate with continuous paths.

Detection of archaeological remains with thermal imaging

12

About this chapter

This chapter describes the automatic reconstruction of thermal products depicting archaeological remains from the so-called Castle of Arenas (Jaén). These results are further analyzed to identify anomalies that may shed some light on remains that are still undiscovered. Despite these being visible on thermal reconstructions, the process was automatized in 3D with a voxelization that facilitated neighbourhood-based operations. The overall procedure of this work is shown in Figure 12.4.

12.1 Historical background of Castle of Arenas . . 227
12.2 On the inspection of archaeological remains228
12.3 Study area 230
12.3.1 Acquisition of ground control points 231
12.4 Reconstruction of RGB point cloud 232
12.5 DEM and orthomosaic 232
12.6 Thermal modelling . . 232
12.7 Anomaly detection . . 233
12.8 Experimental results and discussion 234
12.9 Conclusions and future work 236

12.1 Historical background of Castle of Arenas

The Arabic ruling in Spain started in 711, with the defeat of King Rodrigo in the Guadalete River. It had a huge development that only left a small reduct of Christians in the Northern mountains. From this Christian holdout, a reconquest was started that lasted eight centuries. The battle of Navas de Tolosa (Jaén) in 1212, also known as the battle of Al-Iqāb in Arabic historiography, was a key event for Christian kingdoms as part of the Spanish Reconquest. Most of the nowadays Andalusian lands fell to Christian forces, thus solely leaving the newly created Nasrid Kingdom of Granada, which had continuous changes in its boundaries. They finally surrendered to the Catholic Monarchs in 1492, and from this date, the entire Iberian peninsula was ruled by Christians. During this period, multiple defensive structures were built, changed and augmented, standing out those with a distinguished role for either part of the Reconquest. The Castle of Arenas, which is the main concern in this work, was one of these relevant fortresses. It is located over the top of a hill (Cerro del Castillo) at the end of the Alta Coloma mountain range. Therefore, the fortress has complete control over the valley of the Guadalbullón River. It is a calcareous gorge that forms a natural crossing of the East side of the Subbetica mountain range where several remains of previous ages have been found. Archaeological findings range from an Argaric passage-ritual area known as 'Puerta de Arenas', located in the South, to a Roman pavement in the North.

The Castle of Arenas originated under Arabic rule, though it may be possible that even a Roman fortress existed in this place [424]. However, it was located at the boundaries between the Arabic Kingdom of Granada and the Castillian Kingdom, and therefore, it was part of both of them for centuries. It was first occupied by Fernando III in 1246, with the aim of recovering Jaén, though a few years later it was annexed to the Kingdom of Granada. Again, it was recovered by Alfonso X in 1280 and lost back in

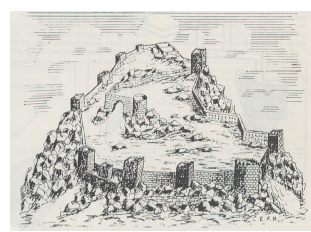


Figure 12.1: Artistic ideation of the Castle of Arenas [424].

1282 as the Castillian Kingdom trembled due to succession conflicts. This sequence was repeated over the next centuries: Alfonso XI pushed back the Arabic army to Granada in 1315, and it was only part of the Castillian Kingdom for four years. In the 14th century, several scuffles were attempted by Miguel Lucas de Iranzo to recover it from the Arabic kingdom. It was not until 1481 that the castle was again mentioned in some brief notes as part of the domains of Jaén.

The remains of the Castle of Arenas comprise a few towers and walls in a very steep mountainous location that makes it very hard to explore. Furthermore, it has been very poorly conserved in contrast to other notable Andalusian fortresses from the Arabic kingdoms. Most of the documentation found about this fortress was collected by Enrique Fernández Hervás in 1986 [424], while more recent explorations have been performed by Francisco Olivares [425]. The former depicted what was believed to be the shape of the castle (see Figure 12.2), where its main inaccuracies will be shown throughout this chapter by showing the aerial view. Although the main shape is inaccurate, some features were accurately represented and the errors were just the aftermath of ground-based observations.

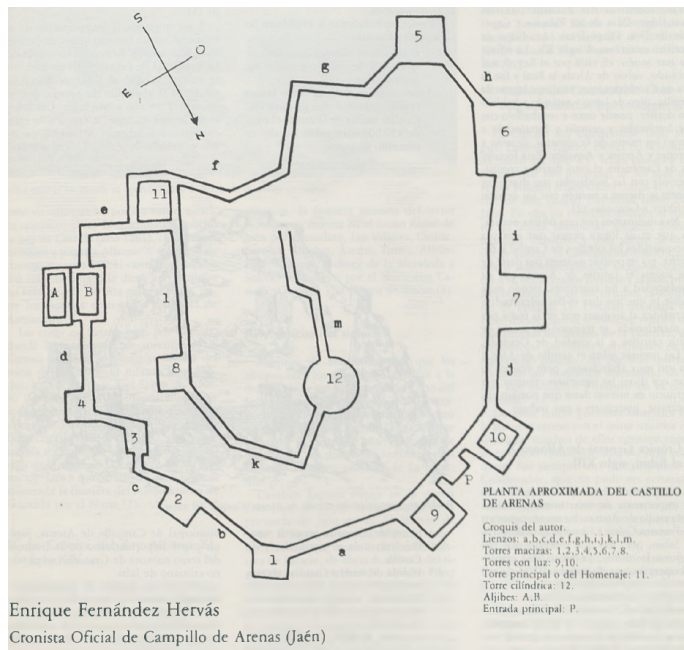


Figure 12.2: Scheme of the structure of the Castle of Arenas, proposed by Enrique Fernández Hervás [424].

12.2 On the inspection of archaeological remains

The preservation of cultural heritage has recently benefited from the use of high-resolution digital cameras that allow the reconstruction of a scenario with high precision. Aerial images acquired from drones not only allow us

to generate 3D models using photogrammetric techniques but also to obtain information beyond the visible wavelengths depending on the coupled sensors. Thermal imagery has been widely explored to find archaeological remains since underlying structures interact with the foreground surface by transferring its temperature, which may be significantly different from the surroundings. From the literature, some of the most outstanding works are authored by Casana et al. [374] and Carmona et al. [377], which successfully attempted to find buried remains of ancient civilizations. The energy transfer of thermal imagers is exhaustively described by Vollmer [33]; however, a briefer summary of what thermal anomalies ought to be like in archaeological remains is explained by Casana et al. [374], as depicted in Figure 12.3. Also, this work showed how temperature varies throughout the day for different surfaces, and therefore, specific hours may be better than others to maximize the thermal contrast).

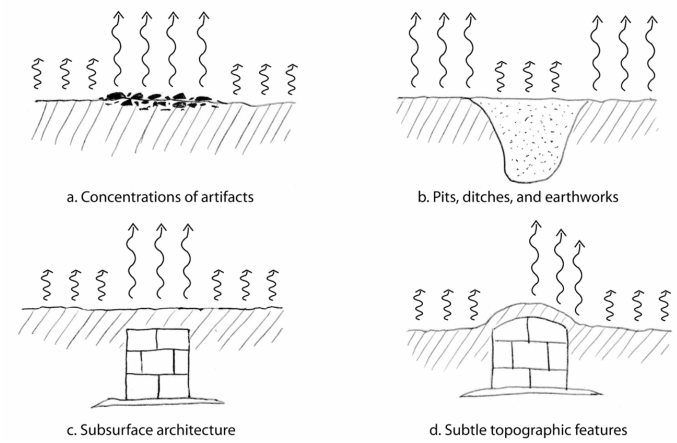


Figure 12.3: Anomalies that could be observed in archaeological remains, according to if these are buried, visible or remains are simply variations in the ground compounds [374].

In contrast to thermal imagery, visible sensors acquire images of higher resolution, enabling the estimation of larger and more dense point clouds. Accordingly, RGB point clouds can be used as a constrained data source to map thermographic information, thus taking advantage of the spatial characteristics. For that purpose, the differences between visible and thermal imagery were estimated using ECC, as explained in a previous chapter. The 3D thermal point cloud was estimated over an RGB point cloud, using the previously computed matrices that allow projecting thermal imagery into RGB data. Finally, a large number of 3D RGB points were projected into RGB (and thermal) imagery using the camera matrices estimated during the photogrammetric reconstruction.

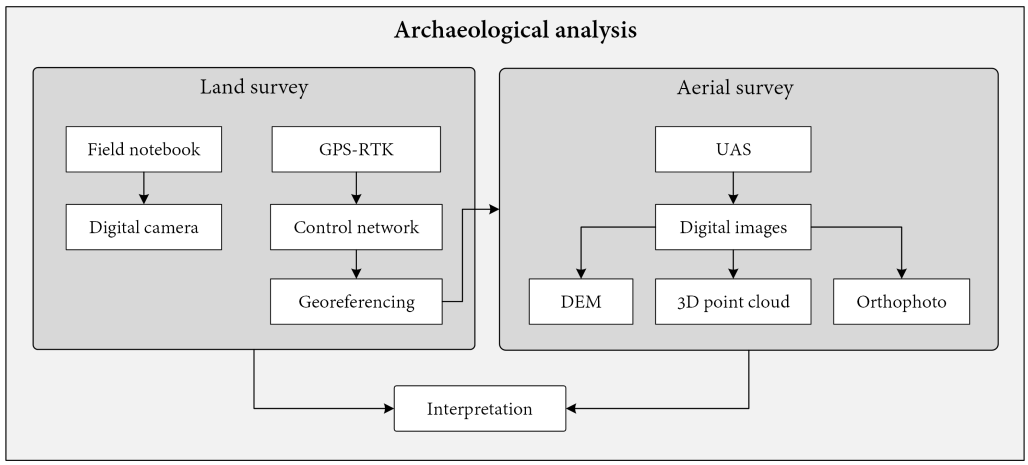


Figure 12.4: Overview of the proposed methodology. The archaeological remains are monitored through land and aerial surveying. The first is simply used to help with the analysis from the latter survey, whereas the UAS results provide a better insight into details that could not be observed from land.

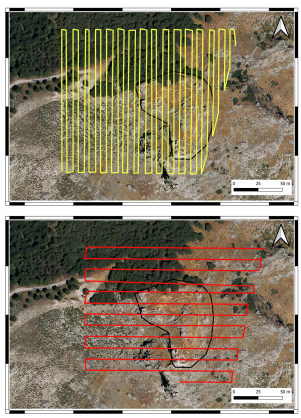


Figure 12.5: Path from two UAS flights performed with different forward directions to capture features from the fortress walls.

12.3 Study area

The Castle of Arenas is located over a hill called Cerro del Castillo, 39 km at the South of Jaén. The geographic coordinates are 37°35′34″ N and 3°37′20″ WG following the ETRS89 reference system. The altitude is approximately 1370 m above sea level, with the summit placed at 1393 m, and the lower wall at 1360 m. The surveyed area covers about 0.6 ha depicted in Figure 12.6. Four flights were performed according to the limited flight time (~90 min), including one *nadir* and three oblique flights. The latter three were performed to observe wall features. Flights were carried out at an altitude of 60 m above ground level for the first one, and 50 m for the rest. The GSD of the collected imagery was 2.1 cm/pixel and 6.0 cm/pixel for visible and thermal imagery, respectively. The flight was conducted close to midday in order to minimize shadows from the remains. This period was preferred over the period between sunset and sunrise due to safety reasons, despite the latter being more optimal for thermal surveys. The collected temperature for each flight is summarized in Figure 12.7.

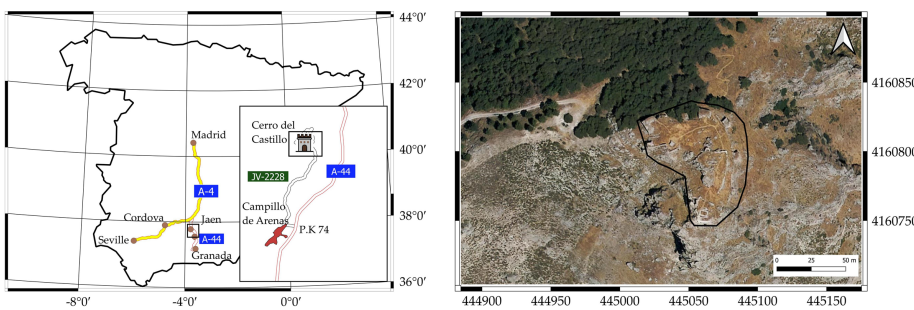


Figure 12.6: Overview of the research area. The right image shows the surveyed area, with the archaeological remains highlighted as a black polygon. The coordinates are in the UTM reference system (m), zone 30, ETRS89.

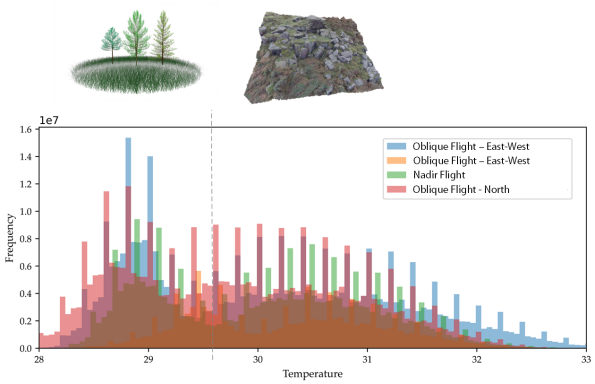


Figure 12.7: Histogram of radiometrically corrected thermal samples as collected for every flight. Lower temperature values may correspond to vegetation, whereas the centre is mainly linked to rock masses, either from archaeological features or not. Higher temperature is linked to anomalies and heated rock masses.

12.3.1 Acquisition of ground control points

The first step in the acquisition of information is to obtain the coordinates of GCPs, since they help to improve the accuracy and quality of later results. These coordinates were acquired using a couple of Topcon GR5 RTK-GNSS in RTK mode. The reference station was placed in the trigonometrical station of Orozco, which belongs to the REGENTE network, placed at 1120 m from our study area. According to the manufacturer specifications, the precision of East and North components is approximately 0.006 m (5 mm ± 0.5 mm), whereas the precision of the $\vec{up}$ component is about 0.011 m (10 mm ± 0.8 mm). Hence, the accuracy of these measurements was in the order of cm. The placement of the GCPs was carefully designed to obtain results georeferenced in a national grid and height datum. From now on, coordinates will be displayed in meters (m), map projections will be depicted in the UTM system, the reference system will be the ETRS89 and altitudes will be calculated according to the mean sea level in Alicante, Spain. Six GCPs were more likely to be enough due to the small size of the study area (~0.6 ha^2), although ten were finally acquired to capture the details of the steep orography. From these, only seven were used to georeference the images, whereas the rest were used as control points. Two of them were located at the side of the hill (1350 m).

The measured GCPs allowed us to evaluate the accuracy of a 3D model in terms of scale. This was checked using the RMSE for the X, Y and Z axes, as defined in Equation (12.1). The acquisition of imagery and GCPs are the baseline of our methodology. From here, 3D results are estimated to provide a better data source on which later visualization and analysis tasks are easier to perform by human operators.

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left|P_i - P_{i_{GNSS}}\right|^2} \tag{12.1}$$

where $P_{i_{GNSS}}$ is the location of a GCP, P_i is the i-th point from a point cloud and n is the number of pairs of points (GCP and reference point). Table 12.1 shows an overview of the acquired GCPs along with the estimated error and the RMSE concerning control points and checkpoints.

12.4 Reconstruction of RGB point cloud

The estimation of RGB point clouds was conducted with the SfM algorithm provided by Agisoft Metashape®(version 1.7.5). The 3D modelling of archaeological sites is usually carried out to evaluate them using a computer-based model, instead of the real-world one. In this regard, UASs are a major breakthrough technology for increasing the knowledge of ancient civilizations. Some of the conditions taken into account, given that this work is focused on archaeology and thermal imaging, were to collect the data with sufficient light and to avoid a high downscaling during SfM to ensure a high LOD and preserve most of the features.

12.5 DEM and orthomosaic

The following step is to generate the DEM with high quality, as the one depicted in Figure 12.8. It also involves extracting the analytical hill shading, derivatives of hill-shading from different directions (range of hill shadings, mean of hill shadings and PCA of hill shadings), elevation differentiation, trend removal, slope severity, sky-view factor, solar insolation modelling, as well as composite images using the normalized DSM and the shaded relief. The objective is to interpret the topography by means of visual features and slope variations, i.e., using a geomorphometric approach guided by the ground surface.

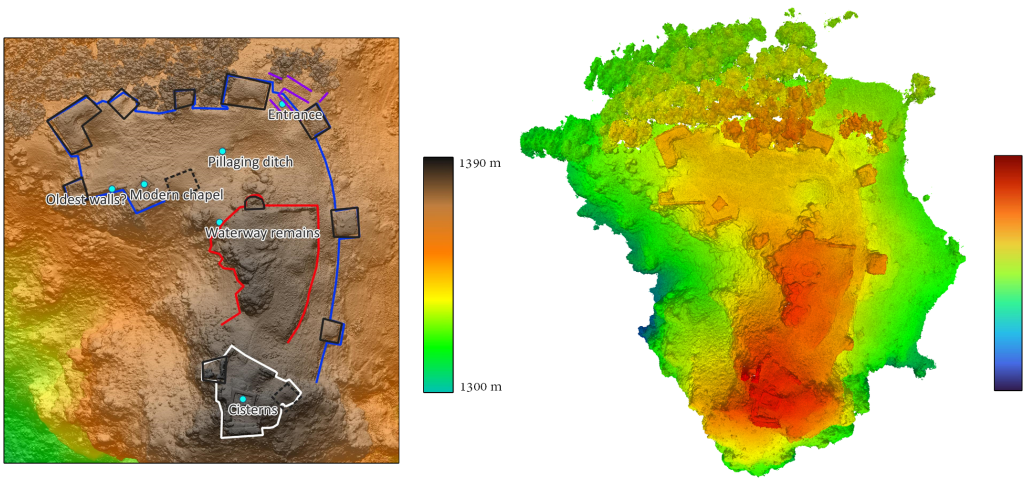


Figure 12.8: DEM of the fortress along with a point cloud shaded according to the relative height.

12.6 Thermal modelling

The 3D thermal modelling of the surveyed environment is performed using the methodology described in Chapter 6. Briefly, thermographic data was

projected into a dense RGB point cloud of nearly 100 million points. Firstly, visible and infrared imagery were geometrically corrected and aligned to estimate the homography matrices that enable overlapping them. Then, camera matrices estimated during SfM were used to project 3D RGB points into RGB imagery, and subsequently, to thermal imagery. The resulting thermal point clouds accumulate two layers: 1) grayscale infrared values, and 2) radiometrically corrected temperature, ranging from 28 °C to 33 °C. The latter is represented as an absolute value, in comparison with the greyscale representation.

12.7 Anomaly detection

Despite outlier areas being visible over the thermal point cloud, their visualization can be enhanced by highlighting relevant features from the rest of the environment. This fusion of thermal and RGB colours was already explained previously; the latter is fused with the first according to $w(t)$, which weighs how anomalous a sample, t, is. Thus, values are classified as anomalies whether they lie outside the interquartile range. Figure 12.9 shows the rendering of the reconstructed thermal point cloud by combining both RGB data and temperature to show outliers. Both of them highlight areas that are suspected to cover yet unknown remains of two different towers.

These outliers can also be highlighted in 3D using infrared data from the surroundings. The main challenge is to efficiently find the neighbourhood. Although the benefits of using Morton encoding and sorting have been shown throughout this dissertation, it does not handle well spatial queries based on a radius since the notion of space is discarded at the expense of efficiently solving operations in a 1D buffer. Therefore, other trivial data structures, such as the regular grid in Figure 12.10, seem more appropriate to solve neighbourhood searches with a complexity of $\mathcal{O}(1)$ (or $\mathcal{O}(n)$ whether a list of primitives is attached to every voxel). Hence, the thermal point cloud is discretized using voxels of a fixed size. It can be efficiently computed in the GPU since points can be easily indexed into a voxel, whereas the memory footprint is parameterized by the voxel size.

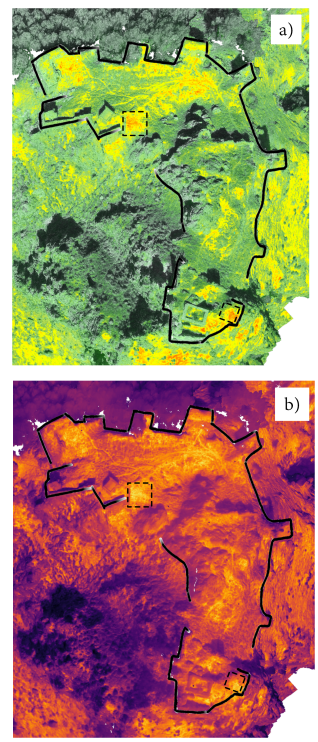


Figure 12.9: a) Thermal point cloud and b) fused rendering of RGB and thermal point cloud to highlight statistical anomalies.

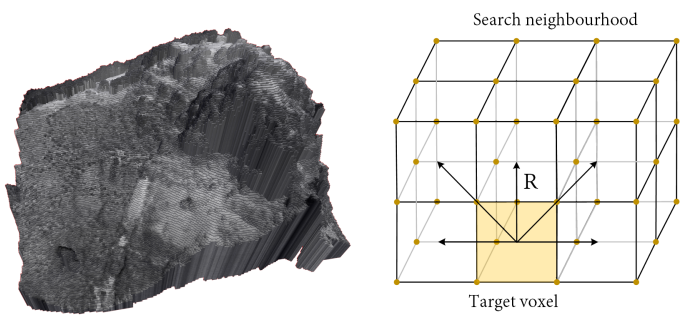
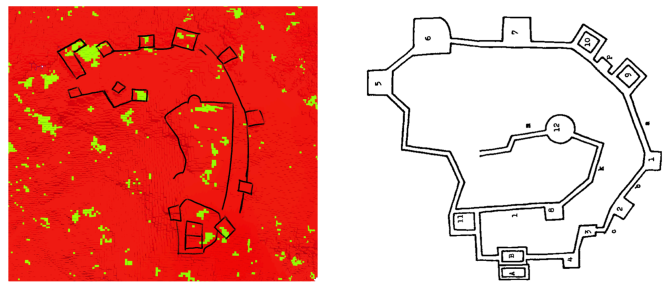


Figure 12.10: Overview of the anomaly detection procedure. a) Voxelized environment, and b) neighbourhood of a voxel at a distance of r.

Following this approach, the point cloud was voxelized using a variable number of subdivisions, with each cell storing the average temperature of the included subset of points. Those voxels whose temperature is notably different from their surroundings are highlighted in the rendering. Accordingly, a voxel is marked as an outlier whether the temperature difference is above $\kappa \cdot \sigma$, provided that σ represents the standard deviation of a neighbourhood, and κ is a user-defined factor that can be interactively changed. Figure 12.11 shows the thermal anomalies of a 3D voxelization of size $300 \times 150 \times 300$, with a neighbourhood of size 15 and a threshold of 2.2 °C.



a) b)

Figure 12.11: Comparison of a) anomalies detected from a 3D voxelized neighbourhood, depicting the underlying structure, and b) the enhanced structure proposed Fernández Hervás [424].

12.8 Experimental results and discussion

This section is aimed at emphasizing some of the results that have been extracted from the aerial views. The following figures confirm some of the few conclusions drawn by previous literature circa 1990. The steep orography sculpted the shape of the fortress, fitting it to the contour of the hill. The castle has an 'L' shape ground plan where the short limb extends from the North to the West and the longest extends from North to South (see Figure 12.12). This configuration presents three defensive lines in the inner shape.

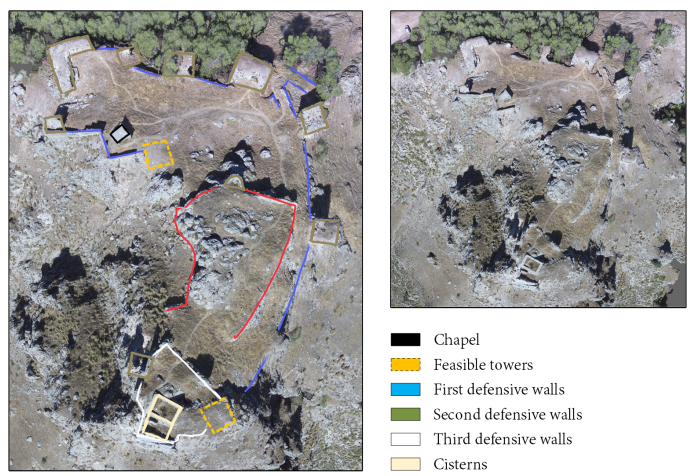


Figure 12.12: Aerial view with highlighted archaeological remains, including both visible shapes and those identified with thermal imagery (dotted).

The first defensive line is composed of the outermost walls, including the short limb as well as the boundaries of the longest. Therefore, it faces the cliff despite not reaching the southernmost part. On the other hand, the entrance located at the northernmost area of the outer wall is very poorly conserved. From the ground, there is only a clearly visible passage crossing two walls at an oblique angle, and it can also be observed in Figure 12.13. Additionally, the aerial views facilitate the finding of further remains that ought to be considered, despite going unnoticed in previous work.

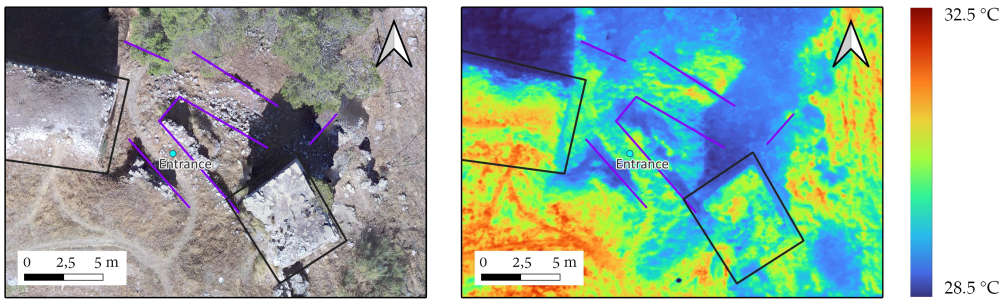


Figure 12.13: Fortress entrance in RGB and thermal orthomosaics.

The second defensive line presents another kind of tower and walls. This area is observed to be surrounded by the first defensive line on the North and East side. It faces the cliff on the West side as well as the third line on the South side. Only a tower seems to belong to this line: a semicircular tower built with big uncut stones. Note that cylindric towers were not usual in Arabic castles, and therefore it might be a construction of the later Christian settlement. Also, there seems to remain a water canalization system at the bottom of the tower, as depicted in Figure 12.14.

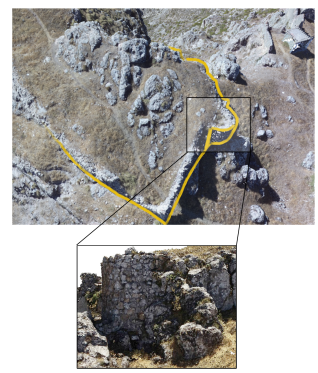


Figure 12.14: Water canalization system highlighted in the RGB point cloud as well as its ground view.

Between these defensive walls, there exists a flat space appropriate for the construction of smaller buildings. These are nowadays inaccessible due to the remains of the collapsed structures and sedimentation, and therefore, further surveys are required to provide better insight. In the thermal imagery, this zone is warmer than its surrounding area. Along with this, a more recent chapel was constructed at the end of the 19th century and rebuilt in recent years. In the long arm of the castle, right in front of the entrance, there is a huge pit with a width of 13 m and 4 m deep, which is supposed to be the result of plunder sometime between the abandonment and the declaration of the castle as a protected archaeological site.

A third defensive line is composed of the walls in the southernmost area. It is also the most secure part of the fortress as it controls every side of the cliff but the North. It may be intended to join the middle walls to create a structure surrounded by several layers. The DEM in Figure 12.8 shows an anomalous geometry at the East side, with a similar size and shape to the West tower. However, it has steeper and more intricate access. The thermal imagery shows a noticeable anomaly in the same spot, as shown in Figure 12.15. With a minor ground survey, it was observed to be what seems like the basement of a tower. The fact that only the layout remains could be

due to the strong winds that take place at the highest point of the hill. The existence of this tower may have passed unnoticed without aerial imagery. This area presents slopes near 50% in the entrance passage, which along with its notable altitude [426], makes this site hard to access and explore with ground-based surveys, such as TLS. The rest of the study area has a similar slope since it is located on a cliff. Therefore, UAS together with oblique flights has helped to collect images of every façade and observe subtle details.

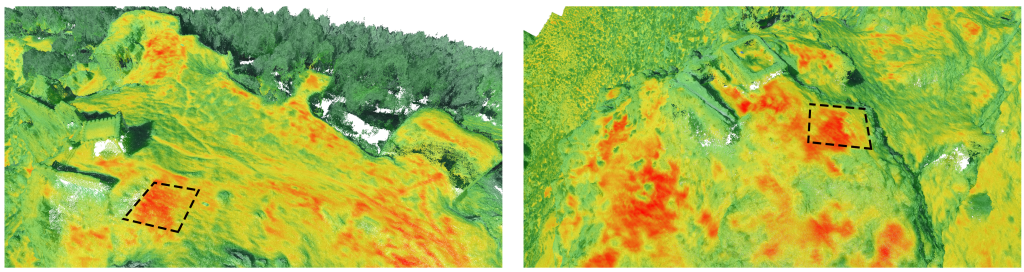


Figure 12.15: Rendering of thermal anomalies over two possible decimated towers.

Finally, there is another element that caught our attention: the water supply system. The orthomosaic shows two well-defined cisterns at the Southern part of the fortress, depicted as two contiguous rectangular shapes. There is also a small canalization that comes out of the middle wall, under the semicircular tower. It is similar to a small tunnel of a few tens of cm, though it follows a coarse canal made of rock in the first area and ends up at the cliff on the South side of the castle. Furthermore, previous work proposed the South side of the fortress as the location of two Arabic cisterns which may be the destination of the water flow.

12.9 Conclusions and future work

The proposed procedure has led to the reconstruction of a 3D model with visible and infrared data, which can be later integrated into GIS and BIM projects, and can help archaeologists in the planning of excavations. Previous work addressing the fortress shape was performed with ground-based observations, whereas the extracted aerial view helps to develop a more precise idea of what the fortress looked like. The final products are represented in meters, thereby enabling the measurement of the dimensions of specific features, the distance among them, etc. Not only the visible remains were connected to provide a feasible structure, but also other abnormal areas were highlighted. These should be further investigated in future archaeological excavations. Furthermore, the objective after UAS, photogrammetry and thermal reconstructions is not only to shed light on ancient settlements but also to build a framework of preventive archaeology. Therefore, it helps in the preservation of cultural heritage and landscaping resources.

Thermal imagery has been proven useful for detecting which areas should be excavated first to clarify how was the original structure. The 3D infrared

reconstruction was performed over a dense RGB point cloud, thus allowing us to visualize the study area with higher LOD. For instance, it helps to discard some anomalies caused by shadows. A semi-supervised automatic detection of anomalies was implemented over a 3D voxelization of the thermal point cloud. To this end, the temperature of each voxel was compared to its surroundings. Using this approach, some notable anomalies were detected near what is suspected to be two missing towers in the North and South area.

In future work, more surveys should be conducted, even in night conditions, attending to the variations reflected in Figure 3.20. Also, a wider number of sensors could be used to provide more information, especially on those areas that are suspected to cover ancient remains. In this regard, magnetometry may shed some light to extract much more informed conclusions. Ground-based magnetometry may not be efficient due to the irregular and steep orography, whereas aerial magnetometry is more suitable for this case study. Even LiDAR technology could be useful for obtaining surface microreliefs and therefore, spot places with buried remains. On the other hand, conventional methods, such as terrestrial photogrammetry, should only be used to provide complementary data sources with a higher LOD in specific areas.

Additional data

Table 12.1: UTM coordinates (m) of GCPs, as well as their East, North and height error in centimetres (δ_E, δ_N, δ_H).

1) Control points							
Id	East (m)	North (m)	Height (m)	δ_E(cm)	δ_N(cm)	δ_H(cm)	Total δ (cm)
1	444958.693	4160812.226	1342.189	-0.4	0.5	2.3	2.4
2	444977.401	4160840.587	1341.810	-1.5	-0.5	1.3	2.0
3	445095.169	4160859.578	1357.320	-0.2	0.4	0.6	0.7
4	445073.400	4160815.904	1374.452	2.0	0.2	1.7	2.6
5	445029.280	4160821.232	1375.903	1.2	-1.5	-2.2	2.9
7	445063.855	4160795.858	1383.090	1.1	0.2	0.3	1.2
9	445059.726	4160754.837	1389.440	-1.9	0.8	-0.2	2.1
		Absolute mean		**1.18**	**0.58**	**1.22**	**1.98**
2) Checkpoints							
Id	E (m)	N (m)	H (m)	δ_E(cm)	δ_N(cm)	δ_H(cm)	Total δ (cm)
6	445049.057	4160807.002	1376.097	6.0	-9.4	3.8	11.8
8	445072.237	4160775.253	1383.474	-1.6	2.0	8.0	8.4
10	445054.274	4160800.568	1378.135	2.9	3.8	9.9	11.0
		Absolute mean		**3.5**	**5.06**	**7.23**	**10.4**

Classification of grapevines using UAS-based hypercubes 13

About this chapter

This chapter comprises the classification of hyperspectral datasets depicting eight varieties of red grapevine and another nine varieties of white grapevine. This is accomplished using feature reduction and Deep Learning to learn how these features correlate to identify the correct variety. The spectrometer data is first analyzed to support the use of Deep Learning networks, as different varieties show similar signatures. These also happen to be similar to the conventional hyperspectral profile from vegetation. Then, a Convolutional Neural Network, also called CNN, is designed to solve this problem. It is enhanced by enhancing the features with spatial-attention layers, whereas the extraction of deeper features is based on the widespread Inception block. The obtained results are analyzed in terms of capacity, response time, separability and capabilities to work over other hyperspectral datasets. Unlike previous work, the proposed network will be proven to be very useful on several datasets, even satellite-based. The tuning of the network is also addressed regarding the dimensionality of input data and ablation tests. The latter decimates the network, or changes some of its layers, to show the proposed structure performs better than any of its variants. More importantly, the network solves the classification of grapevine varieties, which is the main objective of this chapter.

The source code is available at https://github.com/AlfonsoLRz/VineyardUAVClassification .

13.1 On the classification of grapevine varieties . . 239
13.2 Spectrometer analysis 240
13.3 Processing of UAS data 241
13.4 Automated training . . 243
13.4.1 Dataset 243
13.4.2 Implementation 244
13.4.3 Architecture and training 246
13.4.4 Data sampling and regularization 249
13.5 Experimentation and analysis 250
13.5.1 Study area 251
13.5.2 Material 252
13.5.3 Impact of window size . 252
13.5.4 Ablation study 252
13.5.5 Training time and network capacity 254
13.5.6 Separability 256
13.5.7 Confusion matrices . . . 256
13.5.8 Analysis of errors 256
13.5.9 Training over satellite imagery 257
13.6 Conclusions and future work 258

13.1 On the classification of grapevine varieties

Previous work on the classification of vineyard varieties consists of data correction and ML methods using close-range sensing tools. Still, only a few studies have addressed this problem, and the most complete investigation was carried out by Gutierrez et al. [342]. In this work, samples were manually selected by averaging the signature of every variety and filtering out those with a low correlation to such a signature. SVM and MLP were then trained with k-fold to distinguish thirty varieties, with MLP obtaining better results. Another field concerning grapevine classification is the binary segmentation of soil and vegetation; it covers the majority of literature on classification applied to vineyards, together with the detection of diseases and plagues. Furthermore, the classification of hyperspectral data is a hot research topic albeit not being applied to grapevines. From these, only the work of Nezami et al. [359] was found to operate over UAS-based data, with

the purpose of classifying three different tree species using hyperspectral and visible imaging, as well as canopy height models. This work reported an overall accuracy below 95%.

Most of the research on the classification of hyperspectral data is tested over standard satellite-based hyperspectral datasets [360]. These present different materials that notably vary from one hypercube to another; hence, current state-of-the-art methods are trained over individual hypercubes with 2D and 3D CNNs. They typically use very deep and wide CNNs [361] that enhance the extracted features at one of several points, either with spatial-attention [364, 365] or wavelets transformation [362]. All these networks conclude that spatial and spectral features are relevant to the classification of hyperspectral data.

The input data have also a notable impact on the proposed network. Hence, it must be carefully corrected to provide reliable data that can be later classified. In this regard, there is a lot of work addressing both feature transformation and reduction, regardless of previous radiometric corrections. For instance, Gutiérrez et al. [342] addressed the inherent noise in UAS-based data using a combination of Standard Normal Variate (SNV) and de-trending to remove the scatter effect. Then, they smoothed the hyperspectral signature by applying Savitzky-Golay filtering with different step sizes over the first and second derivatives. However, only considering this work is not representative enough of current state-of-the-art techniques. Other widely used methods go from NNMF to PCA, LDA, FA or ICA, together with methodologies that simply filter out those features that have a lesser impact on the sample label, without transforming the feature space. Remark that feature transformation does not imply simplifying the feature space; instead, a more meaningful representation is extracted to help later classification methods.

13.2 Spectrometer analysis

Data acquired from the spectrometer was analyzed to evaluate the distance among signatures of different grapevine varieties. First, some of the collected samples were discarded as they presented an anomalous signature in comparison with the average signature. In contrast to UAS's data, material unmixing is not required here since leaf samples are obtained in the laboratory from single points. Figure 13.1 illustrates the processed spectrometer data and the features that provided the largest variance among different classes are highlighted with dotted lines. Then, multiple selection criteria were applied to narrow the spectrum and transform the multi-dimensional feature space. Feature transformation was performed with PCA, LDA and UMAP (Uniform Manifold Approximation and Projection for Dimension Reduction), with the latter two providing the best clustering. The three features that contributed the most in terms of variance were represented as x, y and z. To this end, n groups were sought for n different vineyard varieties. The results show that differentiation is not trivial with 1D methods and thus requires 2D and 3D-based approaches in seek of hidden patterns

that may shed light on the classification problem.

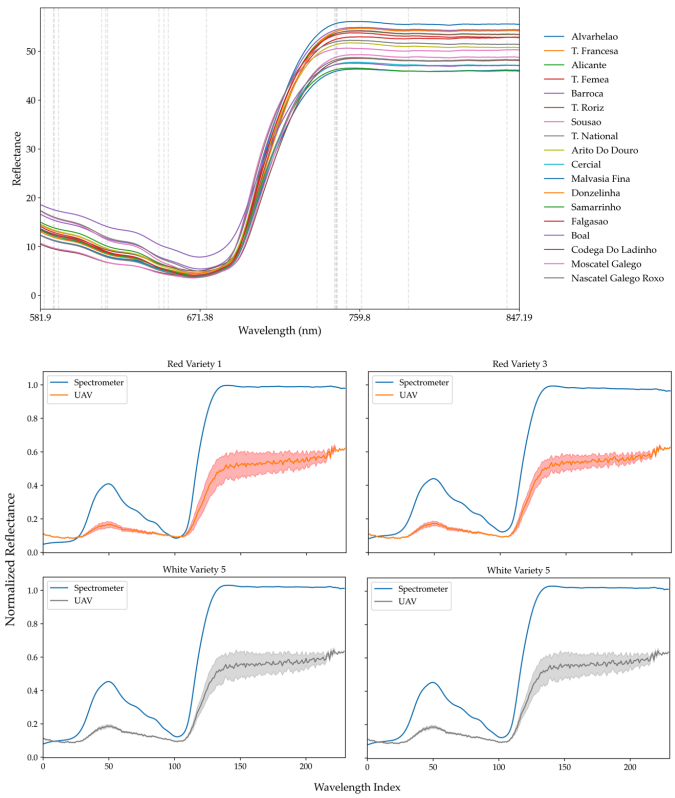


Figure 13.1: At the top, the average spectrometer signature of every vineyard variety. At the bottom, there is a comparison of the normalized spectrometer and UAS data. Variance is upscaled by a factor of ten for enhancing the visualization.

13.3 Processing of UAS data

This section briefly describes the process of obtaining reflectance data from raw hyperspectral imagery, which was already illustrated in Figure 4.16. The computed reflectance for every grapevine class is depicted in Figure 13.14, which is following transformed and standardized to feed a DL network.

The hyperspectral data was collected using a drone and processed using Headwall SpectralView™ software. Several swaths were captured for each study area (we refer the reader to Figure 13.15), and a white sample was marked from the white area in the grayscale tarp. On the other hand, a dark reference was obtained by collecting a hyperspectral sample with the lens cap on before the flight. The sensor exposure and frame period were adjusted before the flight by pointing at a bright reference to avoid clamping samples from white surfaces. The white and dark references were then used to convert the raw data to reflectance. From here, the ortho-rectified swaths were obtained using high-resolution DEMs (25 m) from Copernicus' observation program [19] as well as the drone's GPS and IMU data. However, non-ortho-rectified swaths were used for the analyses presented in this

paper to operate over smaller images and avoid distorting the hyperspectral samples.

In contrast to laboratory data, spectral signatures of hyperspectral swaths' pixels are noisier and fuse several surfaces. Therefore, the classification of these data ought to work under different background surfaces, including soil, low-vegetation and other man-made structures. As proposed in recent work, the material unmixing could be performed with nonnegative matrix factorization (NNMF). To this end, hyperspectral swaths in reflectance units could be flattened to 1D ($n \longleftarrow h \cdot w$) with a dimensionality of $n \times m$, where m is the number of features. Then, this flattened vector could be transformed into weight ($W_{n\times c}$) and component matrices ($C_{c\times m}$), where c is the number of target surfaces (end-members).

However, the number of materials visible in a single image (or vineyard varieties) is not known in nature. Furthermore, reducing all the varieties into a single vegetation signature leads to poor classification performance, despite W_i being different for each sample. Finally, the signatures observed in the spectrometer could be used to initialize C, though it leads to the previous unwanted scenario. Therefore, material unmixing in nature is rather suited for the classification of significantly different signatures rather than performing fine-grained classification. However, the feature space can be transformed and narrowed to a few more representative features, instead of unmixing materials. Methods such as Factor Analysis (FA) are also aimed at decomposing A into $W \times C + \Theta$ without the non-negative restriction, with Θ being the measurement error [427]. In this regard, FA was shown to perform better than other feature reduction algorithms (e.g., PCA, LDA and Singular Value Decomposition (SVD)).

According to this, hyperspectral samples were processed by 1) splitting train and test samples from the beginning, 2) fitting the FA and standardization solely with training samples and 3) transforming training and test samples with both fitted models. Standardising is aimed at removing the mean and scale reflectance to the unit variance. With this approach, the range of input HSI values is restricted in the CNN despite starting values being expected to vary as a result of different sensor exposure, frame period and environmental conditions among distinct flights. On the other hand, HSI noise is tackled by splitting the hyperspectral swaths into patches aimed at classifying pixels using their neighbourhood. Hence, pixels are not significant enough by themselves and aggregations learnt by kernels help to alleviate the noise.

13.4 Automated training

The classification of vineyard varieties with UAS data can be hardly approached with 1D algorithms due to the high similarity of spectral signatures. In this section, a method based on Deep Learning is described to classify 3D hyperspectral patches. Through this section, the proposed method is tuned to achieve a high generalization performance.

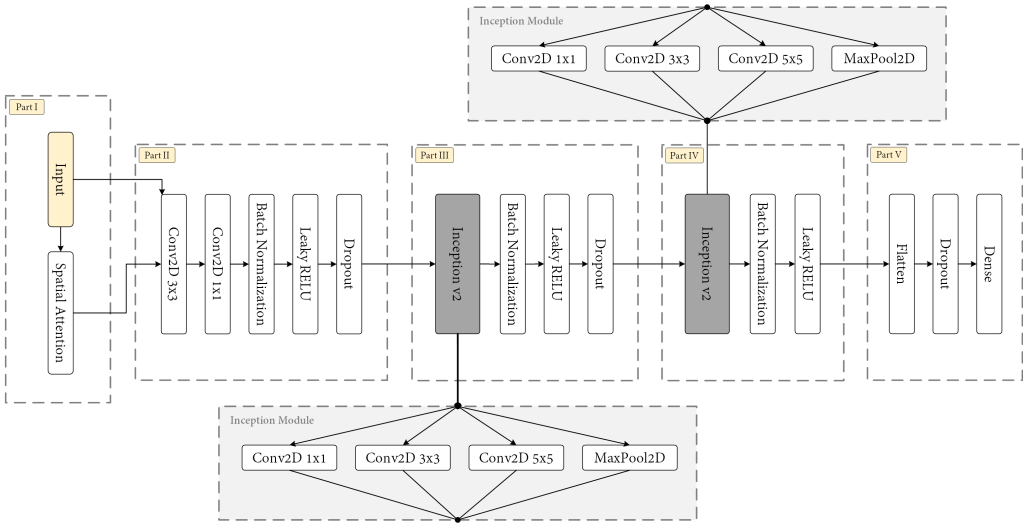


Figure 13.2: Overview of the proposed network architecture, with the input being concatenated to the output of a spatial-attention layer. Then, spectral and spatial features are extracted in Part II. Both Part III and IV comprise an Inception block with frequent follow-up layers. Finally, the resulting features are transformed to obtain the predicted label.

13.4.1 Dataset

The training data is composed of reflectance swaths which are not orthorectified, thus diminishing the required memory storage. The Normalized Difference Vegetation Index (NDVI) was then extracted to highlight vegetation in contrast to the ground, and subsequently thresholded to create a binary mask from each swath. These masks were annotated with Sensarea [428] by marking each row with a different polygon and colour, according to the variety. Different polygons were labelled with the same colours since some varieties appear in several rows. According to this, Table 13.5 shows the number of collected samples for each variety, including spectrometer and UAS data.

Then, HSI cubes and masks were split into 3D patches whose size is a matter of discussion in the later experimentation. The window size used in this study is 23×23 by default, whereas previous work has used patches whose x and y dimensions range from 7 [364] to 64 [362]. Only Liu et al. [358] reported patches of much larger dimensionality (200×200). The larger the patch, the deeper can be the network, though it also increases the number of trainable weights, the training time and the amount of data

to be transferred into/from the GPU. Configurations using larger patch sizes are more suited to images with notable spatial features, whereas ours ought to be primarily discerned through spectral features.

Instead of inputting the label of every patch's pixel, they were reduced to a single label corresponding to the centre of an odd-sized patch. Thus, the classification was performed per pixel rather than an overall semantic segmentation. Before being split, pixels were processed to improve the network's performance by reducing the spectral dimensionality and shifting the data distribution to have zero mean and unit variance. Regarding feature reduction, spectral bands were transformed and narrowed to 30 features.

The processing is performed so that it emulates a real-world application. Training and test pixels were first selected through their indices; then, the feature reduction and standard scaler models were fit using only the training dataset. Once adjusted, both training and validation datasets were transformed with both learnt models. None of the fitted models requires the pixel's labels and thus are very convenient for their application in new and unlabelled areas. The resulting dataset is composed of 291,852 and 568,403 patches for red and white varieties, which were split into training (70%), validation (10%) and test (20%) subsets. With this partitioning, a total of 204k samples are used for training on red varieties, whereas 397k are applied to white variety classification.

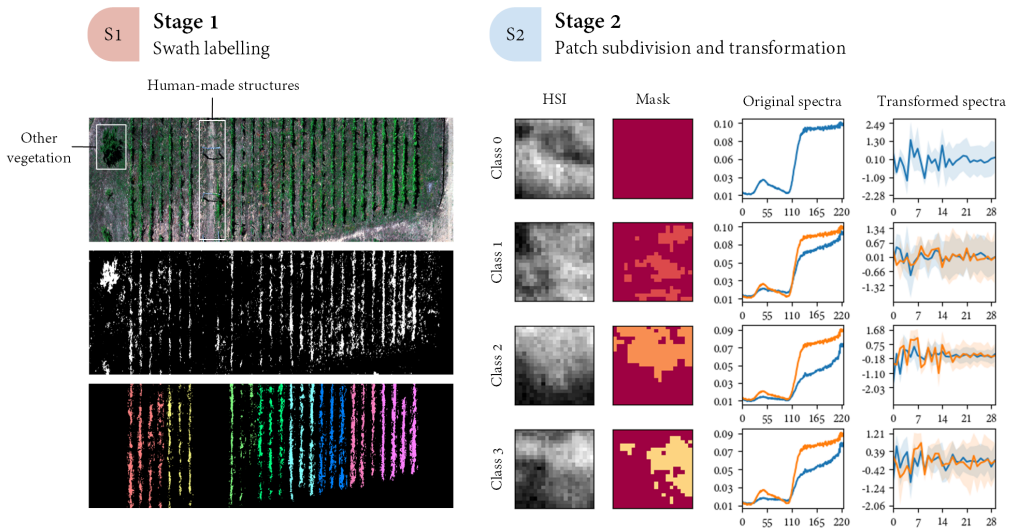


Figure 13.3: Overview of dataset preparation. First, a binary mask was generated using the NDVI and rows were organized into different groups to distinguish vineyard classes. Once pixels are processed as described in Section 13.3, both reflectance and labels were split into patches. The signatures on the right side show the original and transformed reflectance, including the variance per feature.

13.4.2 Implementation

In order to make this paper self-contained, a brief introduction about DL is detailed in this section. Firstly, Deep Learning refers to layered representations that evolve in a learning process where they are exposed to input

data. Typically, the depth of these models is large enough to automatically transform data and learn meaningful representations of data. Despite these models being able to work over any kind of structured data, even 1D, it is here proposed that one's pixel neighbourhood may help with the phenotyping problem. CNNs have achieved remarkably good results in the computer vision field. Convolutions are designed to learn local features, rather than global, by applying element-wise transformations while sliding over input data. These are defined as rank-3 tensors defined by width, height and depth. The width and height determine how large is the neighbourhood of every element, whereas the depth is the number of different learned filters. Hence, a single filter is applied element-wise to compose a response map from input data, whereas the whole filter stack is known as a feature map. If several convolutions are concatenated, these evolve from learning low-level details (e.g., edges) to high-level concepts. Since individual filters are applied element-wise, the learnt patterns are invariant to the position within the image [429]. However, kernels may not be applied for every element. Instead, information can be compressed using steps greater than one, also known as the stride value. Another key concept in CNN is that not every node is connected, thus partially tackling the overfitting problem. Training and test errors ought to remain similar during training, which implies that the network is not learning the training data (overfitting) or generalizing excessively (underfitting). To avoid both situations, the capacity of the model must be tuned in terms of complexity to generalize and reach low training error.

Trainable CNN layers are typically defined by a matrix of weights and biases applied over input data, $f(x; w, b)$, with f being an activation function that allows solving non-linear problems. In this work, Rectified Linear Unit (ReLU) and Leaky ReLU have been applied to tackle the vanishing gradient problem, together with batch standardization. The latter operations work similarly to the standardizer applied as a preprocessing stage. On the other hand, w and b are updated during training with the objective of minimizing a loss function comparing ground truth and predicted values for supervised classification. This is done by an optimizer that changes these trainable parameters using the error gradient scaled by the learning rate, η. The greater η, the faster is achieved the convergence, though it can also lead to significant oscillations. This process is known as Gradient Descent (GD) [430]. As faster convergence may be necessary at the beginning, the decay and momentum concepts were introduced to downscale η during training, thereby omitting abrupt changes.

Besides convolutions and normalization, there exist other layers to narrow data, avoid overfitting and output probabilistic values. The pooling operations, with max and average being the most popular, are aimed at downsampling input data. Dropout layers are used as a mechanism to introduce some noise into the training by zeroing out some output values, thus getting rid of happenstance patterns. Weight regularization also seeks to make the model simpler by forcing the weights to be small. Finally, the output units of the model are aimed at transforming features to complete the classification task. For a multi-label problem, the Softmax represents

the probability distribution of a variable with n possible values.

The kind of problem and label representation are coupled with the cross-entropy function measuring the error. Sample labels were not hot-encoded to reduce storage footprint, and therefore, a sparse categorical cross-entropy as defined in Equation 13.1 is used for training in a multi-class problem. Otherwise, hot-encoding requires transforming labels into binary vectors of size c that activate the indices of the sample label(s), with c being the number of unique labels.

$$L_{CE} = -\ln\left(\hat{y}\left[y\right]\right) \tag{13.1}$$

where $\hat{y}$ is the model's output as a vector of size c with $\hat{y}\left[i\right], i \in [0, c-1]$ indicating the probability of the sample to belong to the i-th class, and y is the ground truth given by an integer value.

13.4.3 Architecture and training

Several architectures were checked over both datasets, transitioning from networks with a few layers to the network proposed in Figure 13.2. Hyper-parameter tunning was also used to define the best values for dropout, activation and convolutional layers, including the number of filters, the percentage of zeroed weights, or the gradient of Leaky ReLU activation. Similarly, the final activation was also checked despite Softmax being usual for multi-class output.

The input of the network is a single patch of size 23×23 and 30 bands. The spatial attention layer proposed by Xue et al. [365] is appended as an additional layer that is shown to help with the classification. The original paper proposed first and second-order pooling before and after the spatial attention module to transform the original features as we did with FA. Also, the inputted images were augmented prior to training, rather than using a single patch with each rotation being applied with a probability.

Attention-based kernels have been recently used for HSI classification to provide more discriminative spectral-spatial features from input data. Xue et al. [365] applied attention-based as a link among the first data transformation, coined as a first-order feature operator, and a second-order kernel. Roy et al. [364] used an attention-based kernel in a similar manner, though this one was replicated throughout the network pipeline. It was also part of a residual network where the input is simultaneously processed through 1) an attention kernel and 2) convolutional layers. The outputs of both pipelines are then concatenated at the ending point of each network's block.

In this work, the attention-based a) kernel of Xue et al. [365] is applied over input data and the result is concatenated in z with the original inputted data. The first operation in the attention-based thread is to normalize the data, from which the kernel will learn the weights. Such a kernel is a correlation matrix that learns the cosine distance between the central pixel and the neighbours. Then, learned weights are normalized through a

Table 13.1: Layer specifications of the proposed network. Inception is simply named, but its layers were not expanded here to make this table more readable.

Part	Layer	Kernel size	Strides	Output size
	Input			$23 \times 23 \times 30$
I	Spatial Attention			$23 \times 23 \times 30$
	Concatenate			$23 \times 23 \times 60$
	Conv2D	1×1	1	$23 \times 23 \times 16$
	Conv2D	3×3	2	$12 \times 12 \times 16$
II	Leaky ReLU ($\alpha \leftarrow 0.1$)			$12 \times 12 \times 16$
	Batch normalization			$12 \times 12 \times 16$
	Dropout (0.2)			$12 \times 12 \times 16$
	Inception v2		1 (Conv2D 1×1), 2	$6 \times 6 \times 112$
	Batch normalization			$6 \times 6 \times 112$
III	Leaky ReLU ($\alpha \leftarrow 0.1$)			$6 \times 6 \times 112$
	Dropout (0.4)			$6 \times 6 \times 112$
	Inception v2		1 (Conv2D 1×1), 2	$3 \times 3 \times 288$
	Batch normalization			$3 \times 3 \times 288$
IV	Leaky ReLU ($\alpha \leftarrow 0.1$)			$3 \times 3 \times 288$
	Flatten			2592
	Dropout (0.2)			2592
V	Softmax			9 (Red), 11 (White)
	#Trainable parameters: 395643 #Non-trainable parameters: 992 #Parameters: 396635			

softmax function that is shown to provide better convergence. With this in mind, the attention-based can be formulated as follows:

$$\mathcal{S}^{M\times M\times M} \leftarrow l_2\left(\mathcal{P}^{M\times M\times B}\left(\mathcal{P}^{M\times M\times B}\right)^{\mathsf{T}}\right) \tag{13.2}$$

$$\mathcal{S}_{central}{}^{M\times M\times 1} \leftarrow \left[\mathcal{S}^{M\times M\times M}_{\left[:,\lfloor\frac{M}{2}\rfloor,:\right]}\right]^{M\times M} \mathcal{K}^{M\times 1} \tag{13.3}$$

$$\mathcal{SA}^{M\times M\times M} \leftarrow \mathcal{S}^{M\times M\times M}\mathcal{S}_{central}{}^{M\times M} + \mathcal{B}^{M\times 1} \tag{13.4}$$

$$\mathcal{P}_{\mathcal{SA}}{}^{M\times M\times B} \leftarrow softmax(\mathcal{SA}^{M\times M\times M}) \cdot \mathcal{P}^{M\times M\times B} \tag{13.5}$$

where l_2 refers to L2 normalization, $\mathcal{P}^{M\times M\times B}$ is a 3D patch, and $\mathcal{S}$, $\mathcal{S}_{central}$ as well as $\mathcal{SA}$ are intermediate states. $\mathcal{P}_{\mathcal{SA}}$ is the result that is later concatenated with the original form. $\mathcal{K}$ and $\mathcal{B}$ are, on the other hand, the learned kernel and bias weights.

Then, the second module is aimed at extracting both spatial and spectral features, as proposed by Pan et al. [431]. The first convolutional layer is applied pixel-wise with a kernel of size 1 and thus operates by extracting patterns on the z-axis. It is used prior to kernels of larger size to compute reductions. Then, the second convolutional layer extracts spatial features over previously extracted filters. The kernel size is only 3 since deeper layers will operate over larger neighbourhoods.

Two similar blocks are included as Part III and Part IV in Figure 13.2. Both

share the same structure: Inception block, normalization, activation and dropout. It s a frequent follow-up of convolutional layers [365, 432], with dropout being greater (0.4) for middle layers than the last and initial layers (0.2). Instead of using max-pooling to downsample the network, strides in convolutional layers were observed to perform better. Thus, strides of size 2 were used in spatial-based CNN layers. The network specifications are shown in Table 13.1.

The Inception block was first proposed by [433]. The first proposal consisted of a module with four parallel layers later concatenated: convolutional layers with different kernel sizes (1 for spectral features and 3 and 5 to obtain aggregations from surrounding pixels), and a max-pooling layer that works directly over input data. Accordingly, a response map with a large number of filters is obtained. The importance of each of them is determined by the following layers that will again downsample data. However, at the time this layout was considered to be prohibitive if the input layer has a large number of filters, especially for kernels of larger size. Therefore, 1×1 convolutions aimed at reducing data were attached before each one of the Inception threads (max-pool and convolutions with $\kappa > 1$). In this work, both proposals were used: the naïve is checked in the experimentation to increase the network's capacity, whereas the second is part of the proposed network. The latter compresses spatial data even more and is following connected to the network output. The code showing the skeleton of the proposed network is shown in Code 13.1.

Listing 13.1: The proposed CNN network aimed at classifying different vineyard varieties.

```
def get_inception_module(input, start_size, strides):
    conv1_a = Conv2D(start_size, (1, 1), strides=1, padding
    ↪ ='same', activation='relu')(input)
    conv2_a = Conv2D(start_size, (3, 3), strides=strides,
    ↪ padding='same', activation='relu')(conv1_a)

    conv1_b = Conv2D(start_size, (1, 1), strides=1, padding
    ↪ ='same', activation='relu')(input)
    conv2_b = Conv2D(start_size, (5, 5), strides=strides,
    ↪ padding='same', activation='relu')(conv1_b)

    max_pool_c = MaxPooling2D(strides, padding='same')(
    ↪ input)
    conv1_c = Conv2D(start_size, (1, 1), padding='same',
    ↪ strides=1, activation='relu')(max_pool_c)

    output = Concatenate(axis=3)([conv2_a, conv2_b, conv1_c
    ↪ ])
    return output

in_patch = Input(shape=img_size)
x = in_patch

y = Reshape([img_size[0] * img_size[1], img_size[2]])(x)
y = Lambda(lambda z: K.l2_normalize(z, axis=-1))(y)
y = SpatialAttention()(y)
y = Reshape([img_size[0], img_size[1], img_size[2]])(y)
x = Concatenate(axis=3)([x, y])

```

```
x = Conv2D(start_size * 1, 1, strides=1, padding="same")(x)
x = Conv2D(start_size * 1, kernel_size, strides=strides,
    ↪ padding="same")(x)
x = LeakyReLU(alpha=0.1)(x)
x = BatchNormalization()(x)
x = Dropout(0.2)(x)
x = get_naive_inception_module(x, start_size * 2, strides=
    ↪ strides)
x = BatchNormalization()(x)
x = LeakyReLU(alpha=0.1)(x)
x = Dropout(0.4)(x)
x = get_inception_module(x, start_size * 6, strides=strides
    ↪ )
x = BatchNormalization()(x)
x = LeakyReLU(alpha=0.1)(x)
x = Flatten()(x)
x = Dropout(0.2)(x)
outputs = Dense(num_classes, activation="softmax")(x)
model = Model([in_patch], outputs)
```

Finally, the model is fitted with training data and its performance is assessed with validation samples. For supervised problems like ours, data is composed of both samples and ground truth. In this work, the training samples were split into several sets according to the hardware limitations, and the model was iteratively trained with a significant number of iterations for every set (for instance, 100). Besides mitigating storage limitations, this leave-p-out cross-validation also helps to generalize by not training over the complete dataset. Furthermore, each one of these clusters is further split into small batches during a single iteration. The batch size must be large enough to include a balanced representation of samples. In this work, the batch size was set to 2^{10}. This phase can be terminated early if no improvements are observed during 20 epochs.

13.4.4 Data sampling and regularization

It can be observed from Figure 13.3 that the dataset is clearly not balanced. The number of ground-labelled patches is considerably larger, whereas vineyard rows also differ in length and so does the number of examples for each variety. The unbalancing is partially mitigated by discarding ground-labelled samples, which can be also rapidly discarded in nature with the NDVI. With the rest of the samples, several subsampling techniques were checked. These range from random selection to clustering, with the second returning better results. The resulting dataset was again split into multiple batches since they did not fit in the GPU, and therefore, the training required several iterations depicted as peaks in the later experiments. Therefore, all of these batches are balanced. In any case, the CNN is watched with a callback that saves the current best model and prevents saving an overfitted model.

To summarize, the sampling strategy is the following. First, the pre-processing transformations are fit to the training samples, $\mathfrak{X}$. Then, the whole HSI cube, $\mathfrak{B}$, is transformed according to the models learnt with $\mathfrak{X}$,

allowing us to simulate the classification of new, non-labelled, samples in nature. After that, 3) HSI cubes are split into squared-odd-sized patches ($\mathscr{H}^{W\times H\times D} \rightarrow \mathscr{P}^{S\times S\times D}$ with S being an odd number), 4) patches' labels are reduced into the label of the central pixel ($f(\mathscr{P}^{S\times S}_{\mathtt{label}}) \rightarrow \mathscr{P}^{1\times 1}_{\mathtt{label}}$), 5) the dataset is balanced with clustered downsampling ($\mathscr{P} \rightarrow \mathscr{B}$) and 6) $\mathscr{B}$ is divided into train $\mathscr{B}_{0.7} \rightarrow \mathscr{X}$ and test datasets $\mathscr{B}_{0.3} \rightarrow \mathscr{T}$, where the former is again split during training to obtain the validation dataset (10% of $\mathscr{B}$).

In addition, batches were augmented by performing rotations and orientation flips so that learn features are invariant to the flight's positioning conditions (see Figure 13.4). With this approach, the dataset was augmented by ×5. Hence, the regularization was controlled by the proposed downsampling and augmentation sequences. Several considerations were also taken into account during the CNN design: 1) the CNN must not have a large number of trainable parameters to avoid overfitting and a sufficient number to cope with underfitting, and 2) dropout layers were included to randomly reset some output weights and thus lead to proper generalization.

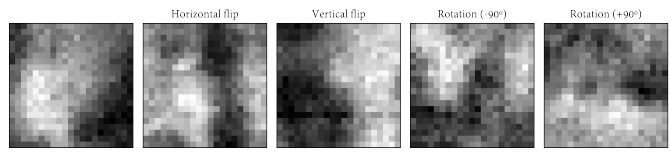


Figure 13.4: Augmentation of a single hyperspectral patch (23 × 23).

13.5 Experimentation and analysis

A brief explanation of the study area and sensors is first provided in this section; then, the performance of the proposed network is evaluated over multiple hyperspectral swaths collected in the two surveyed areas. Results are presented in terms of overall accuracy (OA), average accuracy (AA), statistical kappa (κ) and f1-score. The first shows the percentage of correctly classified samples, the AA represents the average class-wise accuracy, the κ coefficient is the degree of agreement between the classification results and the ground truth, and finally, the f1-score measures the model's precision by leveraging both precision and recall metrics. Several representative neural networks are compared with our method: LtCNN [358], JigsawHSI [361], SpectralNET [362], HybridSN [363], Nezami et al. [359] and A-SPN [365]. From these, only a few address airborne-sensing imagery [358], whereas the rest are focused on satellite data. Hence, several considerations must be addressed: 1) some of these manuscripts apply different transformations to the input data and 2) the number of spectral bands also differ from our sensing tool. Therefore, the preprocessing pipeline was selected as the one providing better performance over our input data, either our pipeline or the one proposed in the reference work. However, FA showed a higher performance for every network if input data was transformed according to this fitting method, rather than the following:

- Nezami et al. [359], Liu et al. [358] (LtCNN) used the corrected re-

flectance with no preprocessing.

- Moraga and Duzgun [361] (JigsawHSI) used PCA, FA, SVD and NNMF with 9-12 final features.
- Liu et al. [358] (A-SPN) and Roy et al. [363] (HybridSN) used PCA to transform reflectance with $n \leftarrow 15, 30$, respectively, whereas n is unknown for [365].
- Chakraborty and Trehan [362] (SpectralNet) used FA with only 3 features.

Regarding implementation, all the tests were performed on a PC with AMD Ryzen Threadripper 3970X 3.6 GHz, 256 GB RAM, Nvidia RTX A6000 GPU and Windows 10 OS. The proposed CNN as well as the compared networks were implemented with Keras (version 2.10.0) and TensorFlow (version 2.10.1) in Python. CUDA 11.8 and CuNN 8.6 were installed to reduce the fitting time. Not every network from previous work could be applied as published; for example, LtCNN is designed for large image patches (200×200) and thus convolutional striding and MaxPooling cannot be applied when patches reach a size of 1×1. For LtCNN [335], kernel size and max pooling's strides were reduced as depicted in Figure 13.17. The rest of the networks are also documented, from Figure 13.16 to Figure 13.21.

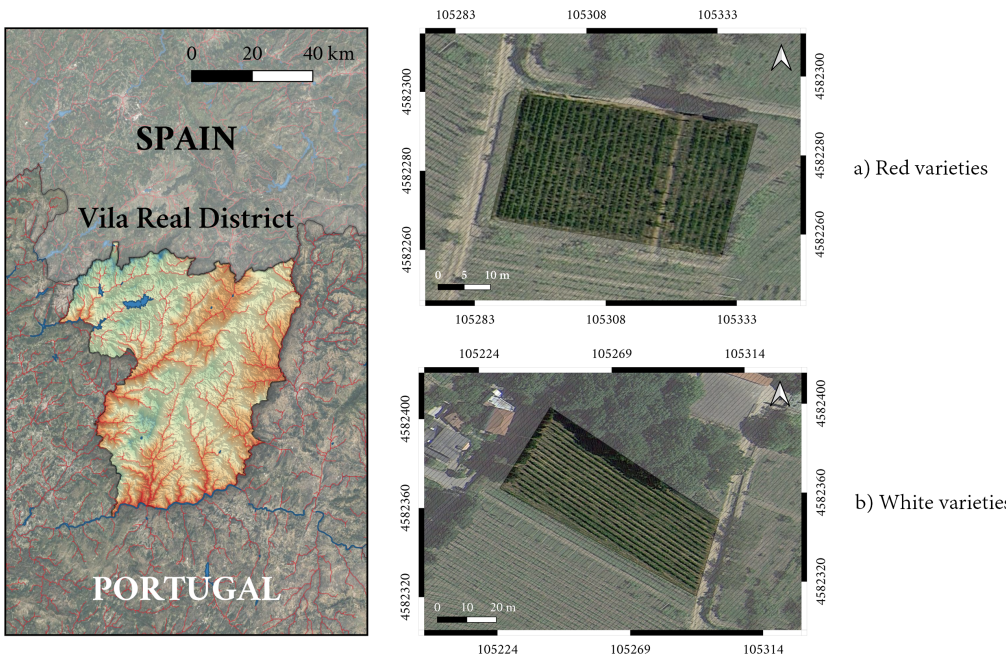


Figure 13.5: Overview of the areas surveyed with UAS hyperspectral imaging for the classification task. Two different vineyard crops are depicted according to their main variety: a) red and b) white.

13.5.1 Study area

The vineyards used as study areas in this work are situated in the Northern region of Portugal, specifically in Vila Real (Figure 13.5). Each vineyard plot

is dedicated to either red or white grapevine variants, and each grapevine variety is cultivated in one or more contiguous rows. Data acquisition occurred in two ways: 1) by gathering leaves from each row, directly on the field, and 2) by conducting a UAS flight, which is described subsequently. The initial outcomes were analyzed in a laboratory, while the second survey was dedicated to collecting HSI swaths.

13.5.2 Material

The M600 hexacopter equipped with a Nano-Hyperspec from Headwall was used for the UAS flight. The flight was planned using Universal Ground Control Station at an altitude of 50 m with a 40% side overlap. 5 and 8 swaths were obtained for red and white grapevine varieties, respectively, as depicted in Figure 13.15. The leaves collected were analyzed at the University of Trás-os Montes e Alto Douro (UTAD) using a spectrometer, where reflectivity was acquired by sampling the front and back of each leaf under controlled illumination. The first and last reflectance samples were discarded since they were noisier, thus narrowing down the spectral interval from 187.63-1100.35 nm to 450-950 nm. Even after applying filters, some spectral signatures differed significantly from the expected average profile (μ) and were removed based on the distance calculation from μ. Table 13.5 provides a summary of the collected grape leaf samples.

13.5.3 Impact of window size

The patch size is probably the most relevant parameter concerning the network architecture. Larger patches are assumed to also work as irrelevant spatial features can be zeroed out. However, it also comes at the expense of increasing the training time. On the contrary, lower patches come at the risk of not being sufficient for classifying samples as accurately as done by the proposed network. Figure 13.6 shows the whole battery of metrics obtained with patch sizes ranging from 9 to 31. On the other hand, Figure 13.7 emphasizes the training time, in minutes, and the dimensionality of the network. According to the obtained results, patches have been split with dimensionality 23 to balance network capacity and accuracy, despite higher patch sizes achieving slightly better results. Accordingly, the highest patch size reached an OA of 99.63% for red varieties, whereas the lowest reached 86% (9). On the other hand, the selected dimensionality achieves an OA of 99.37%. Regarding white varieties, the proposed network did not achieve better results with larger patch sizes. The best OA was obtained using patches of size 23, whereas the lowest was reached using the lowest patch size, 9, with an OA of 85%.

13.5.4 Ablation study

The proposed network is intended to be validated in this section by removing some of the network features, while the rest remain unchanged. The proposed changes are the following:

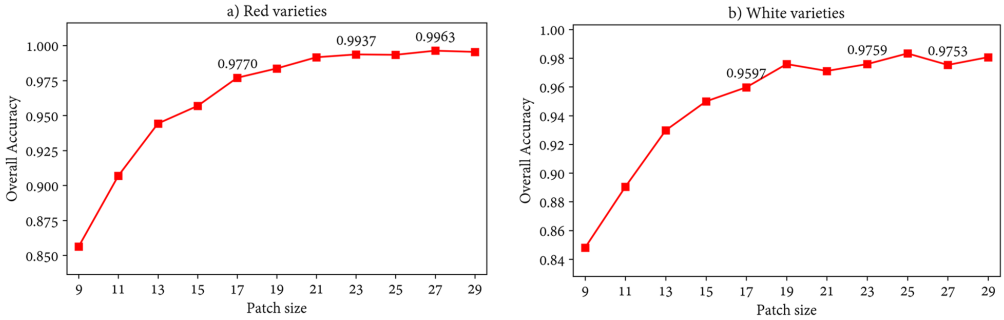


Figure 13.6: Overall accuracy obtained for patches of different sizes, from 9 to 31.

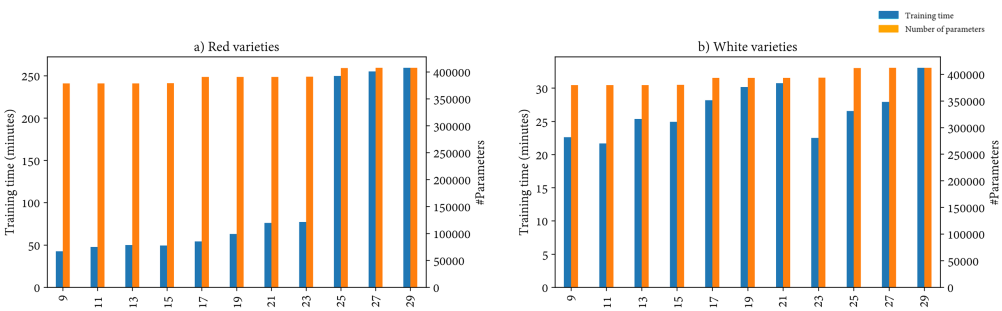


Figure 13.7: Training time, in minutes, and number of parameters as the window size increases.

1. The first two convolutional layers to extract spectral and spatial features were removed, thus only remaining the two Inception blocks.
2. Both Inception blocks were modelled using the naïve version originally proposed by [433] (see Figure 13.8). The main difference between this architecture and ours is that the former extracts features with different neighbourhood sizes using the spectral bands as input for the later convolutional layers.
3. Only the first Inception block was exchanged by a naïve version, as the one used in the previous experiment.
4. The spatial attention layer was not included.

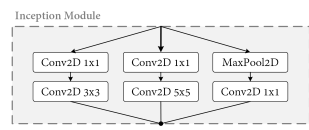


Figure 13.8: First proposal of Inception block [433].

The obtained results are shown in Table 13.2. Removing the SA layers led to a slight decrease in performance, similar to removing the first convolutional layers. However, the latter returned worse results since spectral features were not retrieved prior to normalizing attributes for the first time and passing them to the Inception layers. Unsurprisingly, using the first version of the Inception layer twice led to a significant performance decrease for every metric, as it kept transforming the spectral dimensionality in a deep layer. The second Inception version also transforms spectral features but rather provides them as an additional layer (concatenation) that can be weighted according to their contribution to the output. Following this reasoning, swapping the first Inception layer with the first version of it

Table 13.2: Results in terms of Overall Accuracy (OA), Average Accuracy (AA) and Kappa coefficient with different methods.

Metric	Ours	1) Without initial convolutions	2) Double naïve Inception	3) Naïve & advanced Inception	4) Without spatial-attention
Red					
OA	**99.31 ± 0.05**	98.80 ± 0.14	98.58 ± 0.08	99.00 ± 0.06	99.00 ± 0.06
AA	**99.29 ± 0.03**	98.51 ± 0.16	97.70 ± 0.13	98.98 ± 0.08	98.92 ± 0.07
Kappa (κ)	**99.20 ± 0.07**	98.21 ± 0.25	98.57 ± 0.14	98.80 ± 0.08	98.86 ± 0.07
f1	**99.31 ± 0.05**	98.80 ± 0.14	98.58 ± 0.08	99.00 ± 0.06	99.00 ± 0.06
White					
OA	**98.43 ± 0.13**	97.87 ± 0.58	97.47 ± 0.15	98.09 ± 0.16	98.14 ± 0.09
AA	**99.18 ± 0.04**	98.81 ± 0.25	98.64 ± 0.09	99.01 ± 0.09	99.01 ± 0.04
Kappa (κ)	**98.36 ± 0.16**	96.48 ± 1.16	97.24 ± 0.17	97.77 ± 0.20	98.53 ± 0.04
f1	**98.43 ± 0.13**	97.86 ± 0.59	97.48 ± 0.15	98.09 ± 0.16	98.14 ± 0.09

did not involve a huge performance drop. Improvements to the proposed architecture over the c) variant were very small and therefore may suggest that using either one of them does not offer great changes in the performance. Similar results to this last setup were achieved by removing the spatial attention layer; it did not lead to a significant performance drop, though better results were obtained using it.

13.5.5 Training time and network capacity

The benchmark on the classification problem is relevant to show that training time is not excessive and that the proposed network does not have much more capacity than the problem warrants. Regarding the capacity, Figure 13.9 shows that training and test signatures are similar, and therefore do not show overfitting nor underfitting behaviours. Furthermore, the training AO and loss worsen as a new dataset is introduced, whereas the validation metrics remain similar. Thus, the network is not memorizing input data. Along with this, the network is only parameterized by nearly 400k parameters, while other state-of-the-art models exceed ten million parameters (see Figure 13.10). The number of parameters is derived from the proposed architectures, using an input of size $23 \times 23 \times 30$. Note that the network of Lu et al. [335] was decimated in our experimentation with pooling operations of a lower size than proposed to adapt it to smaller patches. Finally, the response time for training the proposed network is below an hour, whereas others require up to several hours. The fitting over datasets concerning white varieties required half an hour since a faster convergence was achieved and most of the compared network, including ours, exited the training without completing it. Note that every available sample was used during training, instead of using strides; otherwise, the training time can be reduced. Further insight into these results is provided by Table 13.3.

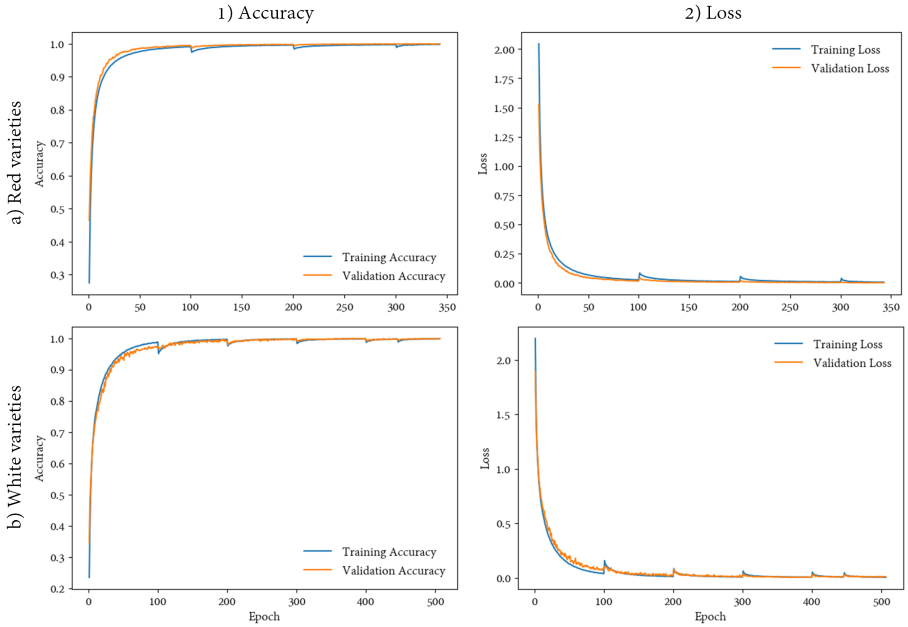


Figure 13.9: Overall accuracy and loss during training.

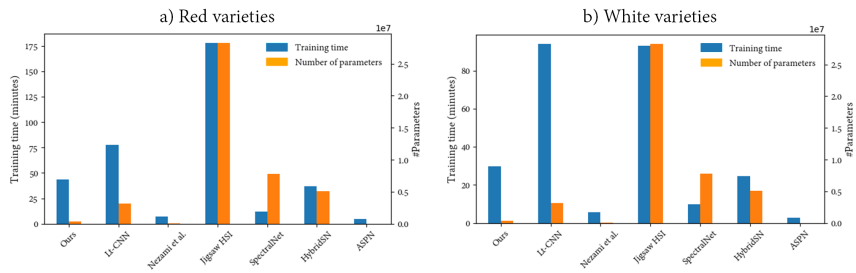


Figure 13.10: Response time for training the network over a) red and b) white grapevine samples, as well as the number of parameters for every compared network, including ours.

Table 13.3: Overall results in terms of Overall Accuracy (OA), Average Accuracy (AA) and Kappa coefficient with different methods.

Metric	Ours	LtCNN	Nezami et al.	JigsawHSI	SpectralNET	HybridSN	A-SPN
Red variety							
OA	**99.31 ± 0.05**	87.37 ± 2.15	92.04 ± 0.42	91.18 ± 0.78	88.57 ± 0.15	69.05 ± 0.43	76.62 ± 0.63
AA	**99.29 ± 0.03**	85.39 ± 3.02	90.94 ± 0.56	91.06 ± 0.98	89.59 ± 0.26	62.51 ± 0.51	70.62 ± 0.53
Kappa	**99.20 ± 0.07**	87.43 ± 2.24	92.87 ± 0.35	90.16 ± 0.91	86.16 ± 0.25	68.96 ± 0.66	76.56 ± 0.46
f1	**99.31 ± 0.05**	87.65 ± 2.06	92.12 ± 0.41	91.33 ± 0.76	88.66 ± 0.16	70.11 ± 0.38	76.89 ± 0.52
White variety							
OA	**98.43 ± 0.13**	79.43 ± 6.85	89.07 ± 0.87	80.42 ± 5.28	71.05 ± 1.45	57.08 ± 1.25	57.53 ± 1.47
AA	**99.18 ± 0.04**	86.47 ± 6.05	92.81 ± 0.46	87.67 ± 3.59	79.41 ± 1.36	69.52 ± 1.08	72.58 ± 1.14
Kappa	**98.36 ± 0.16**	77.17 ± 8.01	85.84 ± 1.07	80.51 ± 4.26	72.15 ± 1.30	50.44 ± 1.6	49.91 ± 3.03
f1	**98.43 ± 0.13**	79.20 ± 7.08	88.99 ± 0.91	80.35 ± 5.00	70.87 ± 1.45	56.01 ± 1.37	55.37 ± 2.03

13.5.6 Separability

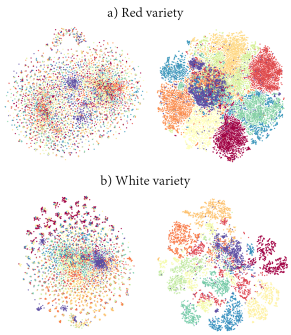


Figure 13.11: Clustering of samples according to the feature transformation performed by UMAP over 1) the starting hyperspectral features and 2) features extracted by the CNN prior to transferring it to the final Softmax layer.

The output of the proposed network can be assessed in terms of separability by removing the final Dropout and Dense layers. Data is transformed and flattened according to the network's learnt weights. It is subsequently embedded with UMAP [336] to compress high-dimensionality data into a few features, thus allowing us to visualize the new data representation. The same procedure can be followed over the original data to compare how was the data manifold uncrumpled. As shown in Figure 13.11, different labels were not perfectly unmixed, although the improvement in comparison to the starting representation is notable. To provide this result, the last Flatten layer was connected to UMAP fitting with $n = 2$; hence, 2592 features were narrowed to two features to represent the embedding in a two-dimensional chart.

13.5.7 Confusion matrices

Figure 13.12 shows the OA of the proposed network against any grapevine variety. Hence, classification over red varieties showed uniform results, with most of them being close to 99%. Note that these percentages were rounded, and therefore, some of these results are below 99%, while others are above. When averaged, all these results lead to an OA of $\approx 99.3\%$, as shown in Table 13.3. On the other hand, most white varieties were better classified, with a significant number being perfectly annotated. However, others were more often mistaken, for instance, class five, which led to a worse OA.

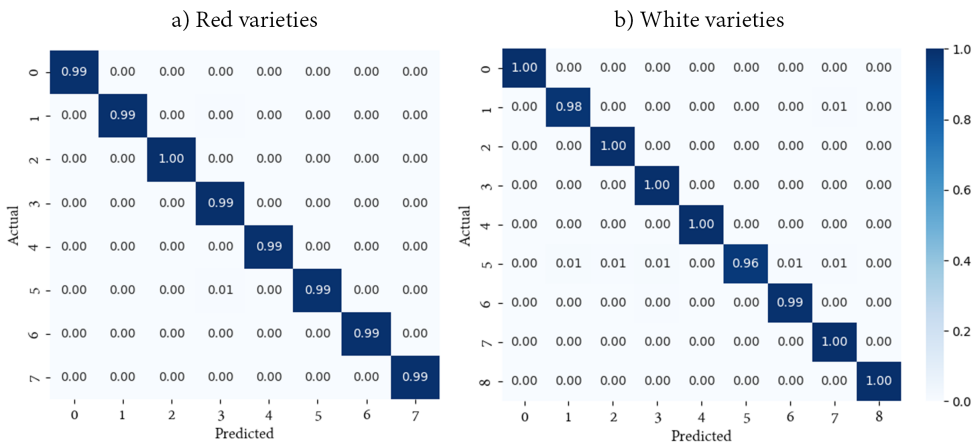


Figure 13.12: Confusion matrices for the classification of red and white varieties.

13.5.8 Analysis of errors

As observed in previous sections, our architecture achieved a high OA and AA. Still, there is a margin for improvement that must be tackled by

finding which are the weaknesses in the overall labelling, transformation and classification pipeline. Instead of predicting randomly selected samples, another experiment is to predict every hyperspectral swath sample, thus allowing us to determine where errors take place within the study area. As observed in Figure 13.13, these errors are spatially clustered instead of being sparsed over the study area. If these are compared against the RGB mosaic of the hypercubes, errors are observed to belong to 1) small vegetation clusters, mainly proceeding from small vegetation mistakenly labelled as vineyard and 2) samples surrounded by ground or metallic vineyard supports. Note that these are hard to notice during the labelling since they present signatures similar to the target leaves and they are surrounded by vegetation, thus hardening the definition of a geometrical shape for rapidly tagging which is relevant or not. Still, some errors are present in grapevine samples surrounded by ground and other surfaces since these have a notable impact on the sample's neighbourhood, thus distorting the final probability. Note that every boundary sample is surrounded by vegetation, and therefore, it should be expected that their signature is the fusion of the signatures concerning a target variety and ground. However, it may be not relevant enough to tell apart varieties, thereby leading to more errors on samples heavily surrounded by ground.

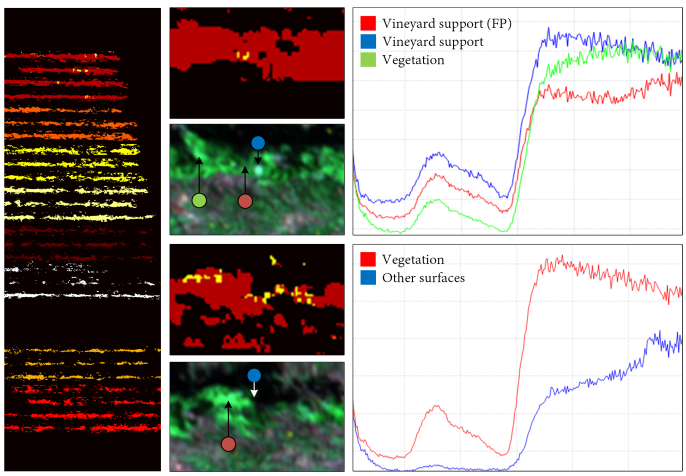


Figure 13.13: Errors observed in the classification of red varieties, together with the hyperspectral signature of a few samples concerning different surfaces.

13.5.9 Training over satellite imagery

The main shortcoming of compared CNNs focused on satellite imagery is that they obtain a poor performance over the proposed UAS datasets and vineyard varieties. Hyperspectral imagery from UASs is noisier than satellite observations, with the spectral signature of the latter being more smoothed out. Therefore, previous work did not overcome noise in the classification of vineyards and most of them showed poor performance. Only another architecture tested over UAS samples managed to reach an OA near 90%, whereas others showing similar results proposed huge networks with millions of parameters, thus overkilling the classification with a large training time.

Table 13.4: Classification of hyperspectral imagery from satellite platforms in terms of Overall Accuracy (OA) and Kappa coefficient (κ).

	Ours			State-of-the-art		
Dataset	OA	Kappa (κ)	Batch size	OA	Kappa (κ)	Reference work
Pavia university	99.77 ± 0.02	99.70 ± 0.06	128	100 ± 0.00	100 ± 0.00	[361]
Indian pines	99.92 ± 0.00	99.96 ± 0.00	128	99.93 ± 0.07	99.89 ± 0.10	[434]
Salinas valley	100 ± 0.00	100.0 ± 0.00	512	100 ± 0.00	100 ± 0.00	[361]

The aim of this chapter is to remark on how our network behaves over satellite hyperspectral imagery. Although the network was not designed for this purpose, several tests were launched over publicly available datasets frequently used for comparison. The number of labels in these data ranges from nine to sixteen, and the number of spectral bands also differs from our imaging device. However, FA was fitted to obtain only 30 features per pixel, as proposed for UAS imagery, and so does the architecture of the network. According to the number of samples of each dataset, the batch size was adapted as shown in Table 13.4. Every dataset had unlabelled samples which were removed from the training and test datasets to establish a fair comparison with previous work. Unlike our UAS datasets, labels in satellite imagery were imbalanced, with some of them having only a few dozen of examples. Hence, balancing was not applied in satellite datasets to avoid levelling the rest of the classes with others that present scarce examples.

From the results, it can be observed that the proposed network can also be applied to this kind of imagery despite not being its main area of expertise. All of the datasets converged early to classification metrics over 98%, and despite not providing state-of-the-art results, these are very close to the current networks that show the best performance. As the purpose was not to tune the network for satellite imagery, the learning rate remained as before, and the batch size was scaled according to the number of samples.

13.6 Conclusions and future work

The proposed network presents a CNN model that uses current state-of-the-art procedures applied to classify grapevine varieties. The spatial-attention layer was proved to enhance the results, whereas Inception modules were checked to discuss which one could provide better results over hyperspectral imagery. Note that the majority of CNN works in the literature are oriented towards RGB imagery, and therefore, their transition to hypercubes is not as trivial. Also, only a few works have addressed the classification of UAS-based HSI, since it is considerably noisier. Despite this network being proposed for phenotyping applied to grapevines, it was also tested over standard satellite HSI datasets which are frequently used to establish fair comparisons. The results showed that samples from satellite imagery were also classified with notable accuracy (over 99.7%), although current state-of-the-art networks achieve better results. In contrast, most of the compared models achieved poor results over UAS-based imagery ($\sim$ 90% at most). Our

training time and the number of parameters were also lower than most of the compared classification models; some of them are composed of millions of parameters, whereas ours solely consist of ∼400K parameters.

Another explored step, which may be the most relevant, is the preprocessing of reflectance. First, hypercubes are composed of a huge number of bands that cannot be processed by CNNs in a reasonable response time. Furthermore, it hardens the fitting stage and most of these bands may be either redundant or irrelevant to the classification. A significant number of feature reduction and transformation methods were checked, which led to Factor Analysis as the one providing better results in an automatic pipeline. The latter concept is relevant since other methods may be capable of providing better clusters at the expense of demanding the number of different materials collected in hyperspectral imagery. Using FA, hypercubes with 270 bands were narrowed to only 30 features which led to reducing the number of trained parameters and response time.

Regardless of the high OA and AA, the analysis of errors also depicted some clustered samples that were mislabelled. If checked against false-colour RGB imagery from hypercubes, most of them were low-vegetation labelled as grapevine varieties or samples surrounded by other surfaces which hardened the classification.

As a future work, this network could be further extended to integrate as many more varieties as possible. It also ought to be explored whether a different growth stage has some effects on the classification. Hence, proving this would enable this work to be used over any study area at any stage of the year as long as it comprises some of the varieties over which the model has been trained. Also, a considerable effort is being made to collect information about crops receiving grants from national and European funds. For instance, it should be inferred if a crop is abandoned or not to make a decision on whether these funds are granted. Similarly to this, estimating the area of vineyard rows, even the harvested varieties, could also help in this and other decision-making processes. Finally, the labelling may greatly benefit from using a DEM as collected by the surveying UAS, in order to avoid some labelling errors and speed up this manual task.

Additional material

Table 13.5: Summary of information collected from different grapevine varieties. For each variety, the number of field rows is shown, along with the number of laboratory samples and the number of collected leaves. Regarding UAS acquisition, samples are obtained by labelling aerial imagery.

			Spectrometer		UAS
Root variety	**Variety**	**#Rows**	**#Leaves**	**#Samples**	**#Samples**
Red	Alicante	3	8	96	20,910
	Alvarhelao	4	6	36	100,285
	Barroca	3	9	108	17,219
	Sousao	3	6	36	29,960
	Touriga Femea	3	6	36	38,568
	Touriga Francesa	3	9	54	14,894
	Touriga National	3	9	108	38,925
	Tinta Roriz	4	9	108	31,091
White	Arito Do Douro	1	6	50	58.813
	Boal	3	8	72	22,751
	Cercial	1	6	36	13,556
	Codega Do Ladinho	3	16	140	146,083
	Donzelinho Branco	1	2	12	34,019
	Malvasia Fina	3	16	129	139,985
	Moscatel Galego	1	6	54	48,352
	Moscatel Galego Roxo	1	6	36	47,745
	Samarrinho	1	6	36	43.560

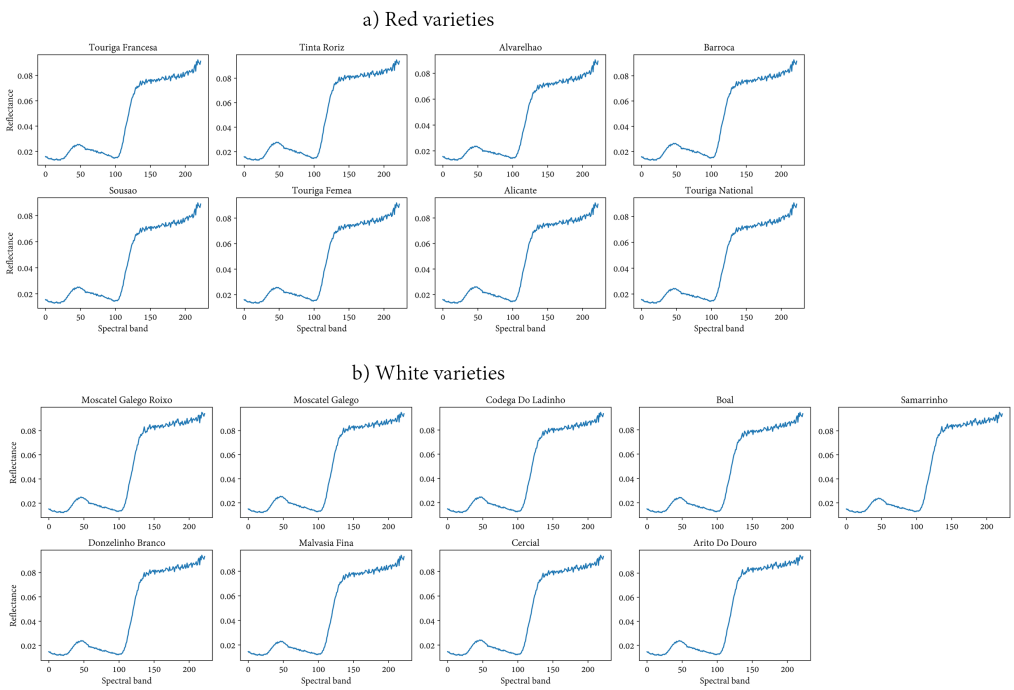


Figure 13.14: Mean spectral signature of a) red and b) white grapevine varieties.

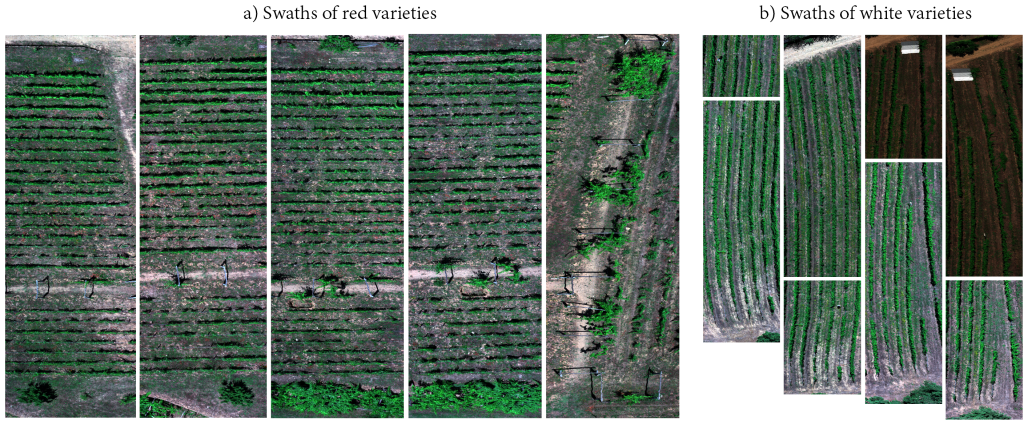


Figure 13.15: False RGB colour images from the collected hypercubes depicting two different areas, focused on 1) red and 2) white grapevine varieties.

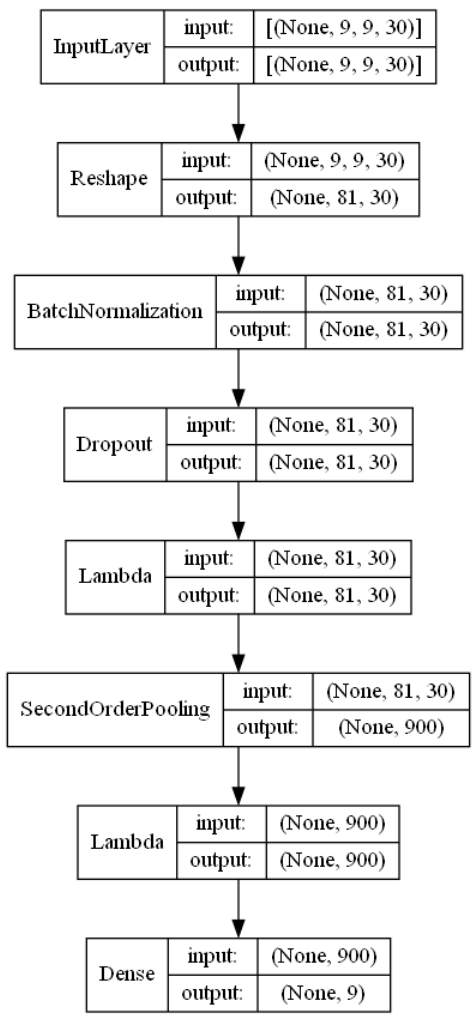


Figure 13.16: CNN architecture proposed by Xue et al. [365] as implemented in this study to establish a comparison.

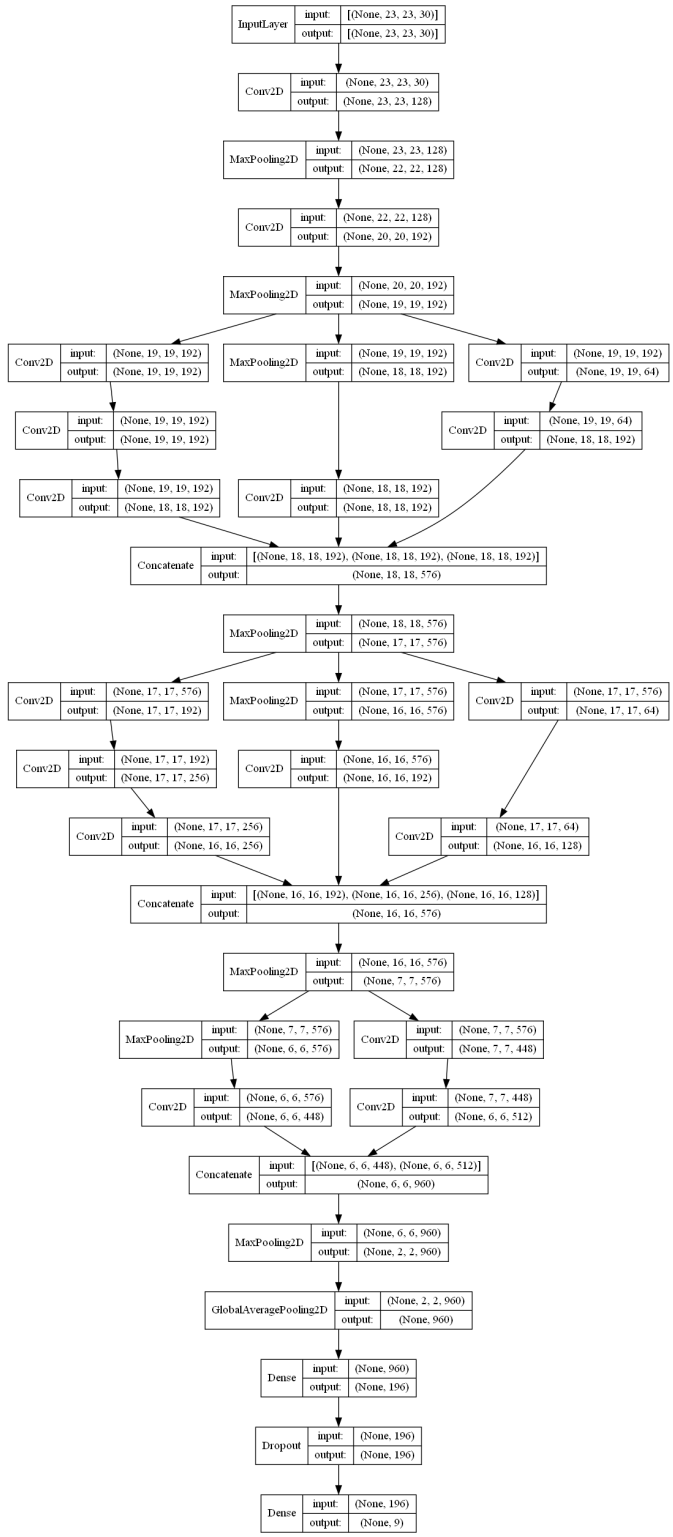


Figure 13.17: CNN architecture proposed by Liu et al. [358] as implemented in this study to establish a comparison.

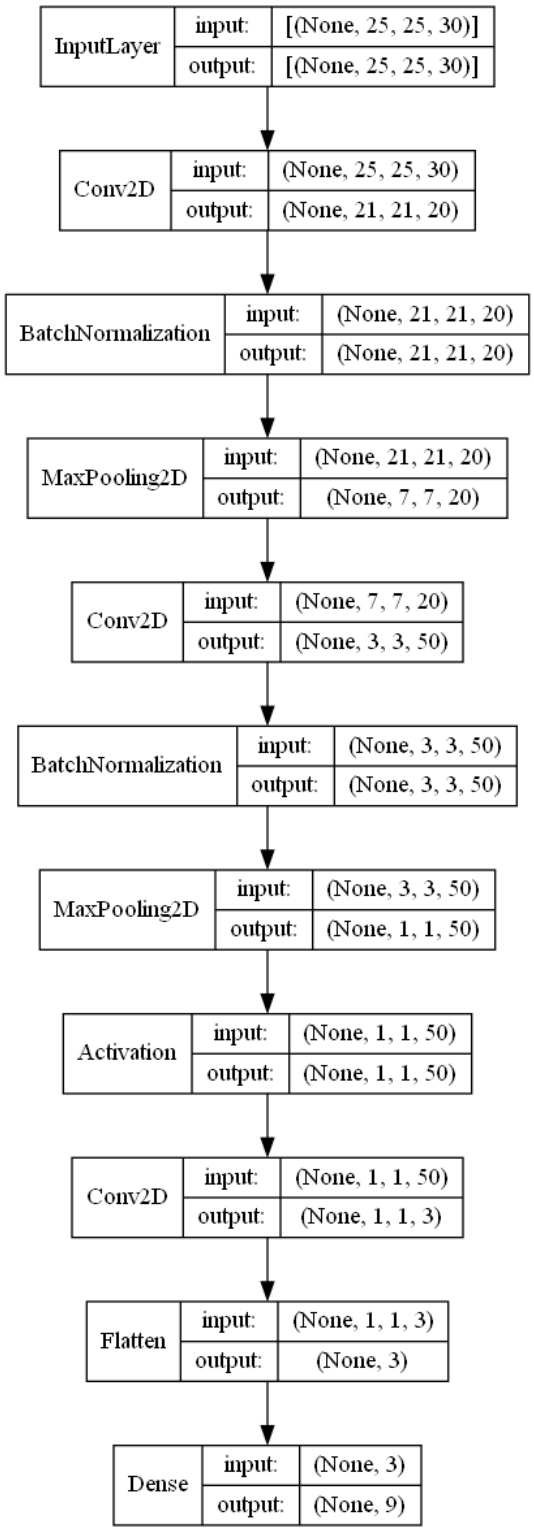


Figure 13.18: CNN architecture proposed by Nezami et al. [359] as implemented in this study to establish a comparison.

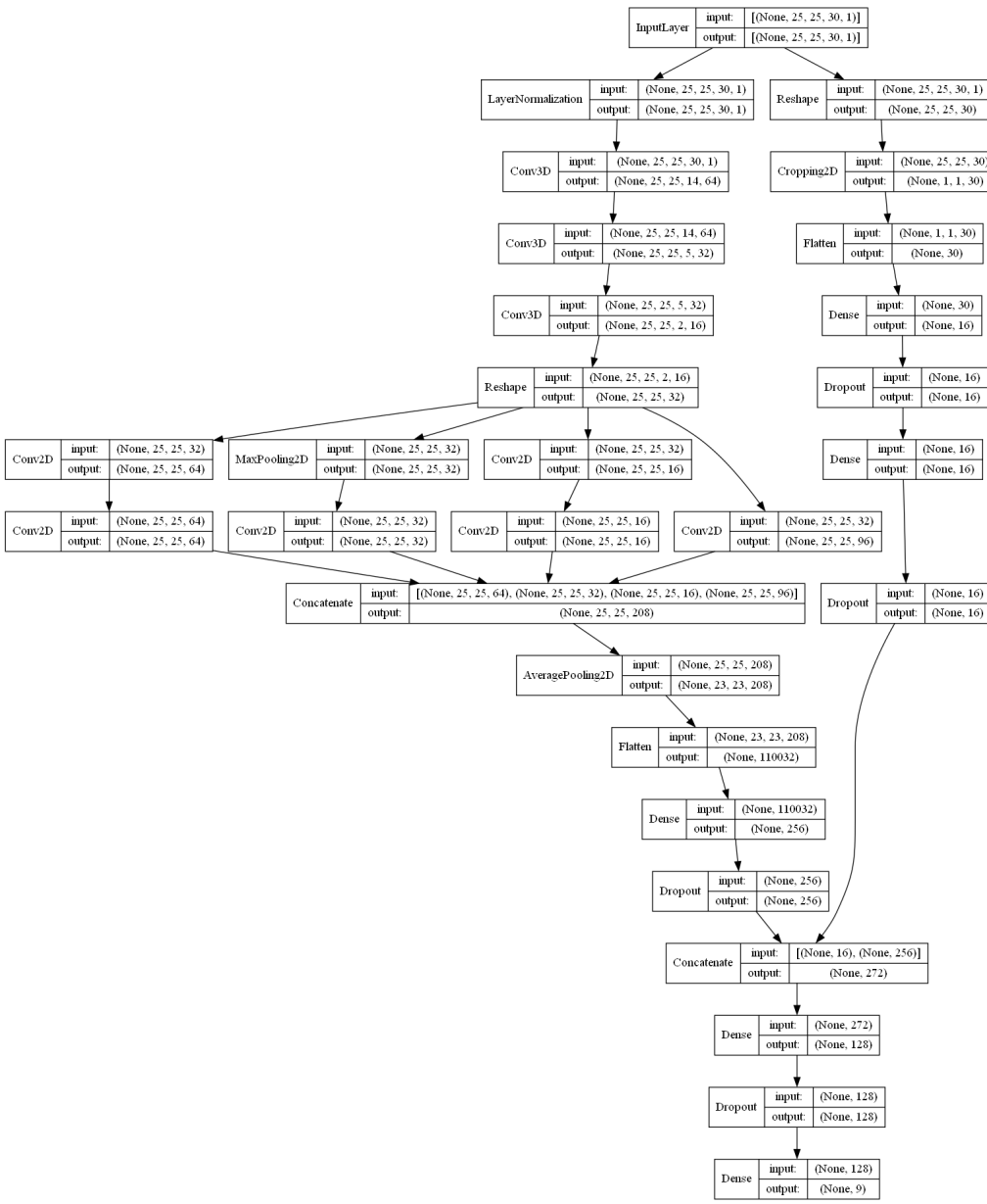


Figure 13.19: CNN architecture proposed by Moraga and Duzgun [361] as implemented in this study to establish a comparison.

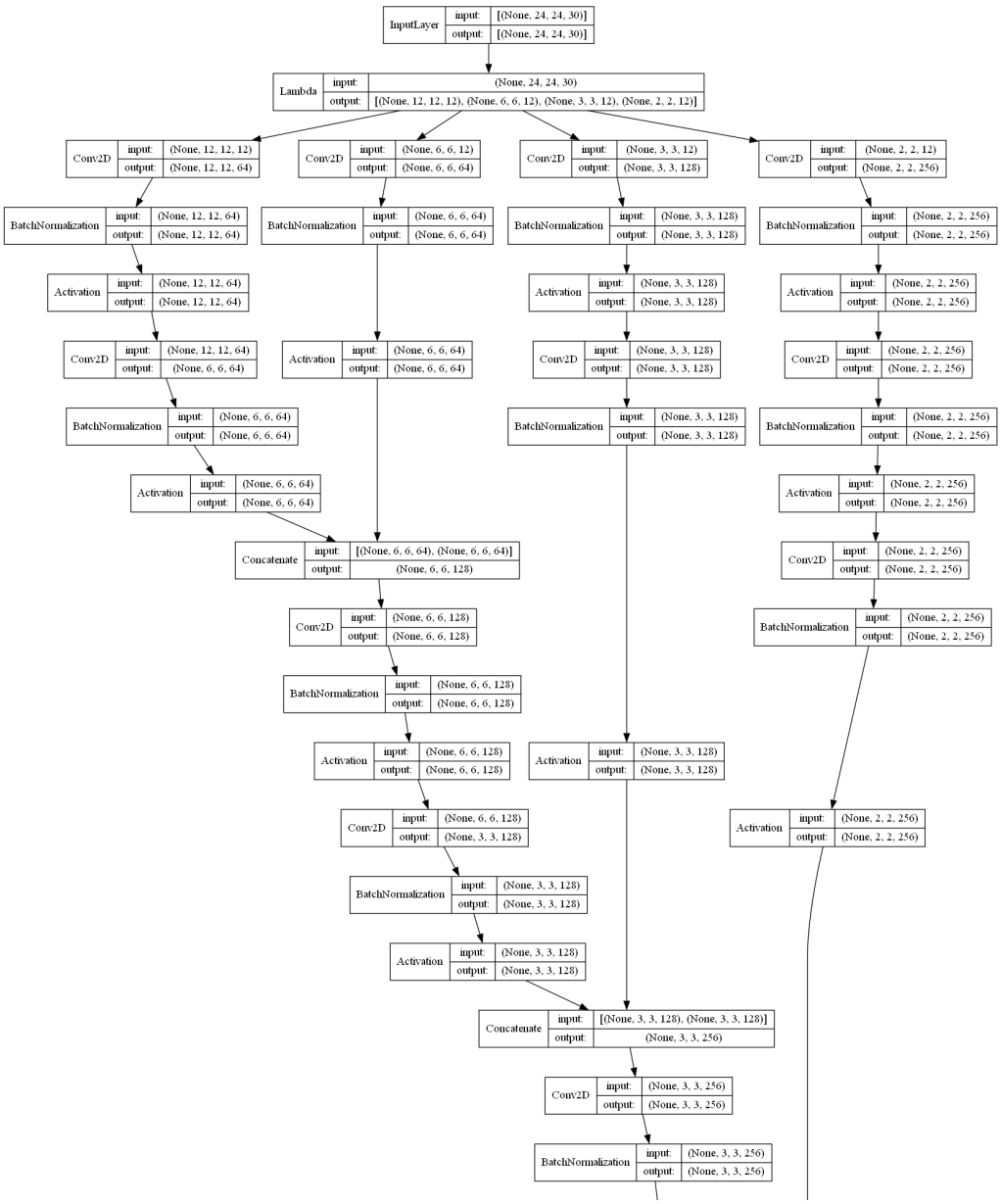


Figure 13.20: CNN architecture proposed by Chakraborty and Trehan [362] as implemented in this study to establish a comparison.

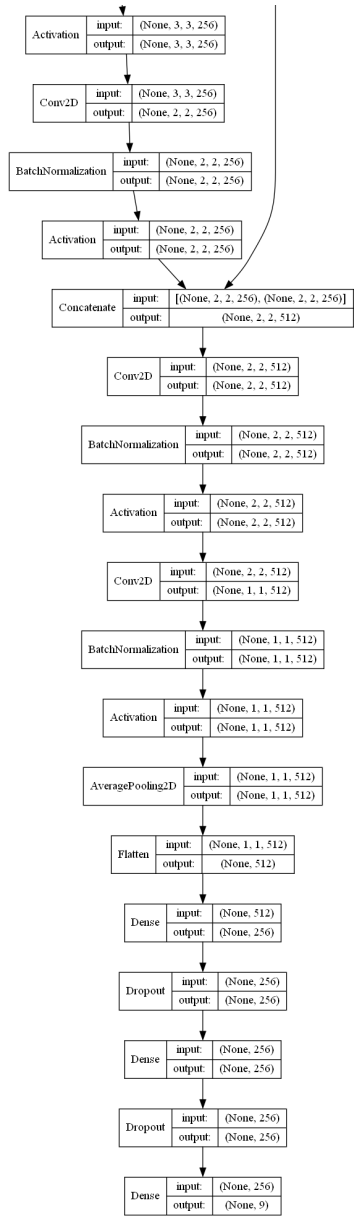


Figure 13.20a: Continuation of Figure 13.20.

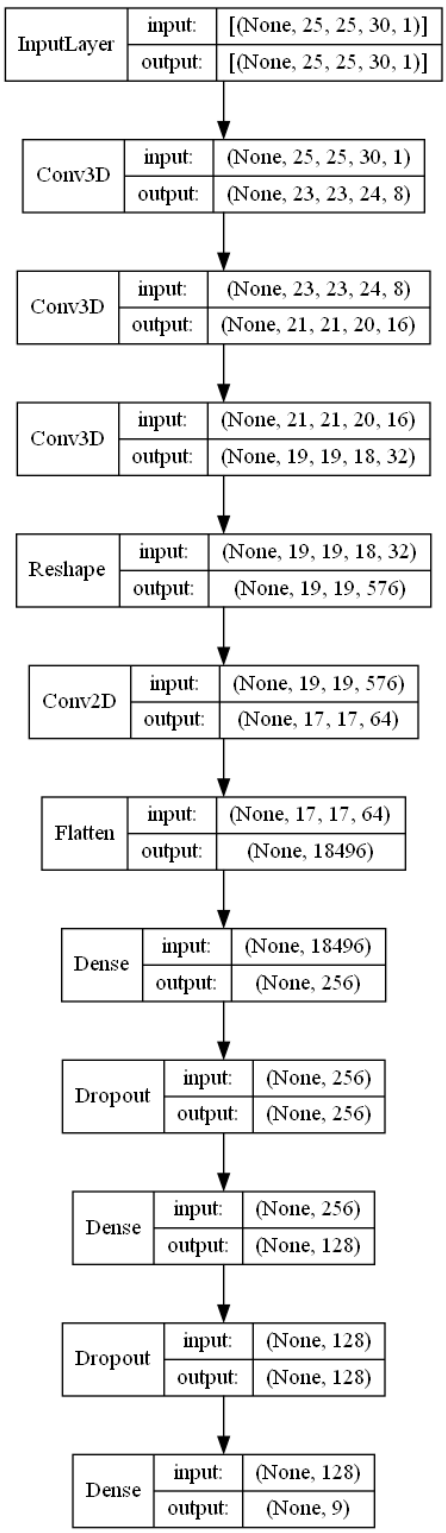


Figure 13.21: CNN architecture proposed by Roy et al. [363] as implemented in this study to establish a comparison.

Part VI

Concluding remarks

Conclusions 14

14.1 Summary of contributions 271
14.2 Future work 274

In this dissertation, the fusion of visible, multispectral and infrared imagery has been solved to shape a multi-layer system. This approach was extended to 3D point clouds by projecting the previous information to very dense RGB point clouds that were accurately reconstructed with photogrammetry. Hyperspectral data was, on the other hand, fused with visible information and projected to a 2.5D point cloud. The compression and real-time rendering of large hypercubes were similarly solved with a stack-based representation. Then, a LiDAR simulator was proposed to avoid the cumbersome tasks related to processing remotely sensed data as well as the acquisition of prohibitive technology, such as LiDAR. Therefore, this simulator enabled generating large TLS and ALS with semantic annotations and intensity from procedural and static scenarios. Both the simulation and the fusion of real data were approached with GPGPU, thus reducing considerably the response time, even outperforming commercial solutions. Finally, simulated and real data were successfully applied to several case studies, from the optimization of TLS planning to the identification of thermal anomalies and the classification of hyperspectral patches to distinguish grapevine varieties.

On a non-technical level, these last three years have considerably contributed to improving as a researcher. The work carried out has led to important results that have been published in top journals and conferences from the Remote Sensing field, while others are still under review. In addition, I have had the pleasure of contributing to other colleagues' work, and despite those publications not fitting in this dissertation, they have greatly contributed to broadening my knowledge of Computer Graphics in other fields which are not my main area of expertise.

14.1 Summary of contributions

First, a methodology for **correcting and fusing images from multiple data sources** at the image level was described. The Enhanced Correlation Coefficient was proposed as it was highly suitable for remotely sensed images sensible to different wavelength intervals. It behaved surprisingly well over datasets with significant radiometric differences. This method was massively applied to match visible, multispectral and infrared datasets. Furthermore, this algorithm was checked to highlight both the efficiency and quality of results and it provided remarkable results for images of lower dimensionality, even being blurred. In fact, missing part of the image details as well as reducing the image size helped in reducing significantly the response time while not obtaining much worse results.

The image-matching procedure represents the baseline work of the first part of this dissertation. From here, **the aligned images were projected**

into 3D point clouds. The advantages of working with point clouds rather than images are clear. For instance, they are better for visualizing a scenario at one glance, and location awareness is enhanced for human operators. However, point clouds are not as helpful if they are sparse or have geometrical errors. Therefore, a formal framework was presented for the generation of point clouds comprising visible, thermal and multispectral data. First, an RGB point cloud was reconstructed using photogrammetry, typically with hundreds of millions of points and a high point density. The rest of the datasets were projected into this RGB point cloud, thus tackling most of the drawbacks derived from applying photogrammetry over low-resolution imagery, e.g., thermal and multispectral. Additionally, this methodology outperformed commercial software in terms of 1) response time, 2) density and 3) size. Other considerations were to build an occlusion-aware projection using z-buffers and to optimize the aggregation of multiple image samples with aggregation and penalty functions. Following this approach, the aggregated radiometric data minimized the dissimilarity between the result and the initial samples.

Furthermore, this pipeline was implemented in the **GPU** to establish a fair comparison with commercial software such as Agisoft Metashape and Pix4Dmapper. Yet, the proposed methodology outperformed these commercial solutions in terms of response time. Other enhancements were checked, such as the global and local sorting of the point clouds. The conducted tests showed that globally reordering the point cloud significantly improved the response time, despite adding the latency from sorting. On the other hand, the global sorting followed by the shuffling of small groups, albeit promising, did not achieve better results for large point clouds due to the sorting overhead.

The following step was to construct **hyperspectral point clouds**, which to the best of our knowledge had not been achieved before, at least not out of a laboratory. Hyperspectral swaths were fused with visible point clouds with a different pipeline adapted to the particularities of push broom sensors. The geometrical distortions were corrected by matching swaths with an RGB orthomosaic, and then, a hyperspectral orthomosaic was composed by, again, minimizing the dissimilarity of the aggregated value and the starting collection of data. Then, this orthomosaic was projected into a 2.5D point cloud which was voxelized according to the hyperspectral GSD. The large volume of the resulting hypercube was compressed with a stack-based representation in the radiometric dimension. However, this approach also led to a higher response time during rendering because of the iterative logic implemented in *shaders*, especially for showing the last layers. This shortcoming was alleviated by constructing the images in multiple frames. In a similar way to previous work, the whole pipeline was implemented with GPGPU and therefore it was proven to be very efficient.

The tedious task of acquiring, processing and including additional data in real-world datasets was partially avoided by generating synthetic LiDAR data. **The LiDAR technology was simulated** with the ToF principle plus systematic and random errors described in the literature. Atmospheric particles, highly reflective surfaces or slope- and height-derived errors were

emulated. In addition, the sensor was parameterized to cover a wide range of commercial sensors through their specifications. Another key factor was the generation of procedural environments intended for scanning a large number of different environments and therefore, constructing large LiDAR datasets. Scenes were first labelled and linked to materials according to name patterns that facilitate this manual task. Note that this task is only performed once in procedural scenes that are later composed following pre-defined instancing rules. Then, a comparison was established on the estimation of intensity in the returns: it was first modelled with traditional shading BRDFs, and then, these were simulated using BRDF databases comprising measurements obtained from a gonio-photometer. The comparisons showed that the obtained intensity histograms were similar, whereas the latter approach is more efficient and physically accurate, at the expense of having a larger memory footprint. Nevertheless, the volume of data was considerably reduced by assuming that LiDAR's emitter and receiver had the same position. The simulation was implemented in the GPU, from the generation of LiDAR beams to scanning, to rapidly solve aerial and terrestrial surveys.

Unlike previous work, the proposed simulator includes **terrestrial and aerial surveys**, where the latter has barely been addressed. The path followed by both terrestrial and airborne vehicles can be defined, though it is especially relevant for ALS, where paths are either user-defined or automatically calculated. Airborne surveys also required the simulation of different scanning patterns, from which parallel, zigzag and elliptical patterns stand out. The implementation of multiple returns was also relevant to simulate the filtering of canopy and vegetation to construct DEMs from airborne missions. Similarly, bathymetric LiDARs was also covered to scan shallow underwater areas.

Datasets and results from previous chapters were utilized in practical case studies. First, the previous LiDAR simulation helped in the **planning for scanning at indoor scenes**, using metaheuristics guided by objective functions. These evaluation functions were designed to maximize the number of intersected triangles, minimize the number of required scans and provide a dense point cloud. In addition, they do not require the user to intervene as most of the parameters are calculated as part of the preprocessing stage. Then, **thermal maps and point clouds were applied to inspect an archaeological site**, where two anomalous areas that may correspond to buried towers were spotted. Finally, **a hyperspectral dataset was applied to the classification of grapevine varieties**. Traditional techniques, based on the correlation of spectral signatures, did not fit well on the classification of very similar spectral signatures. Instead, a Deep Learning network was explored for the classification of grapevine varieties, using state-of-the-art layers such as Inception or spatial attention. The hyperspectral swaths were radiometrically corrected and transformed to have a lower number of features with Factor Analysis. Then, these were split into patches with a size tuned during experimentation, and reduced to a single label; hence, the classification of samples was influenced by its neighbourhood. Not only this network was proven to perform well over UAS datasets,

but it also was able to obtain results close to the the-start-of-the-art overall accuracy in satellite imagery.

14.2 Future work

Remote sensing and Deep Learning fields are rapidly evolving, and yet, there exist some future investigations that should be explored from this work:

- Image matching was performed using commodity hardware, and therefore, it was explored whether downscaling and blur led to better response time. Indeed, it offered similar results with lower latency. From here, it could be explored if a pyramid scheme, from lower to higher size, helps to obtain better results while yet obtaining a low latency (see Figure 14.1). This way, larger transformations are computed from low-resolution imagery, whereas the most fine-grained changes are calculated with the starting dimensions. Besides reducing the response time, this approach could also help to match images with more notable differences.
- Image-matching was included as part of the reading phase in the comparisons established with commercial photogrammetry software. Therefore, this stage can be further accelerated by transferring the ECC algorithm, currently obtained from OpenCV, to the GPU.
- The generation of dense and large visible, thermal and multispectral point clouds helps to better visualize the scenario; however, point clouds with such a dimensionality are also harder to render in real-time. Although the compute-shader rendering alleviated this drawback [122], it could be further enhanced with different levels of detail (LOD) [111]. Similarly, it could help in the rendering of compressed hypercubes, since we already had to expand its real-time rendering over a few frames until completion.
- LiDAR optimization was studied as the simplest case study: a set of optimal locations that help to survey a scene with TLS. However, there are other kinds of LiDAR which may be worth exploring, such as those that rely on a path rather than a set of locations. Swarm-based

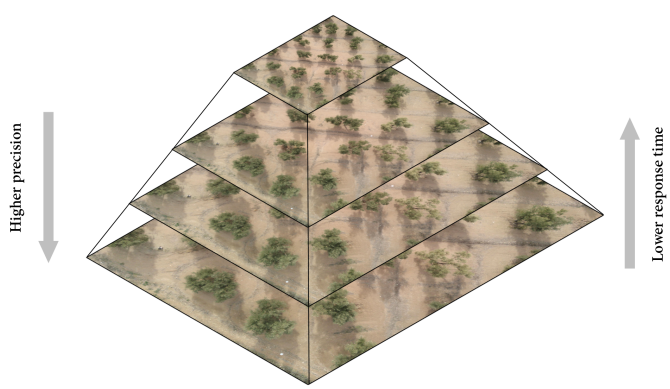


Figure 14.1: Overview of an iterative image-matching algorithm, with higher image dimensionality providing slower but more fine-grained transformations.

optimizations may fit better if a path is required [435]. Optimization of airborne LiDAR could also be explored to determine which is the optimal configuration to densely scan a scene from a DSM, including, but not limited to the followed path.

- A similar approach to the latter proposal is to assess UAS missions by computing a path that guarantees the required overlap or coverage according to the time, altitude and UAS flight speed, among other factors. This has been achieved for generic UAS missions [436], though it could be further improved based on the scene geometry, represented by rapidly sketched models. It could be especially relevant for oblique view directions, by setting it as an optimizable parameter rather than a user-defined factor.
- So far, the LiDAR simulations have helped to generate large LiDAR datasets. However, there still remain some experiments that ought to be carried out to check whether these datasets are helpful for Deep Learning tasks. Note that some of the simulated errors will be filtered out once the point clouds are voxelized, which is the most widespread approach for feeding LiDAR datasets [246, 255]. However, it should be discerned if these synthetic datasets are similar enough to the point distribution generated by real sensors.
- Part IV addressed only the simulation of LiDAR sensors. However, other sensors can be simulated by means of synthetically generated imagery. Generative Adversarial Networks (GAN), such as the one depicted in Figure 14.2, have been gaining interest as they allow building computer-made unsupervised datasets. For instance, infrared data could be simulated from visible imagery to create larger datasets. The current state-of-the-art has not successfully achieved this task by learning solely from visible data [228, 229, 437–439], since it may require further information to isolate surfaces that have different radiance.
- The phenotyping of grapevine varieties was approached with Deep Learning; however, this model could be improved by further integrating more varieties and collecting datasets at different stages of the year. Also, there are other interesting approaches to address this very same task, from multi-instance [369] to contrastive learning [368].

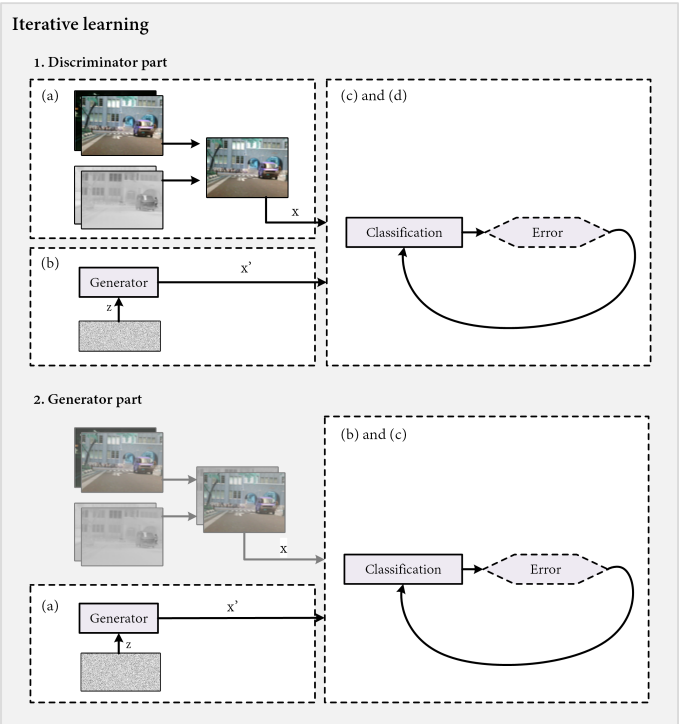


Figure 14.2: Overview of a Generative Adversarial Network composed of discriminator and generator modules. The generator is trained to produce realistic data as similar as possible to the input data, while the discriminator is trained to distinguish between real and generated data.

Part VII

Appendices

Documentación en castellano A

La actual normativa de la Universidad de Jaén para la defensa de tesis doctorales, adaptada a las directrices del R.D. 99/2011, y aprobada en Consejo de Gobierno el 6 de febrero de 2012 establece que todas las tesis doctorales escritas en un idioma distinto al castellano deberán incluir los siguientes apartados en dicho idioma: título, índice, introducción, resumen y conclusiones. En base a esta normativa se ha incluido el siguiente anexo.

Introducción

A.1 Contexto histórico y tecnología 286
A.2 Propósitos y objetivos . 289
A.3 Organización del documento 290

La teledetección ayuda a controlar, predecir y optimizar las actividades de las que dependen procesos del mundo real, y por tanto, a actuar en función de los datos analizados. Existen numerosos sensores que posibilitan esta tarea, los cuales pueden clasificarse en función de la distancia a la que operan, el intervalo de longitudes de onda al que son sensibles y el mecanismo de adquisición de datos, ya sea pasivo o activo. Las dos últimas categorías dependen del mecanismo del dispositivo, mientras que seleccionar una distancia u otra en la adquisición de datos está muy influenciado por el caso de estudio y los objetivos que se persiguen. No obstante, otros factores igualmente relevantes son el presupuesto del que se dispone, así como restricciones de tamaño del sensor, o las propias condiciones ambientales del área de interés. Es posible utilizar sensores que requieren de contacto directo con la superficie que debe monitorizarse para medir variables como humedad, temperatura o cantidad de lluvia. Sin embargo, este tipo de dispositivos necesitan mucho más mantenimiento y son más propensos a deteriorarse en condiciones ambientales adversas [1, 2]. Además, cada uno de estos sensores envía información de una posición espacial muy específica, que ocupa un lugar en una red lógica de sensores. Por tanto, lo que se hace es discretizar el espacio mediante un conjunto de dispositivos ubicados en forma de malla o en posiciones de especial interés. Por tanto, es difícil trasladar un sistema como éste a áreas de estudio de grandes dimensiones. Además, la disposición de estos sensores, tanto a nivel lógico como físico, debe ser cuidadosamente diseñada para permitir la intra y extra-comunicación utilizando redes y dispositivos que pueden tener un rango y unas capacidades muy limitadas. Por lo tanto, este tipo de monitorización se adapta mejor a aplicaciones que requieren un control constante, eficiente y en tiempo real del área de estudio.

La **teledetección** ayuda a mitigar algunas de las anteriores desventajas, de tal manera que permite cubrir áreas de grandes dimensiones utilizando plataformas aéreas, plataformas terrestres o satélites. Además, las mediciones se obtienen desde una distancia variable desde la cual se observa y adquieren datos. La teledetección se define como el arte y la ciencia de adquirir información de objetos y fenómenos sin estar en contacto directo con éstos [3]. El proceso es muy similar al seguido por el ojo humano; los datos se obtienen mediante impulsos generados por estímulos de luz, los cuales a su vez se corresponden con unas longitudes de onda muy específicas. Al menos dos componentes se encuentran involucrados en este proceso de lectura, tanto los sensores como las plataformas en las que éstos se acoplan. Los sensores serán el eje motivador de esta tesis, mientras que las plataformas pueden ser muy variadas. No obstante, inicialmente éstas se limitaban a satélites.

Las técnicas de **teledetección** evitan algunas de las limitaciones espaciales que encontramos en otros dispositivos que necesitan contacto directo con la superficie objetivo o la atmósfera.

Las imágenes de satélite son más adecuadas para monitorizar cambios a larga escala, utilizando series temporales que abarcan meses, años e

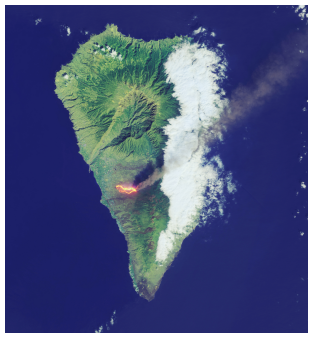


Figure A.1: Observación de la erupción del volcán Cumbre Vieja desde el satélite Landsat-8 [4].

incluso décadas. Estos datos permiten comprender la dinámica existente entre el ser humano y la naturaleza, así como el impacto de los fenómenos naturales (Figura A.1). Algunas aplicaciones de dichos conjuntos de datos son el control de uso del suelo, deforestación, cambios en la superficie terrestre y asentamientos urbanos [5]. No obstante, la resolución espacial y el periodo para revisitar un mismo punto en satélites no comerciales dificulta su aplicabilidad a tareas de monitorización que requieren de un nivel de detalle elevado (Level of detail: LOD). Este nivel de detalle puede referirse tanto la **resolución espacial** como a la **resolución temporal**, o a ambas, siendo éstas las principales **limitaciones** de las imágenes de satélite, además de su elevado coste. A pesar de ello, el uso de imágenes satélite se encuentra en aumento, como muestra la Figura A.2, debido a la cada vez menor distancia representada en un píxel (Ground Sampling Distance: GSD) así como a la disminución del periodo empleado para revisitar un mismo punto.

La Figura A.2 muestra el número de artículos en el que se emplean cada una de las plataformas que veremos en este capítulo. La búsqueda realizada en Scopus es la siguiente: $(p_1 \vee p_2 ... \vee p_n) \wedge (remote \ \wedge \ sensing)$, donde p_i es una de las plataformas que se muestra en la leyenda.

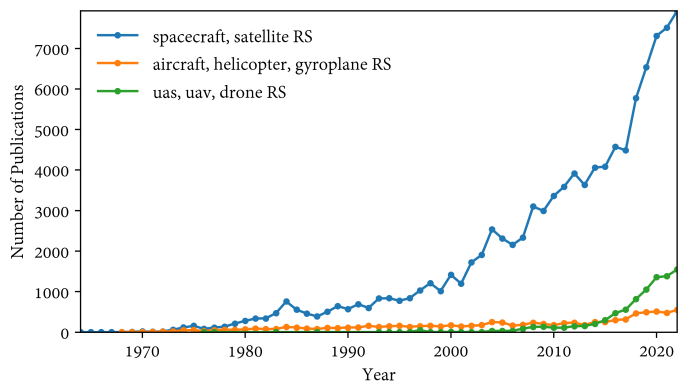


Figure A.2: Número de artículos relacionados con diferentes plataformas de teledetección.

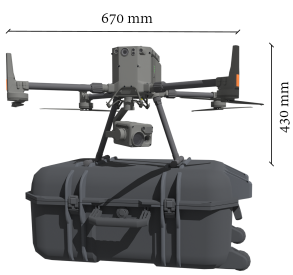


Figure A.3: Quadcóptero Matrice 300 RTK, acoplado con un dispositivo dual RGB-termográfico.

Además de las limitaciones técnicas, la resolución de las misiones de satélite se encontraba restringida hasta hace muy poco por algunos gobiernos. Dichas restricciones y limitaciones propiciaron el uso de plataformas alternativas. Además de satélites, **aeronaves** (ala fija y helicópteros), **sistemas aéreos no tripulados** (Unmanned Aerial Systems: UAS) (figura A.3) y **plataformas terrestres** (móviles y estáticas con detección próxima) son algunas de las alternativas más frecuentemente revisadas en la literatura [3]. La selección de una u otra se deberá sopesar considerando requisitos tales como maniobrabilidad, cobertura terrestre, resolución espacial, precisión espacial, coste y campo de visión (Field of View: FOV). Sin embargo, los vehículos aéreos no tripulados han copado buena parte del interés tanto en investigación como en la industria en la última década. Han pasado de ser simples herramientas militares a principios de la década de los 2000, a sistemas fáciles de desplegar, pequeños y de bajo coste, ampliando así su campo de aplicación a población civil e investigación. Estas plataformas abarcan desde aeronaves del tamaño de una mano a otras de gran tamaño

que pueden ser controladas por operadores humanos, o incluso ser parcial o totalmente autónomas. La figura A.3 muestra un dron diseñado por DJI que puede transportar hasta 2,7 kg, con unas dimensiones de $810 \times 660 \times 430$ mm.

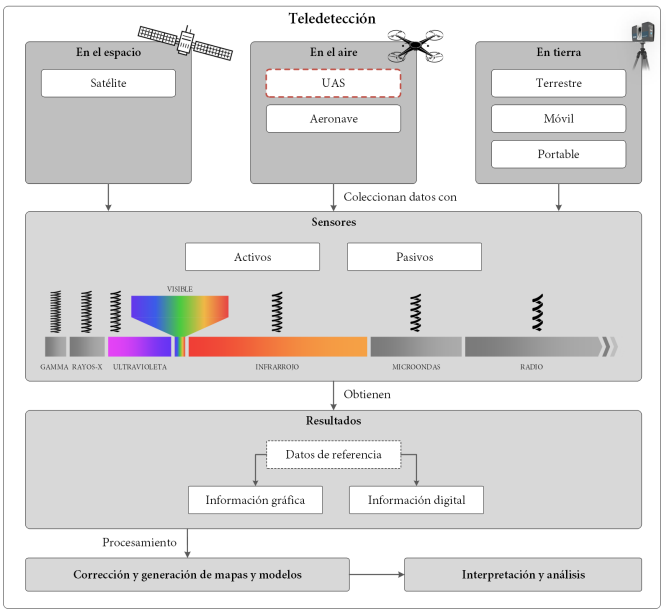


Figure A.4: Procedimiento general de adquisición de datos con plataformas y sensores de teledetección. En primer lugar, los sensores se acoplan en vehículos espaciales, aéreos o terrestres, e incluso pueden ser transportados por humanos. A continuación, se diferencian dos categorías dentro de los productos obtenidos: resultados gráficos y numéricos, aunque la mayoría puede producir ambos tipos de datos. Por último, dichos productos se procesan e interpretan para proporcionar a los usuarios información útil en el control de algún proceso.

Por tanto, los UAS son vehículos muy económicos para la adquisición de datos de teledetección. Los sensores que más comúnmente se integran son cámaras y sensores LiDAR (Light Detection and Ranging), es decir, sensores pasivos y activos, respectivamente. Mientras que los sensores activos se basan en una arquitectura transmisor-receptor, los sistemas pasivos simplemente se limitan a captar la energía emitida por las superficies observadas. Las observaciones de los sensores pasivos suelen materializarse en imágenes, mientras que los resultados de un sensor activo son frecuentemente datos que constan de tres dimensiones. De esta manera, la **tecnología activa** es mucho más **prohibitiva** en términos económicos, en comparación con algunas tipos de sensores pasivos que llevan siendo utilizados décadas y que han dado lugar incluso a la producción de versiones de menor coste dirigidas al público general. En cualquier caso, ambos tipos de tecnología pueden complementarse. Sin embargo, la adquisición de múltiples conjuntos de datos de alta resolución presenta ventajas e inconvenientes.

En primer lugar, los datos adquiridos por sensores diferentes representan características que forman parte de un **sistema de conocimiento**, donde cada una de ellas aporta información procedente de un intervalo de longitud de onda distinto. Por tanto, buena parte de los algoritmos dedicados al análisis de datos se pueden ver beneficiados de dichas características complementarias. A pesar de ello, la contribución de todas estas propiedades se podrá ponderar en un proceso de entrenamiento, en función de cómo de importante sean éstas para alcanzar una conclusión. Por tanto, se puede

La digitalización de objetos, procesos y sistemas se beneficia de las posibles reconstrucciones geométricas que pueden llevarse a cabo mediante la información recabada por sensores. De esta manera, es posible conectar réplicas virtuales y físicas a través de un canal de comunicación. Esto es lo que se conoce hoy en día como gemelo digital.

afirmar que es mejor tener un mayor número de características, salvo en aquellos casos en los que son excesivas, dando lugar al llamado problema de la maldición de la dimensionalidad. Además, los datos con una **resolución elevada** ayudan a generar modelos más **precisos**, los cuales son mucho más **fáciles de visualizar y analizar por el ser humano**. La utilización de datos con mayor densidad también deriva en una reconstrucción cuya geometría es mucha más densa, y por ende, omite menos detalles de la escena y facilita la construcción de modelos digitales, simulando procesos y entornos reales.

En una situación ideal, todas estas capas de información carecen de ruido y se representan en un mismo sistema de coordenadas. Sin embargo, los sistemas de navegación que incorporan los vehículos presentan pequeños errores de posicionamiento que pueden dificultar la fusión de datos. Estos errores se manifiestan en el espacio de la imagen mediante transformaciones relativas de traslación, rotación y escala entre fuentes de datos heterogéneas. Además, las condiciones ambientales, incluyendo composición y partículas, viento, temperatura o radiación solar, pueden variar de un vuelo a otro y por tanto, también los datos adquiridos, a pesar de utilizar planes de vuelo y sensores iguales o similares. Por tanto, estos cambios no sólo afectan a la información adquirida en una misma misión, sino que son especialmente notables en series temporales. No obstante, estas diferencias producidas por errores de posicionamiento y orientación pueden mitigarse mediante el uso de puntos de control. Para ello, es posible georeferenciar un conjunto de puntos con una precisión muy alta, y después, se indica dónde se encuentran representados dichos puntos en la imagen; por tanto, éstos deben ser fácilmente reconocibles. Otros inconvenientes son el ruido procedente de detectores defectuosos, especialmente en el caso de sensores activos, partículas atmosféricas no deseadas o superficies cuyas propiedades reflectivas impiden la observación de una geometría precisa. Todos estos factores evidencian la necesidad de definir un sistema capaz de fusionar con precisión múltiples conjuntos de datos, facilitando así su posterior procesamiento y análisis.

Además de la información adquirida directamente del sensor, pueden ser necesarios otros tipos de datos para tareas de **visión por computador**. Por ejemplo, **conocer la etiqueta semántica de un píxel** o un punto 3D sería de gran ayuda para entrenar modelos orientados a la distinción de materiales y objetos. No obstante, es difícil obtener esta información sin incorporar conocimiento previo o sin utilizar modelos ya entrenados con anterioridad. Esto implica que esta tarea se acaba desarrollando en algunos casos por operadores humanos o algoritmos no supervisados que transforman y extraen características relevantes. Sin embargo, ninguno de estos método es perfecto y pueden introducir errores que, en definitiva, derivan en modelos entrenados con datos incorrectos. Por tanto, construir conjuntos de datos de cualquier sensor **conlleva bastante tiempo**, como resultado de las etapas de adquisición, procesamiento y extracción de características. Todo esto puede evitarse, hasta cierto punto, con conjuntos de datos disponibles en repositorios públicos, los cuales crecen diariamente para dar cabida a algoritmos de Inteligencia Artificial. Éstos cubren un número

bastante amplio de sensores e intervalos de longitudes de onda, pero las propiedades adicionales previamente mencionadas siguen presentando el mismo problema. Además, nada nos garantiza que exista un conjunto de datos para nuestro caso de estudio particular.

Dadas las desventajas de adquirir datos con sensores reales, entre las que destacan el coste de los mismos y el tiempo que consume este proceso, se podría reemplazar total o parcialmente un conjunto de datos con **información sintética**. Para ello, se modela un **sensor virtual** que debe simular el comportamiento del sensor real. De esta manera, se puede añadir mucho más conocimiento en cada unidad mínima, como es el punto, y se puede resolver la tarea de generación de conjuntos de datos en mucho menos tiempo. Esta tarea requiere escenarios sintéticos cuyos modelos no presentan incertidumbre alguna, y por tanto, se pueden vincular éstos con propiedades tales como una etiqueta semántica, que en última instancia se transfieren a los datos adquiridos por el sensor virtual. Se evita así el coste de adquirir sensores tan prohibitivos como la tecnología LiDAR. En la literatura, los sensores más comúnmente simulados son Radar y LiDAR, aunque los últimos avances en Deep Learning (DL) también posibilitan la generación de imágenes sintéticas.

Los conjuntos de datos obtenidos con anterioridad suelen **transformarse** en otros tipos de **representaciones derivadas** no obtenidas de manera directa de un sensor. Por ejemplo, la identificación de anomalías térmicas podría no ser lo suficientemente trivial en el ámbito de la imagen, especialmente para situar dichas anomalías en un escenario. Una visión mucho más completa e intuitiva se deriva del uso de mapas 2D y nubes de puntos 3D. Estos resultados pueden calcularse estimando la posición y orientación de los diferentes puntos de adquisición, y mediante la búsqueda de características comunes en varias imágenes. Durante esta transformación, las propiedades radiométricas y geométricas estimadas pueden perder precisión a consecuencia de la interpolación de datos, por lo que puede medirse la precisión del resultado comparando con otros tipos de datos más fiables. Por ejemplo, las nubes de puntos LiDAR obtienen una geometría mucho más precisa que podría utilizarse para evaluar la calidad de las nubes de puntos reconstruidas mediante imágenes. Del mismo modo, las propias imágenes pueden utilizarse para medir la calidad de la información radiométrica derivada.

Por otro lado, **los grandes volúmenes de datos dificultan el procesamiento, el almacenamiento y la visualización de datos**. No es sencillo procesar miles de imágenes, o grandes nubes de puntos, en ordenadores destinados a un público general, los cuales suelen disponer de un almacenamiento muy restringido y unas capacidades de cómputo no orientadas a ofrecer un alto rendimiento. A pesar de esto, es más común encontrar hoy en día ordenadores personales y profesionales con una elevada capacidad de almacenamiento, un acceso más eficiente a datos y una mayor capacidad de cómputo y paralelismo. Este último factor dependerá de la Unidad Central de Procesamiento (Central Processing Unit: CPU), cuya tendencia es aumentar el número de núcleos, y de la Unidad de Procesamiento Gráfico (Graphics Processing Unit: GPU), compuesta por millones de pequeñas

unidades que permiten resolver tareas muy simples de manera paralela, así como intercambiar datos entre diversos hilos de manera eficiente. La sección 2.2 profundiza en los formatos de datos más comunes en teledetección; por ahora, basta con hacernos una idea de que estos datos son simplemente millones de números (Digital Number, DN). A pesar de que hoy en día el almacenamiento presenta un coste mucho más reducido, algunos inconvenientes aún persisten, como es el caso del acceso eficiente a parte de la información almacenada, especialmente en aplicaciones interactivas [9, 10].

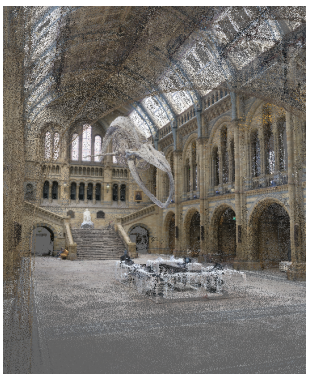


Figure A.5: Nube con 2.4 millones de puntos, reconstruida utilizando 900 imágenes adquiridas en la Sala Hintze (Modelo de *Thomas Flynn* en *Sketchfab*).

Otro inconveniente de las aplicaciones en tiempo real es la **visualización de grandes volúmenes de datos**. Los métodos tradicionales de sombreado contienen un conjunto de etapas predefinidas, en las que la geometría y la topología del modelo se transforman iterativamente para producir una imagen compuesta por píxeles. Los colores de éstos se calculan considerando la influencia de una o varias fuentes de luz sobre las texturas de una superficie, discretizando así la formulación que describe la interacción de la luz con la escena. A diferencia de los modelos sintéticos diseñados por el ser humano, los datos obtenidos por sensores reales se sombrean en función de la radiancia observada en un conjunto de longitudes de onda a las que es sensible el dispositivo. Por tanto, es necesario plantear metodologías de visualización de datos mucho más flexibles que eviten algunas etapas innecesarias, como es el sombreado de primitivas. De acuerdo con la arquitectura de una GPU, los datos pueden ordenarse y organizarse en estructuras de datos que permiten balancear dos factores: carga de trabajo y precisión geométrica. Ésta última permite alterar la percepción del usuario si se gestiona de manera adecuada. Además, estas restricciones son aún más mayores en aplicaciones de Realidad Virtual (VR) que sombrean la escena al menos dos veces, una por cada ojo, e incluso puede ser más en función de las técnicas de iluminación utilizadas.

Esta tesis tiene como principal campo de aplicación la Agricultura de Precisión y por tanto, el análisis de datos se ha acotado a 1) segmentar cultivos, y 2) clasificar variedades de éstos de manera no destructiva.

Los métodos anteriores no son más que un procedimiento para extraer información textual o visual a partir de datos de sensores. En consecuencia, el **último paso** es la **clasificación, segmentación e identificación de características** a partir de los datos de entrada. Las principales aportaciones de las técnicas de teledetección en la Agricultura de Precisión son la estimación del rendimiento de una plantación, la clasificación del tipo de cultivo, la medición del agua disponible, el índice de área foliar (Leaf Area Index, LAI) o el control de enfermedades e insectos, así como la monitorización de humedad, cambios, crecimiento, estrés y falta de agua, además de otros factores de riesgo como la nieve o el fuego [11]. Con estas tareas de vigilancia se pretende maximizar la rentabilidad y minimizar los residuos y la contaminación.

A.1 Contexto histórico y tecnología

El término **teledetección** se originó en la década de 1960, acuñado por Eveleyn Pruitt para referirse a instrumentos satelitales y aéreos que posibilitaban medir la radiancia reflejada y emitida. Por otro lado, las técnicas

topográficas fueron inicialmente diseñadas en 1849, y llevadas a la práctica en 1858 por F. Tournachon en un globo aerostático que sobrevoló Francia. A partir de aquí, el tamaño de las cámaras se ha ido reduciendo para hacerlas más cómodas de manejar. Otros medios aéreos empleados para una teledetección temprana, ya fueran producto de la investigación o de acontecimientos históricos tales como guerras, fueron cometas (1906; Figura A.6), aves (1909) y aeronaves (1908).



Figure A.6: Fotografía de San Francisco después del terremoto de 1906, capturada desde una cámara soportada por siete cometas.

La teledetección mediante satélite, tal y como se conoce hoy en día, es posible gracias a la idea inicial de Konstantin Tsiolkovsky de utilizar cohetes para explorar el espacio, publicada bajo el título *Exploring Space using jet propulsion devices*. Esta idea se vio finalmente realizada con el primer lanzamiento con éxito de un satélite, el Sputnik (1957), e igualmente contribuyó al desarrollo de satélites destinados a la vigilancia atmosférica. El satélite de observación por televisión e infrarrojos (Television and Infrared Observation Satellite: TIROS) entró en funcionamiento en 1960, integrando un pequeño sistema infrarrojo y una cámara sensible al espectro visible con un ángulo de visión muy reducido. En 1978, TIROS-N ("N" haciendo referencia a "new", nuevo) marcó un antes y un después tanto a nivel de satélite como de sensores. Esta misión integraba un radiómetro con una resolución de 1 km que podía capturar información en el espectro visible, infrarrojo cercano, infrarrojo medio e infrarrojo lejano. En la actualidad, una versión más avanzada de este satélite opera con el nombre de AVHRR-3 (Advanced Very High-Resolution Radiometer), con una resolución similar y capacidad para capturar seis bandas espectrales diferentes, en lugar de sólo cuatro [13].

Entre los proyectos de satélite actualmente operativos, Landsat es el que proporciona la mayor cantidad de datos de teledetección que se han adquirido de manera continuada. Tuvo su origen en el programa NIMBUS de la NASA, y en la actualidad hay dos satélites activos, Landsat-8 y Landsat-9, mientras que los otros siete han sido desmantelados o está previsto que así sea en los próximos años (por ejemplo, esto último ocurre con Landsat-7). La misión Landsat-9 consta de dos sensores: Operational Land Imager (OLI-2) y un sensor infrarrojo térmico (TIRS-2). Cada día obtiene un total de 740 escenas en 11 intervalos espectrales diferentes, incluyendo rojo, azul, verde, infrarrojo cercano, infrarrojo de onda corta, lejano, pancromático, costero y de cirros, con una resolución que oscila entre 15 m y 100 m. Otras misiones de satélite destacadas son el Satélite de Recursos Terrestres China-Brasil (China–Brazil Earth Resources Satellite: CBERS) y el programa Copernicus, financiado por la Comisión Europea. El satélite CBERS-4A está

Un amplio repositorio de misiones de satélite se encuentra disponible en el *Earth Observation Portal* perteneciente a la Agencia Espacial Europea [14].

equipado con tres sensores que capturan cinco intervalos diferentes con una resolución espacial entre 2 m y 55 m [18]. Por su parte, la misión Sentinel-2 es capaz de obtener datos en 13 bandas distribuidas en el espectro visible, infrarrojo cercano e infrarrojo de onda corta (ver Figura A.7), con una resolución espacial que oscila entre 10 m y 60 m [19]. El periodo de tiempo para revisitar el mismo punto de la Tierra es de diez días, en comparación con los 16 días que necesita Landsat-9 o los 31 días de CBERS-4A.

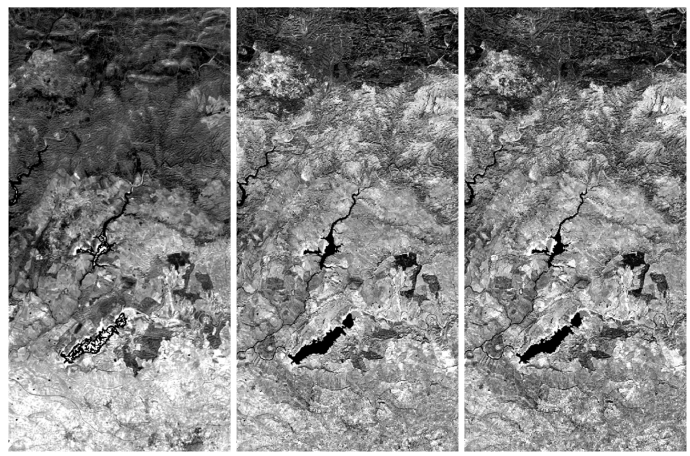


Figure A.7: Tres bandas en el infrarrojo cercano capturadas por la misión Sentinel-2 (banda 9, 935-955 µm, banda 11, 1567-1658 µm, y banda 12, 2114-2889 µm).

Por tanto, las principales limitaciones de la observación mediante satélite son la resolución temporal y espacial. No obstante, dichos inconvenientes se están mitigando en la actualidad, especialmente en misiones comerciales. Por ejemplo, el satélite comercial Pléiades Neo (VHR-2020) de Airbus Defense & Space [20] es capaz de adquirir diariamente siete bandas con una resolución de 30 cm por píxel. Incluso en misiones financiadas por entidades públicas, el periodo de repetición se reduce a menudo debido a la existencia de diversas misiones complementarias que operan con órbitas y sensores similares. De esta manera, las misiones Landsat-8 y Landsat-9 obtienen datos de un mismo punto cada 8 días [15]. Aun así, acceder a datos de satélite con una resolución elevada es mucho más prohibitivo en comparación con otras técnicas.

Además de la observación mediante satélite, el uso de UAS ha aumentado considerablemente en la última década, atrayendo así el interés tanto de la industria como de la investigación. Los componentes más frecuentes de un UAS son sensores ópticos así como sistemas de navegación y comunicación. Estos últimos permiten al operador desplazar el vehículo dentro de un rango de comunicación, y transferir datos en un canal bidireccional. Independientemente de la ruta de navegación seguida, ya sea manual o establecida mediante algunos puntos de control, se utilizan sistemas de posicionamiento global (Global Positioning System: GPS) y unidades de medición inercial (Inertial Measurement Unit: IMU) para calcular y registrar la posición, orientación y movimiento. Estos componentes son especialmente relevantes para realizar misiones con un alto grado de precisión, la cual suele indicarse mediante el error vertical y horizontal en metros (m). Por ejemplo, las últimas series de drones de DJI incluyen en sus es-

pecificaciones errores verticales y horizontales de hasta 10 cm, utilizando posicionamiento cinemático en tiempo real en lugar de GPS. Como cabría esperar, aquellos drones que integran sistemas de navegación más ligeros y precisos también son más prohibitivos.

En términos de normativa de seguridad, el término UAS no sólo se refiere al propio vehículo, sino también a los sensores acoplados. Los sensores que suelen integrarse en vehículos aéreos varían en función de la altitud de vuelo y la velocidad de la plataforma, así como de los requisitos del caso de estudio. Los vehículos de ala fija y rotatoria están mucho más limitados en altura y velocidad respecto de otras plataformas tales como helicópteros y autogiros. Aparte de las propias limitaciones de la plataforma, la altitud de vuelo puede estar limitada por las normativas correspondientes, con el fin de evitar entrar en el dominio aéreo de otros vehículos. Por ello, los sensores y las misiones que requieren menor velocidad, menor altitud de vuelo y, por tanto, mayor precisión, son especialmente convenientes para drones. Un ejemplo muy extendido de esto mismo en teledetección es la monitorización de torres y líneas de transmisión, éstas últimas con una estructura muy fina. Por tanto, este caso de estudio requiere un proceso de toma de datos mucho más lento. En comparación, otras tecnologías como InSAR (Interferometric Synthetic Aperture Radar) se han aplicado principalmente a misiones espaciales para monitorizar cambios en la superficie terrestre.

La operabilidad de las aeronaves no tripuladas en la Unión Europea está regulada por el reglamento 2019/947. Entre otras restricciones, la altitud máxima de vuelo se limita en 120 m sobre la superficie terrestre, salvo que sobrevuele un obstáculo.

A.2 Propósitos y objetivos

Dados los problemas planteados, el objetivo de esta tesis es contribuir en algunas de las etapas presentadas en la Figura A.4, tales como la fusión y el análisis de información. De entre todos los problemas presentados, destaca la adquisición de conjuntos de datos heterogéneos, la corrección geométrica y radiométrica de datos, la fusión de éstos, la generación de productos de mayor dimensionalidad (por ejemplo, pasando del 2D al 3D) o el análisis de los conjuntos de datos. Acorde a estos problemas, los objetivos específicos de esta tesis son los siguientes:

- Corrección y procesamiento de datos adquiridos desde dron. Este objetivo implica corregir tanto los errores geométricos como radiométricos en el espacio de la imagen. Las distorsiones geométricas deben resolverse en la mayoría de escenarios, dado que posibilita la fusión de datos, mientras que las correcciones radiométricas podrán llevarse a cabo o no en función del uso posterior del conjunto de datos.
- El registro de imágenes obtenidas mediante sensores diferentes, eliminando así las diferencias provocadas por la 1) distancia temporal que pudiera existir en el instante de captura, 2) posibles defectos ópticos del sensor, 3) la arquitectura del propio sistema óptico y 4) el intervalo de longitudes de onda adquirido. La fusión de imágenes debe resolverse sobre datos que presentan valores de intensidad muy diferentes, al corresponderse a longitudes de onda distintas. Como resultado de esta fusión, se obtiene una matriz de transformación que posibilita la proyección de unos datos sobre otros.

- Generación de nubes de puntos 3D de elevada densidad en un tiempo de respuesta reducido, conteniendo varias capas procedentes de múltiples conjuntos de datos. Esta metodología debe evitar algunos de los errores más comunes de las técnicas fotogramétricas, como la presencia de errores geométricos o un elevado tiempo de respuesta. Dado el gran volumen de puntos que debe abordarse, y la presencia de múltiples fuentes de datos, la generación de nubes debe resolverse mediante computación paralela.
- Los conjuntos de datos previamente adquiridos desde dron carecen de información adicional que pudiera ser relevante para aplicaciones de Inteligencia Artificial, como la etiqueta semántica de cada punto o píxel. Al elevado tiempo que conllevaría añadir este tipo de datos de forma manual, se debe añadir el tiempo derivado de la adquisición, la corrección y el procesamiento de datos. Además, existen sensores tales como el LiDAR con un coste mucho mayor que el de otros sensores, por ejemplo, cámaras térmicas o multiespectrales. Por tanto, es necesario evaluar si es posible sustituir de manera parcial, o completa, conjuntos de datos reales reales por información sintética, es decir, generadas mediante sensores virtuales. Estos últimos sensores deben simular el comportamiento del sensor real desde un punto de vista geométrico y radiométrico. Además, se trata de una tarea con un elevado coste computacional, y por tanto, debe implementarse de manera eficiente y haciendo uso de paralelismo. Por otro lado, tampoco es trivial estimar la intensidad capturada, razón por la cual deben caracterizarse las superficies modeladas en el escenario sintético.
- Por último, los productos previamente adquiridos, procesados o generados deben de tener un fin último, plasmado en una aplicación. Entre ellas se encuentra la detección de anomalías térmicas en restos arqueológicos, la clasificación de variedades de cultivos o la planificación de escaneos LiDAR. Pueden resolverse mediante técnicas puramente geométricas, o utilizando Inteligencia Artificial.

A.3 Organización del documento

Esta tesis se compone de seis partes diferentes:

PARTE I Esta parte abarca el capítulo actual, donde se introduce la terminología básica de teledetección, al mismo tiempo que se plantean algunos de los principales retos de dicha área. Después, el Capítulo 2 presenta una descripción mucho más amplia de los principales sensores, productos obtenidos y tareas de procesamiento. Por último, el Capítulo 3 contiene la revisión del estado del arte de la fusión de datos adquiridos desde dron, así como la simulación y la aplicación de éstos.

PARTE II En esta parte se detallan los conceptos básicos de este trabajo, incluyendo qué tipo de sensores y vehículos se utilizan. Después, se describe el proceso de corrección y fusión de imágenes sobre el que se sustenta la primera parte de este trabajo. Por tanto, esta sección engloba tanto

materiales como métodos que serán utilizados en capítulos posteriores.

PARTE III Esta parte presenta la generación eficiente de nubes de puntos 3D compuestas por múltiples capas: información en el espectro visible, infrarrojo lejano, multiespectral e hiperespectral. La información visible servirá para generar una nube de puntos 3D de referencia, sobre la cual se proyectará el resto de información. Esto implica el registro de diversas fuentes de información y su posterior proyección en la nube mencionada. Además, dado el tiempo de respuesta necesario, se puede llevar a cabo este proceso en la tarjeta gráfica con el fin de acelerar el proceso.

PARTE IV El procesamiento de algunos conjuntos de datos, como LiDAR o nubes de puntos estimadas con fotogrametría, no es trivial ni eficiente en términos de recursos humanos y computacionales. Por ejemplo, podríamos necesitar etiquetas semánticas sobre dichos conjuntos de datos. Por otro lado, sensores tales como el LiDAR son mucho menos económicos que buena parte de las cámaras empleadas en la Parte III. En esta sección, se propone una herramienta de simulación LiDAR sobre escenarios sintéticos, enfocada en 1) la estimación de las propiedades geométricas de una nube de puntos LiDAR (Capítulo 9) y 2) el cálculo de la radiancia observada (Capítulo 10).

PARTE V En esta parte se incluyen algunas aplicaciones derivadas del procesamiento de nubes de puntos térmicas e imágenes hiperespectrales, así como de la simulación de escaneos LiDAR. La primera aplicación es la optimización de escaneos terrestres, con el fin de indicar qué posiciones son las óptimas de acuerdo a la configuración del sensor y el entorno seleccionado. Después, se evaluarán mapas y nubes de puntos térmicas para identificar anomalías en yacimientos arqueológicos. Finalmente, la información hiperespectral se utilizar para clasificar de manera automática distintas variedades de viñedos utilizando redes neuronales.

PARTE VI El Capítulo B concluye este trabajo mostrando los principales resultados obtenidos así como posibles líneas futuras de trabajo.

Conclusiones B

B.1 Resumen de las contribuciones presentadas 293
B.2 Trabajo futuro 296

En esta tesis se han propuesto soluciones para registrar imágenes RGB, multiespectrales y térmicas, con el objetivo de alcanzar un sistema compuesto por múltiples capas de información. Después, se ha extendido este enfoque al 3D mediante la proyección de la información registrada en nubes de puntos RGB con una densidad de puntos muy elevada. También se ha logrado proyectar información hiperespectral en una nube de puntos 2.5D. Además, el hipercubo generado se ha comprimido y renderizado en tiempo real utilizando una estructura de datos basada en *stacks*. A continuación se simuló un sensor LiDAR como alternativa a la captura de datos reales, evitando así el coste de adquisición del sensor y las tareas de adquisición, procesamiento, limpieza y anotación de datos capturados. Este simulador permitió obtener conjuntos de datos muy grandes, tanto de LiDAR aéreo como terrestre, con anotaciones semánticas e intensidad. Tanto el simulador como los procesos de fusión se han implementado en la GPU, obteniendo una mejora notable de rendimiento en comparación con otras soluciones comerciales. Por último, tanto los datos reales como el simulador se han aplicado a múltiples casos de estudio, desde optimización de escaneos LiDAR, a la identificación de anomalías térmicas y la clasificación de variedades de vid en imágenes hiperespectrales.

A nivel personal, estos tres últimos años me han ayudado a mejorar como investigador. El trabajo realizado ha dado lugar a importantes resultados que han sido publicados en revistas y congresos de primer nivel en el campo de la teledetección, mientras que otros están aún en proceso de revisión. Además, he tenido el placer de participar en trabajos de otros compañeros, que, a pesar de no formar parte de la temática de esta tesis, han contribuido enormemente a aprender mucho más acerca de la informática gráfica.

B.1 Resumen de las contribuciones presentadas

En primer lugar, se propuso una metodología para **corregir y fusionar imágenes procedentes de múltiples sensores**. El algoritmo de registro de imágenes ECC se utilizó para registrar información obtenida en longitudes de onda muy diferentes. Se obtuvieron buenos resultados a pesar de las diferencias radiométricas tan significativas, y se aplicó en la fusión de imágenes visibles, multiespectrales y térmicas. Además, la eficacia de dicho algoritmo se evaluó mediante la correlación obtenida en el registro de imágenes con un tamaño menor y sobre las que se aplicaron operaciones de desenfoque. De estos últimos experimentos se concluye que tanto el desenfoque aplicado como la reducción de tamaño permiten reducir considerablemente el tiempo de respuesta obteniendo una correlación similar.

La metodología de registro de imágenes sienta la base de los primeros

capítulos de esta tesis. El siguiente paso es la **proyección de las imágenes registradas en nubes de puntos 3D**. Trabajar con nubes de puntos es mucho más intuitivo para visualizar e identificar objetos en una escena. Sin embargo, éstas deben reunir ciertas condiciones, como ser lo suficientemente densas o presentar una geometría sin errores en la reconstrucción. Por tanto, se ha definido una metodología para la generación de nubes de puntos con información RGB, térmica y multiespectral. En primer lugar, se reconstruye una nube de puntos RGB precisa y de alta densidad mediante fotogrametría, y a continuación, el resto de conjuntos de datos se proyecta sobre esta nube de puntos que garantiza unas condiciones óptimas (tanto en geometría como en densidad de puntos). De este modo, se evitan algunos de los principales inconvenientes de la fotogrametría en imágenes de baja resolución, como imágenes térmicas y multiespectrales. Además, se comparó nuestra solución con software comercial, evidenciando una mejora bastante notable en términos de 1) tiempo de respuesta, 2) densidad y 3) tamaño. Otros factores que se tuvieron en cuenta fueron la proyección de información teniendo en cuenta la oclusión mediante el uso de *z-buffers*, o la optimización de la agregación de valores procedentes de imágenes diferentes mediante el uso de funciones de agregación y funciones *penalty*. Con este último enfoque se consigue que los valores agregados minimicen la disimilitud del valor agregado respecto de las muestras iniciales.

Además, esta metodología se ha implementado en la **GPU** para compararla con soluciones comerciales como Agisoft Metashape o Pix4Dmapper. Se obtuvo un tiempo de respuesta mucho menor que el observado en estas otras soluciones. También se comprobaron otros tipos de mejoras, como la reordenación global y local de la nube de puntos. Los experimentos llevados a cabo mostraron que la reordenación global mejoraba significativamente el tiempo de respuesta, a pesar de incluir el tiempo derivado de la ordenación. Por otro lado, la ordenación local no obtuvo mejores resultados en nubes de puntos de mayor tamaño debido a la sobrecarga derivada de ordenar globalmente y después reordenar aleatoriamente grupos muy pequeños. Se concluye, por tanto, que este último enfoque podría ser más viable para procesos que se ejecutan más de una vez, y por tanto, sólo ordenan la información al comienzo.

Después se resolvió la generación de **nubes de puntos hiperespectrales**, que, hasta donde sabemos, no se había logrado anteriormente, al menos fuera de un laboratorio. Para ello, las imágenes hiperespectrales se corrigieron utilizando como referencia un ortomosaico RGB, y después, se construyó un ortomosaico hiperespectral utlilizando la optimización previamente resuelta con funciones de agregación y funciones *penalty*. Posteriormente, la información hiperespectral se proyectó sobre una nube de puntos 2.5D cuyo tamaño de vóxel se corresponde con el GSD de las imágenes hiperespectrales. Debido al gran tamaño del hipercubo resultante, éste se comprimió en la dimensión radiométrica siguiendo un enfoque basado en *stacks*. No obstante, esta compresión también dificultó la visualización en tiempo real dado que requería múltiples iteraciones por píxel en un *shader*. Este inconveniente se mitigó construyendo las imágenes en múltiples fotogramas. Al igual que en soluciones anteriores, toda la

metodología descrita se ha desarrollado en la GPU, presentando por tanto un tiempo de respuesta reducido.

A continuación, se describió un **simulador LiDAR** con el objetivo de evitar tareas tan tediosas como la adquisición, limpieza, procesamiento o etiquetado de datos. Dicho simulador permite resolver intersecciones de rayos y geometría en un tiempo de respuesta muy reducido, además de introducir errores sistemáticos y aleatorios descritos en la literatura. Estos errores se derivan de partículas atmosféricas, materiales con una reflectancia elevada, o bien se corresponden con la distancia de escaneo o la inclinación de la superficie intersectada. La simulación se ha parametrizado para integrar un amplio número de sensores LiDAR comerciales. Además, otro factor clave es la generación de escenarios procedurales que permiten obtener un gran número de escenas (y nubes de puntos) diferentes. Los modelos que conforman estas escenas se asocian con materiales y etiquetas semánticas a través de una especificación basada en el nombre de los modelos. Esta tarea sólo se lleva a cabo una vez para escenarios procedurales, y por tanto, es mucho más eficiente. Posteriormente, se ha establecido una comparativa entre dos métodos diseñados para la estimación de intensidad LiDAR. Primero, se resolvió este problema con BRDFs analíticas, como en un sombreado tradicional, y posteriormente se utilizaron BRDFs procedentes de bases de datos que albergan datos obtenidos mediante un goniofotómetro. La comparativa evidenció que el segundo método presentaba un mayor consumo de memoria, pero también es más eficiente y emplea datos de materiales reales. En cualquier caso, se redujo su consumo de memoria al máximo asumiendo que tanto el emisor como el receptor del LiDAR ocupaban la misma posición. Todo el proceso de simulación LiDAR se implementó en la GPU para resolver de manera eficiente escaneos aéreos y terrestres, desde la etapa de generación de pulsos LiDAR hasta la propia resolución de colisiones.

A diferencia de los simuladores LiDAR previamente descritos en la literatura, nuestra solución permite llevar a cabo **escaneos tanto terrestres como aéreos**. El primer tipo de simulación es el más habitual, mientras que el segundo apenas se ha descrito con anterioridad. Ambos requieren una trayectoria definida por el usuario o calculada automáticamente, mientras que las misiones aéreas también emplean diferentes patrones de escaneo, entre los que se incluyen el barrido en paralelo, en zigzag y elipsoidal. También se simulan múltiples retornos, los cuales permiten mostrar cómo se eliminan intersecciones de árboles y vegetación para obtener los puntos pertenecientes al suelo, y un LiDAR batimétrico que permite alcanzar superficies debajo del agua.

Finalmente, los conjuntos de datos y resultados obtenidos en capítulos anteriores se han empleado en diversos casos prácticos. En primer lugar, se utilizó la simulación LiDAR para diseñar un **plan de escaneo** en edificios, minimizando y optimizando las posiciones necesarias en función de la oclusión de la escena y la calidad de los retornos simulados. Este problema se resolvió mediante metaheurísticas guiadas por funciones objetivo en las que no interviene el usuario, y por tanto, se integraron fácilmente en la simulación de la que ya disponíamos. Después se han utilizado **imágenes**

y nubes de puntos térmicas para inspeccionar un yacimiento arqueológico, donde se detectaron zonas anómalas que podrían corresponderse con restos aún no explorados. Por último, se han procesado y corregido imágenes hiperespectrales para la posterior **clasificación de variedades de vid**. Las técnicas tradicionales, basadas en la correlación del perfil espectral, no son apropiadas para la clasificación de materiales cuyos perfiles espectrales son muy similares. En su lugar, se emplearon redes neuronales con capas descritas en el estado del arte, como el módulo de atención espacial. Los hipercubos se han dividido en pequeños bloques, con un tamaño que se ha ajustado durante la experimentación, y se han reducido a una sola etiqueta. Por tanto, la clasificación de cada muestra depende también de su vecindario. No sólo se ha comprobado la red propuesta sobre datos de dron, a pesar de constituir el objetivo principal, sino también sobre datos de satélite, obteniendo una precisión muy alta en ambos casos.

B.2 Trabajo futuro

La teledetección y la inteligencia artificial se encuentran en constante evolución, y derivado de ello existen algunos trabajos futuros que debieran explorarse a partir de esta tesis:

- El registro de imágenes se llevó a cabo utilizando hardware que no estaba orientado a altas prestaciones, y por lo tanto, se exploraron alternativas para mejor la eficiencia, como la reducción de tamaño o el desenfoque de las imágenes. No obstante, otra alternativa consiste en explorar un esquema piramidal que, partiendo de un tamaño muy reducido, permita obtener una transformación inicial con menor nivel de detalle (pero capaz de identificar grandes cambios). Este esquema se muestra en la Figura B.1. A medida que se incrementa el tamaño de imagen, el nivel de detalle del registro es mayor, pero los cambios son de menor entidad, y por tanto, se estiman en un menor tiempo de respuesta. Además de reducir la latencia, este enfoque también podría ayudar a registrar imágenes con diferencias mucho mayores.
- El registro de imágenes formaba parte de la fase de lectura y procesamiento de datos en las comparaciones establecidas con software

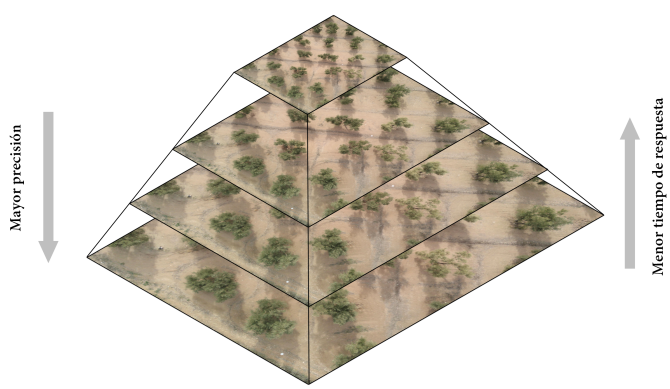


Figure B.1: Registro de imagen iterativo, donde un mayor tamaño de imagen implica mayor tiempo de respuesta y mayor precisión en la estimación de la matriz de transformación.

de fotogrametría. Por lo tanto, esta etapa podría acelerarse aún más implementando ECC en la GPU.

- La generación de nubes de puntos 3D con una densidad de puntos muy elevada, con información RGB, térmica y multiespectral; sin embargo, las nubes de puntos con una dimensionalidad tan elevada son también más difíciles de renderizar en tiempo real. Aunque el renderizado utilizando *compute shaders* ayuda considerablemente a reducir a la latencia [122], podría mejorarse aún más con diferentes niveles de detalle [111]. Del mismo modo, podría ayudar a renderizar hipercubos comprimidos, dado que uno de los principales inconvenientes encontrados fue un renderizado mucho menos eficiente que de costumbre.
- La optimización LiDAR se ha explorado en la identificación de ubicaciones óptimas para el escaneo con LiDAR terrestre. Sin embargo, hay otros tipos de tecnología LiDAR que podrían explorarse, especialmente aquellos que requieren una trayectoria. De esta manera, las optimizaciones basadas en poblaciones podrían ser de gran utilidad para el cálculo de rutas [435]. La optimización de LiDAR aéreo también podría explorarse para determinar cuál es la configuración óptima para escanear una escena específica con una densidad elevada, partiendo por ejemplo de un DSM.
- Una línea similar consiste en evaluar misiones de dron calculando una trayectoria que garantice ciertos requisitos de solapamiento y cobertura en función del tiempo, la altitud y la velocidad de vuelo, entre otros factores. Este mismo problema se ha estudiado para misiones de dron genéricas [436]; no obstante, podría mejorarse aún más en función de la geometría de la escena, representada mediante modelos rápidamente esbozados. Esto podría ser de especial interés para vuelos que requieren observaciones oblicuas, con un ángulo configurable por el usuario.
- Hasta ahora, las simulaciones LiDAR se han utilizado para generar grandes conjuntos de datos ya etiquetados. Sin embargo, los conjuntos de datos generados deben comprobarse en tareas de clasificación y segmentación mediante Deep Learning, con el fin de verificar que los conjuntos de datos de entrada podría sustituirse, al menos de manera parcial, por datos sintéticos. Nótese que algunos de los errores simulados pueden omitirse durante el proceso de voxelización de una nube de puntos [246, 255]. Por tanto, es necesario comprobar si los conjuntos de datos generados pueden realmente sustituir, al menos parcialmente, a conjuntos de datos reales.
- Además de LiDAR, hay otros sensores basados en imágenes que pueden ser simulados mediante redes generativas adversarias (Generative Adversarial Networks) como la que se muestra en la Figura B.2. Éstas permiten generar conjuntos de datos no supervisados que permitirían, por ejemplo, construir imágenes térmicas a partir de imágenes visibles con el fin de ampliar conjuntos de datos ya existentes. Los trabajos ya existentes no han conseguido resolver este problema utilizando únicamente información visible como entrada [228, 229, 437–439]. Sin embargo, esta tarea podría ayudarse de anotaciones

semánticas que permitan identificar superficies distintas, las cuales a su vez presenta información radiométrica diferente.

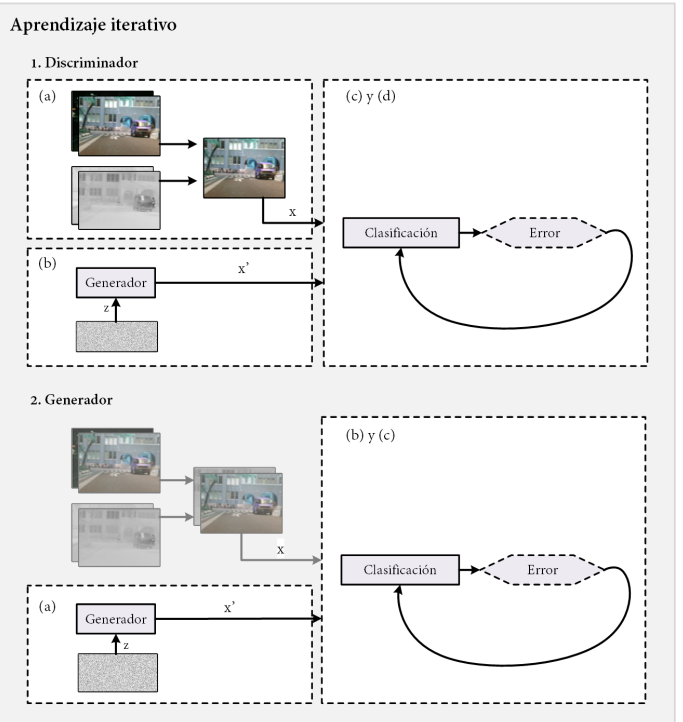


Figure B.2: Esquema de una red generativa adversaria, compuesta a su vez por dos modelos diferentes: generador y discriminador. El primero se entrena para producir datos realistas, mientras el segundo debe discernir, con la mayor eficacia posible, si los datos de entrada son reales o no.

- La clasificación de variedades de vid se resolvió con Deep Learning; sin embargo, este modelo podría mejorarse integrando más variedades y recopilando más conjuntos de datos en diferentes etapas del año. Además, existen otros enfoques interesantes para abordar esta misma tarea, desde la clasificación multi-instancia [369] hasta el aprendizaje por comparación [368].

Bibliography

[1] Nuno Silva et al. 'Low-Cost IoT LoRa®Solutions for Precision Agriculture Monitoring Practices'. In: *Progress in Artificial Intelligence*. Ed. by Paulo Moura Oliveira, Paulo Novais, and Luís Paulo Reis. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 224–235. DOI: 10.1007/978-3-030-30241-2_20 (cited on pages 9, 281).

[2] Raul Morais et al. 'A Versatile, Low-Power and Low-Cost IoT Device for Field Data Gathering in Precision Agriculture Practices'. In: *Agriculture* 11.7 (July 2021). Number: 7 Publisher: Multidisciplinary Digital Publishing Institute, p. 619. DOI: 10.3390/agriculture11070619 (cited on pages 9, 281).

[3] Thomas Lillesand, Ralph W. Kiefer, and Jonathan Chipman. *Remote Sensing and Image Interpretation, 7th Edition*. Google-Books-ID: eQXYBgAAQBAJ. Wiley, Feb. 2015 (cited on pages 9, 10, 21, 40, 281, 282).

[4] NASA Earth Observatory. *Lava Burns a Path Through La Palma*. Text.Article. Publisher: NASA Earth Observatory. Sept. 2021. URL: https://earthobservatory.nasa.gov/images/148880/lava-burns-a-path-through-la-palma (visited on 02/01/2023) (cited on pages 9, 282).

[5] Anju Asokan and J. Anitha. 'Change detection techniques for remote sensing applications: a survey'. In: *Earth Science Informatics* 12.2 (June 2019), pp. 143–160. DOI: 10.1007/s12145-019-00380-5 (cited on pages 9, 282).

[6] Charles Toth and Grzegorz Jóźków. 'Remote sensing platforms and sensors: A survey'. In: *ISPRS Journal of Photogrammetry and Remote Sensing*. Theme issue 'State-of-the-art in photogrammetry, remote sensing and spatial information science' 115 (May 2016), pp. 22–36. DOI: 10.1016/j.isprsjprs.2015.10.004 (cited on page 10).

[7] Yansheng Li, Jiayi Ma, and Yongjun Zhang. 'Image retrieval from remote sensing big data: A survey'. In: *Information Fusion* 67 (Mar. 2021), pp. 94–115. DOI: 10.1016/j.inffus.2020.10.008 (cited on page 12).

[8] Saikat Basu et al. 'DeepSat: a learning framework for satellite imagery'. In: *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. SIGSPATIAL '15. New York, NY, USA: Association for Computing Machinery, Nov. 2015, pp. 1–10. DOI: 10.1145/2820783.2820816 (cited on page 12).

[9] Juan A. Béjar-Martos et al. 'Strategies for the Storage of Large LiDAR Datasets—A Performance Comparison'. In: *Remote Sensing* 14.11 (Jan. 2022). Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, p. 2623. DOI: 10.3390/rs14112623 (cited on pages 13, 286).

[10] Carlos J. Ogayar-Anguita et al. 'Nested spatial data structures for optimal indexing of LiDAR data'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 195 (Jan. 2023), pp. 287–297. DOI: 10.1016/j.isprsjprs.2022.11.018 (cited on pages 13, 44, 53, 286).

[11] Yanbo Huang et al. 'Agricultural remote sensing big data: Management and applications'. In: *Journal of Integrative Agriculture* 17.9 (Sept. 2018), pp. 1915–1931. DOI: 10.1016/S2095-3119(17)61859-8 (cited on pages 13, 286).

[12] Bill Emery and Adriano Camps. *Introduction to Satellite Remote Sensing: Atmosphere, Ocean, Land and Cryosphere Applications*. Elsevier, Aug. 2017 (cited on pages 14, 19–21, 27, 32).

[13] National Oceanic and Atmospheric Administration. *AVHRR/3*. URL: https://www.esa.int/Applications/Observing_the_Earth/Meteorological_missions/MetOp/About_AVHRR_3 (visited on 02/01/2023) (cited on pages 14, 287).

[14] Earth Observation Portal. *Earth Observation Missions - eoPortal*. URL: https://www.eoportal.org/ (visited on 02/06/2023) (cited on pages 14, 287).

[15] Jeffrey G. Masek et al. 'Landsat 9: Empowering open science and applications through continuity'. In: *Remote Sensing of Environment* 248 (Oct. 2020), p. 111968. DOI: 10.1016/j.rse.2020.111968 (cited on pages 14, 15, 288).

[16] NASA Earth Observatory. *Landsat 9*. Nov. 2021. URL: https://landsat.gsfc.nasa.gov/satellites/landsat-9/landsat-9-overview/ (visited on 02/01/2023) (cited on page 14).

[17] U.S. Geological Survey. *Landsat 9*. USGS Numbered Series 2019-3008. Reston, VA: U.S. Geological Survey, 2019, p. 2 (cited on page 14).

[18] Instituto Nacional de Pesquisas Espaciais. *INPE/CBERS*. June 2019. URL: http://www.cbers.inpe.br/sobre/cbers04a.php (visited on 02/01/2023) (cited on pages 14, 288).

[19] European Environment Agency. *EU Digital Elevation model*. Type: File. May 2017. URL: https://land.copernicus.eu/user-corner/publications/eu-dem-flyer (visited on 12/14/2022) (cited on pages 14, 93, 241, 288).

[20] Airbus. *Pléiades Neo*. Section: Space. Oct. 2021. URL: https://www.airbus.com/en/products-services/space/earth-observation/earth-observation-portfolio/pleiades-neo (visited on 02/01/2023) (cited on pages 15, 288).

[21] Joaquim J. Sousa et al. 'UAV-Based Hyperspectral Monitoring Using Push-Broom and Snapshot Sensors: A Multisite Assessment for Precision Viticulture Applications'. In: *Sensors* 22.17 (Jan. 2022), p. 6574. DOI: 10.3390/s22176574 (cited on pages 22, 30, 31, 85).

[22] Ruiliang Pu. *Hyperspectral Remote Sensing: Fundamentals and Practices*. Boca Raton: CRC Press, Aug. 2017 (cited on pages 22, 23, 25, 75, 76, 153).

[23] Marica Franzini et al. 'Geometric and radiometric consistency of parrot sequoia multispectral imagery for precision agriculture applications'. In: *Applied Sciences* 9.24 (2019), p. 5314 (cited on page 24).

[24] João Cunha et al. 'Prediction of the Vigor and Health of Peach Tree Orchard'. In: *Computational Science and Its Applications – ICCSA 2021*. Ed. by Osvaldo Gervasi et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 541–551. DOI: 10.1007/978-3-030-86970-0_38 (cited on page 24).

[25] Melisa A. Isgró, M. Dolores Basallote, and Luis Barbero. 'Unmanned Aerial System-Based Multispectral Water Quality Monitoring in the Iberian Pyrite Belt (SW Spain)'. In: *Mine Water and the Environment* (2021), pp. 1–12 (cited on page 24).

[26] Erekle Chakhvashvili et al. 'Comparison of Reflectance Calibration Workflows for a UAV-Mounted Multi-Camera Array System'. In: *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*. July 2021, pp. 8225–8228. DOI: 10.1109/IGARSS47720.2021.9555143 (cited on page 24).

[27] J. J. Hutton et al. 'High Accuracy Direct Georeferencing of the Altum Multi-Spectral Uav Camera and its Application to High Throughput Plant Phenotyping'. In: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 43 (2020), pp. 451–456 (cited on page 24).

[28] Han Lu et al. 'Experimental Evaluation and Consistency Comparison of UAV Multispectral Minisensors'. In: *Remote Sensing* 12.16 (2020), p. 2542 (cited on page 24).

[29] Kumar Navulur. *Multispectral Image Analysis Using the Object-Oriented Paradigm*. Boca Raton: CRC Press, Dec. 2006 (cited on pages 23, 25, 39).

[30] Lei Deng et al. 'UAV-based multispectral remote sensing for precision agriculture: A comparison between different cameras'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 146 (Dec. 2018), pp. 124–136. DOI: 10.1016/j.isprsjprs.2018.09.008 (cited on page 26).

[31] J. M. Jurado et al. 'Semantic Segmentation of Natural Materials on a Point Cloud Using Spatial and Multispectral Features'. In: *Sensors* 20.8 (Jan. 2020), p. 2244. DOI: 10.3390/s20082244 (cited on page 26).

[32] Sebastian Candiago et al. 'Evaluating Multispectral Images and Vegetation Indices for Precision Farming Applications from UAV Images'. In: *Remote Sensing* 7.4 (Apr. 2015). Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, pp. 4026–4047. DOI: 10.3390/rs70404026 (cited on page 26).

[33] Michael Vollmer and Klaus-Peter Möllmann. *Infrared Thermal Imaging: Fundamentals, Research and Applications*. Wiley, Nov. 2017 (cited on pages 26–28, 35, 48, 57, 67, 80, 91, 229).

[34] Roque Alfredo Osornio-Rios, Jose Alfonso Antonino-Daviu, and Rene de Jesus Romero-Troncoso. 'Recent Industrial Applications of Infrared Thermography: A Review'. In: *IEEE Transactions on Industrial Informatics* 15.2 (Feb. 2019), pp. 615–625. DOI: 10.1109/TII.2018.2884738 (cited on page 26).

[35] D. Lőrinczy. 'Thermal analysis in biological and medical applications'. In: *Journal of Thermal Analysis and Calorimetry* 130.3 (Dec. 2017), pp. 1263–1280. DOI: 10.1007/s10973-017-6308-2 (cited on page 26).

[36] Małgorzata Jarząbek-Rychard, Dong Lin, and Hans-Gerd Maas. 'Supervised Detection of Façade Openings in 3D Point Clouds with Thermal Attributes'. In: *Remote Sensing* 12.3 (Jan. 2020), p. 543. DOI: 10.3390/rs12030543 (cited on pages 26, 29, 61, 110).

[37] Angeliki Kylili et al. 'Infrared thermography (IRT) applications for building diagnostics: A review'. In: *Applied Energy* 134 (Dec. 2014), pp. 531–549. DOI: 10.1016/j.apenergy.2014.08.005 (cited on page 26).

[38] Concepta McManus et al. 'Infrared thermography in animal production: An overview'. In: *Computers and Electronics in Agriculture* 123 (Apr. 2016), pp. 10–16. DOI: 10.1016/j.compag.2016.01.027 (cited on page 26).

[39] Dimosthenis C. Tsouros, Stamatia Bibi, and Panagiotis G. Sarigiannidis. 'A Review on UAV-Based Applications for Precision Agriculture'. In: *Information* 10.11 (Nov. 2019), p. 349. DOI: 10.3390/info10110349 (cited on page 26).

[40] Guglielmo Grechi et al. '3D Thermal Monitoring of Jointed Rock Masses through Infrared Thermography and Photogrammetry'. In: *Remote Sensing* 13.5 (Jan. 2021), p. 957. DOI: 10.3390/rs13050957 (cited on pages 26, 47, 60, 81).

[41] Rikke Gade and Thomas B. Moeslund. 'Thermal cameras and applications: a survey'. In: *Machine Vision and Applications* 25.1 (Jan. 2014), pp. 245–262. DOI: 10.1007/s00138-013-0570-5 (cited on pages 26, 28).

[42] Clare Webster et al. 'Three-dimensional thermal characterization of forest canopies using UAV photogrammetry'. In: *Remote Sensing of Environment* 209 (May 2018), pp. 835–847. DOI: 10.1016/j.rse.2017.09.033 (cited on pages 27, 60, 63, 81, 124).

[43] Francisco Yandun Narvaez et al. 'A Survey of Ranging and Imaging Techniques for Precision Agriculture Phenotyping'. In: *IEEE/ASME Transactions on Mechatronics* 22.6 (Dec. 2017), pp. 2428–2439. DOI: 10.1109/TMECH.2017.2760866 (cited on pages 27, 109).

[44] Andres Montes de Oca and Gerardo Flores. 'A UAS equipped with a thermal imaging system with temperature calibration for Crop Water Stress Index computation'. In: *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. June 2021, pp. 714–720. DOI: 10.1109/ICUAS51884.2021.9476863 (cited on page 27).

[45] P. J. Zarco-Tejada et al. 'Previsual symptoms of Xylella fastidiosa infection revealed in spectral plant-trait alterations'. In: *Nature Plants* 4.7 (July 2018), pp. 432–439. DOI: 10.1038/s41477-018-0189-7 (cited on pages 27, 122).

[46] Wenan Yuan and Daeun Choi. 'UAV-Based Heating Requirement Determination for Frost Management in Apple Orchard'. In: *Remote Sensing* 13.2 (Jan. 2021), p. 273. DOI: 10.3390/rs13020273 (cited on pages 27, 29).

[47] Waldemar Minkina. 'How Infrared Radiation Was Discovered—Range of This Discovery and Detailed, Unknown Information'. In: *Applied Sciences* 11.21 (Jan. 2021). Number: 21 Publisher: Multidisciplinary Digital Publishing Institute, p. 9824. DOI: 10.3390/app11219824 (cited on page 27).

[48] E F J Ring. 'The discovery of infrared radiation in 1800'. In: *The Imaging Science Journal* 48.1 (Jan. 2000). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/13682199.2000.11784339, pp. 1–8. DOI: 10.1080/13682199.2000.11784339 (cited on page 27).

[49] W. Minkina and S. Dudzik. *Infrared Thermography: Errors and Uncertainties*. Wiley, 2009 (cited on page 27).

[50] Othón González et al. 'Thermal Radiation Dynamics of Soil Surfaces with Unmanned Aerial Systems'. In: *Pattern Recognition*. Ed. by Jesús Ariel Carrasco-Ochoa et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 183–192. DOI: 10.1007/978-3-030-21077-9_17 (cited on pages 28, 57, 60, 61, 81).

[51] Dale A. Quattrochi and Jeffrey C. Luvall. 'Thermal infrared remote sensing for analysis of landscape ecological processes: methods and applications'. In: *Landscape Ecology* 14.6 (Dec. 1999), pp. 577–598. DOI: 10.1023/A:1008168910634 (cited on pages 28, 57).

[52] Alfonso López et al. 'An optimized approach for generating dense thermal point clouds from UAV-imagery'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 182 (Dec. 2021), pp. 78–95. DOI: 10.1016/j.isprsjprs.2021.09.022 (cited on page 29).

[53] Hyeon Jeong Jo et al. 'Dense Thermal 3D Point Cloud Generation of Building Envelope by Drone-based Photogrammetry'. In: *Journal of the Korean Society of Surveying, Geodesy, Photogrammetry and Cartography* 39.2 (2021), pp. 73–79. DOI: 10.7848/ksgpc.2021.39.2.73 (cited on pages 29, 45, 61).

[54] Joanna Paziewska and Antoni Rzonca. 'Integration of Thermal and RGB Data Obtained by Means of a Drone for Interdisciplinary Inventory'. In: *Energies* 15.14 (Jan. 2022). Number: 14 Publisher: Multidisciplinary Digital Publishing Institute, p. 4971. DOI: 10.3390/en15144971 (cited on page 29).

[55] Lu Jiang et al. 'Diurnal variations in directional brightness temperature over urban areas through a multi-angle UAV experiment'. In: *Building and Environment* 222 (Aug. 2022), p. 109408. DOI: 10.1016/j.buildenv.2022.109408 (cited on page 29).

[56] V. Guilbert et al. 'Fusion of thermal and visible point clouds: application to the Vaches Noires landslide, Normandy, France'. In: vol. 43. 2020, pp. 227–232. DOI: 10.5194/isprs-archives-XLIII-B2-2020-227-2020 (cited on pages 29, 60).

[57] Lorenzo Comba et al. '2D and 3D data fusion for crop monitoring in precision agriculture'. In: *2019 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*. Oct. 2019, pp. 62–67. DOI: 10.1109/MetroAgriFor.2019.8909219 (cited on pages 29, 63).

[58] Farid Javadnejad et al. 'A photogrammetric approach to fusing natural colour and thermal infrared UAS imagery in 3D point cloud generation'. In: *International Journal of Remote Sensing* 41.1 (Jan. 2020), pp. 211–237. DOI: 10.1080/01431161.2019.1641241 (cited on pages 29, 48, 57, 58, 60, 62, 80, 81, 110, 116–118, 124).

[59] A. Adán et al. 'Fusion of thermal imagery and LiDAR data for generating TBIM models'. In: *2017 IEEE SENSORS*. Oct. 2017, pp. 1–3. DOI: 10.1109/ICSENS.2017.8234261 (cited on pages 29, 58, 60).

[60] Dong Lin et al. 'Fusion of thermal imagery with point clouds for building façade thermal attribute mapping'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 151 (May 2019), pp. 162–175. DOI: 10.1016/j.isprsjprs.2019.03.010 (cited on pages 29, 58, 60–63, 109).

[61] Patrick Westfeld, David Mader, and Hans-Gerd Maas. 'Generation of TIR-attributed 3D Point Clouds from UAV-based Thermal Imagery'. In: *Photogrammetrie - Fernerkundung - Geoinformation* 2015.5 (Nov. 2015). Place: Stuttgart, Germany Publisher: Schweizerbart Science Publishers, pp. 381–393. DOI: 10.1127/1432-8364/2015/0274 (cited on page 29).

[62] R. Boesch. 'Thermal remote sensing with UAV-based workflows'. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XLII-2-W6. Copernicus GmbH, Aug. 2017, pp. 41–46. DOI: 10.5194/isprs-archives-XLII-2-W6-41-2017 (cited on pages 29, 61).

[63] A. Sledz, J. Unger, and C. Heipke. 'Thermal IR imaging: Image quality and orthophoto generation'. In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives 42 (2018), Nr. 1* 42.1 (2018), pp. 413–420. DOI: 10.15488/4082 (cited on pages 29, 57, 60, 61, 110).

[64] Luís Pádua et al. 'Vineyard Variability Analysis through UAV-Based Vigour Maps to Assess Climate Change Impacts'. In: *Agronomy* 9.10 (Oct. 2019), p. 581. DOI: 10.3390/agronomy9100581 (cited on pages 29, 56).

[65] Jose Manuel Amigo and Vladlena Benson. *Hyperspectral Imaging: Volume 32*. Amsterdam, Netherlands, Oct. 2019 (cited on pages 29, 76).

[66] Sandra Jakob, Robert Zimmermann, and Richard Gloaguen. 'The Need for Accurate Geometric and Radiometric Corrections of Drone-Borne Hyperspectral Data for Mineral Exploration: MEPHySTo—A Toolbox for Pre-Processing Drone-Borne Hyperspectral Data'. In: *Remote Sensing* 9.1 (Jan. 2017), p. 88. DOI: 10.3390/rs9010088 (cited on page 30).

[67] Telmo Adão et al. 'Hyperspectral Imaging: A Review on UAV-Based Sensors, Data Processing and Applications for Agriculture and Forestry'. In: *Remote Sensing* 9.11 (Nov. 2017). Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, p. 1110. DOI: 10.3390/rs9111110 (cited on pages 30, 60).

[68] Juan-Pablo Ramirez-Paredes, David J. Lary, and Nicholas R. Gans. 'Low-altitude Terrestrial Spectroscopy from a Pushbroom Sensor'. In: *Journal of Field Robotics* 33.6 (2016), pp. 837–852. DOI: 10.1002/rob.21624 (cited on page 30).

[69] Juan M. Jurado et al. 'An Efficient Method for Generating UAV-Based Hyperspectral Mosaics Using Push-Broom Sensors'. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 14 (2021). Conference Name: IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, pp. 6515–6531. DOI: 10.1109/JSTARS.2021.3088945 (cited on pages 30, 60, 153).

[70] Qingsheng Xue et al. 'Compact, UAV-mounted hyperspectral imaging system with automatic geometric distortion rectification'. In: *Optics Express* 29.4 (Feb. 2021), pp. 6092–6112. DOI: 10.1364/OE.412853 (cited on page 30).

[71] Mojtaba Akhoundi Khezrabad, Mohammad Javad Valadan Zoej, and Alireza Safdarinezhad. 'A new approach for geometric correction of UAV-based pushbroom images through the processing of simultaneously acquired frame images'. In: *Measurement* 199 (Aug. 2022), p. 111431. DOI: 10.1016/j.measurement.2022.111431 (cited on pages 30, 60, 131).

[72] Arko Lucieer et al. 'HyperUAS—Imaging Spectroscopy from a Multirotor Unmanned Aircraft System'. In: *Journal of Field Robotics* 31.4 (2014), pp. 571–590. DOI: 10.1002/rob.21508 (cited on pages 30, 31, 37).

[73] Helge Aasen et al. 'Quantitative Remote Sensing at Ultra-High Resolution with UAV Spectroscopy: A Review of Sensor Technology, Measurement Procedures, and Data Correction Workflows'. In: *Remote Sensing* 10.7 (July 2018), p. 1091. DOI: 10.3390/rs10071091 (cited on page 30).

[74] Vasit Sagan et al. 'Data-Driven Artificial Intelligence for Calibration of Hyperspectral Big Data'. In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–20. DOI: 10.1109/TGRS.2021.3091409 (cited on pages 30, 31, 37).

[75] Si-Bo Duan et al. 'Land Surface Reflectance Retrieval from Hyperspectral Data Collected by an Unmanned Aerial Vehicle over the Baotou Test Site'. In: *PLOS ONE* 8.6 (June 2013), e66972. DOI: 10.1371/journal.pone.0066972 (cited on page 30).

[76] Guijun Yang et al. 'The DOM Generation and Precise Radiometric Calibration of a UAV-Mounted Miniature Snapshot Hyperspectral Imager'. In: *Remote Sensing* 9.7 (July 2017), p. 642. DOI: 10.3390/rs9070642 (cited on page 30).

[77] California Institute of Technology. *AVIRIS Data Portal*. URL: https://aviris.jpl.nasa.gov/dataportal/ (visited on 11/17/2022) (cited on pages 31, 37).

[78] Natalie Queally et al. 'FlexBRDF: A Flexible BRDF Correction for Grouped Processing of Airborne Imaging Spectroscopy Flightlines'. In: *Journal of Geophysical Research: Biogeosciences* 127.1 (2022), e2021JG006622. DOI: 10.1029/2021JG006622 (cited on pages 31, 37).

[79] Wen Jia et al. 'A Kernel-Driven BRDF Approach to Correct Airborne Hyperspectral Imagery over Forested Areas with Rugged Topography'. In: *Remote Sensing* 12.3 (Jan. 2020), p. 432. DOI: 10.3390/rs12030432 (cited on pages 31, 37).

[80] Temuulen T. Sankey et al. 'UAV hyperspectral and lidar data and their fusion for arid and semi-arid land vegetation monitoring'. In: *Remote Sensing in Ecology and Conservation* 4.1 (2018), pp. 20–33. DOI: 10.1002/rse2.44 (cited on pages 31, 65).

[81] Long Gao et al. 'CBFF-Net: A New Framework for Efficient and Accurate Hyperspectral Object Tracking'. In: *IEEE Transactions on Geoscience and Remote Sensing* (2023). Conference Name: IEEE Transactions on Geoscience and Remote Sensing, pp. 1–1. DOI: 10.1109/TGRS.2023.3253173 (cited on page 31).

[82] Lloyd Windrim et al. 'Unsupervised ore/waste classification on open-cut mine faces using close-range hyperspectral data'. In: *Geoscience Frontiers* 14.4 (July 2023), p. 101562. DOI: 10.1016/j.gsf.2023.101562 (cited on page 31).

[83] Jibo Yue et al. 'A Comparison of Crop Parameters Estimation Using Images from UAV-Mounted Snapshot Hyperspectral Sensor and High-Definition Digital Camera'. In: *Remote Sensing* 10.7 (July 2018). Number: 7 Publisher: Multidisciplinary Digital Publishing Institute, p. 1138. DOI: 10.3390/rs10071138 (cited on page 31).

[84] Pinliang Dong and Qi Chen. *LiDAR Remote Sensing and Applications*. CRC Press, Jan. 2018 (cited on pages 32, 33, 175).

[85] M. Hasan Shariq and Ben Richard Hughes. 'Revolutionising building inspection techniques to meet large-scale energy demands: A review of the state-of-the-art'. In: *Renewable and Sustainable Energy Reviews* 130 (Sept. 2020), p. 109979. DOI: 10.1016/j.rser.2020.109979 (cited on pages 32, 71, 206).

[86] Emilia Guisado-Pintado, Derek W. T. Jackson, and David Rogers. '3D mapping efficacy of a drone and terrestrial laser scanner over a temperate beach-dune zone'. In: *Geomorphology* 328 (Mar. 2019), pp. 157–172. DOI: 10.1016/j.geomorph.2018.12.013 (cited on pages 32, 71).

[87] Helena Mitasova et al. 'Geospatial analysis of vulnerable beach-foredune systems from decadal time series of lidar data'. In: *Journal of Coastal Conservation* 14.3 (Sept. 2010), pp. 161–172. DOI: 10.1007/s11852-010-0088-1 (cited on pages 32, 71).

[88] Sampo Kuutti et al. 'A Survey of Deep Learning Applications to Autonomous Vehicle Control'. In: *IEEE Transactions on Intelligent Transportation Systems* 22.2 (Feb. 2021), pp. 712–733. DOI: 10.1109/TITS.2019.2962338 (cited on pages 32, 71).

[89] Mesrop Andriasyan et al. 'From Point Cloud Data to Building Information Modelling: An Automatic Parametric Workflow for Heritage'. In: *Remote Sensing* 12.7 (Mar. 2020), p. 1094. DOI: 10.3390/rs12071094 (cited on pages 32, 71, 206).

[90] Yuzhou Zhou et al. 'Street-view imagery guided street furniture inventory from mobile laser scanning point clouds'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 189 (July 2022), pp. 63–77. DOI: 10.1016/j.isprsjprs.2022.04.023 (cited on page 32).

[91] Bernhard Höfle and Norbert Pfeifer. 'Correction of laser scanning intensity data: Data and model-driven approaches'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 62.6 (Dec. 2007), pp. 415–433. DOI: 10.1016/j.isprsjprs.2007.05.008 (cited on page 33).

[92] Dimitrios Bolkas and Aaron Martinez. 'Effect of target color and scanning geometry on terrestrial LiDAR point-cloud noise and plane fitting'. In: *Journal of Applied Geodesy* 12.1 (Jan. 2018), pp. 109–127. DOI: 10.1515/jag-2017-0034 (cited on page 33).

[93] S. Bechtold and B. Höfle. 'Helios: a Multi-Purpose LIDAR Simulation Framework for Research, Planning and Training of Laser Scanning Operations with Airborne, Ground-Based Mobile and Stationary Platforms'. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* III3 (June 2016), pp. 161–168. DOI: 10.5194/isprs-annals-III-3-161-2016 (cited on pages 34, 69, 192).

[94] Lukas Winiwarter et al. 'Virtual laser scanning with HELIOS++: A novel take on ray tracing-based simulation of topographic full-waveform 3D laser scanning'. In: *Remote Sensing of Environment* 269 (Feb. 2022), p. 112772. DOI: 10.1016/j.rse.2021.112772 (cited on pages 34, 69).

[95] Jonathan Dupuy and Wenzel Jakob. 'An adaptive parameterization for efficient material acquisition and rendering'. In: *ACM Transactions on Graphics* 37.6 (Dec. 2018), 274:1–274:14. DOI: 10.1145/3272127.3275059 (cited on pages 36, 192, 193, 197, 200).

[96] Rosana Montes Soldado and Carlos Ureña Almagro. *An Overview of BRDF Models*. Mar. 2012 (cited on pages 36, 195, 196).

[97] Nicolas Riviere, Romain Ceolato, and Laurent Hespel. 'Multispectral polarized BRDF: Design of a highly resolved reflectometer and development of a data inversion method'. In: *Optica Applicata* 42 (Jan. 2012). DOI: 10.5277/oa120101 (cited on page 36).

[98] Borom Tunwattanapong et al. 'Acquiring reflectance and shape from continuous spherical harmonic illumination'. In: *ACM Transactions on Graphics* 32.4 (July 2013), 109:1–109:12. DOI: 10.1145/2461912.2461944 (cited on page 36).

[99] Guojun Chen et al. 'Reflectance scanning: estimating shading frame and BRDF with generalized linear light sources'. In: *ACM Transactions on Graphics* 33.4 (July 2014), pp. 1–11. DOI: 10.1145/2601097.2601180 (cited on page 36).

[100] Wojciech Matusik et al. 'A data-driven reflectance model'. In: *ACM Transactions on Graphics* 22.3 (July 2003), pp. 759–769. DOI: 10.1145/882262.882343 (cited on pages 36, 75, 192).

[101] Ana Serrano et al. 'An intuitive control space for material appearance'. In: *ACM Transactions on Graphics* 35.6 (Dec. 2016), 186:1–186:12. DOI: 10.1145/2980179.2980242 (cited on pages 36, 192).

[102] Juan M. Jurado et al. 'An efficient method for acquisition of spectral BRDFs in real-world scenarios'. In: *Computers & Graphics* 102 (Feb. 2022), pp. 154–163. DOI: 10.1016/j.cag.2021.08.021 (cited on pages 37, 158).

[103] D. Guarnera et al. 'BRDF Representation and Acquisition'. In: *Computer Graphics Forum* 35.2 (2016), pp. 625–650. DOI: 10.1111/cgf.12867 (cited on pages 37, 38, 195).

[104] Rosana Montes Soldado and Carlos Ureña Almagro. 'An Overview of BRDF Models'. In: Mar. 2012 (cited on page 38).

[105] Cambridge English Dictionary. *Cambridge University Press*. Feb. 2023. URL: https://dictionary.cambridge.org/dictionary/english/ (visited on 02/12/2023) (cited on page 39).

[106] Oxford University Press. *Oxford Learner's Dictionaries*. Mar. 2023. URL: https://www.oxfordlearnersdictionaries.com/ (visited on 04/13/2023) (cited on page 39).

[107] Chao Cao, Marius Preda, and Titus Zaharia. '3D Point Cloud Compression: A Survey'. In: *The 24th International Conference on 3D Web Technology*. Web3D '19. New York, NY, USA: Association for Computing Machinery, July 2019, pp. 1–9. DOI: 10.1145/3329714.3338130 (cited on pages 41–43).

[108] T. Barros et al. 'Multispectral vineyard segmentation: A deep learning comparison study'. In: *Computers and Electronics in Agriculture* 195 (Apr. 2022), p. 106782. DOI: 10.1016/j.compag.2022.106782 (cited on pages 41, 78).

[109] Mohamed Kerkech et al. 'Vine Disease Detection by Deep Learning Method Combined with 3D Depth Information'. In: *Image and Signal Processing*. Ed. by Abderrahim El Moataz et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 82–90. DOI: 10.1007/978-3-030-51935-3_9 (cited on pages 41, 78).

[110] Ana I. De Castro et al. '3-D Characterization of Vineyards Using a Novel UAV Imagery-Based OBIA Procedure for Precision Viticulture Applications'. In: *Remote Sensing* 10.4 (Apr. 2018), p. 584. DOI: 10.3390/rs10040584 (cited on page 41).

[111] Markus Schütz et al. *GPU-Accelerated LOD Generation for Point Clouds*. arXiv:2302.14801 [cs]. Feb. 2023. DOI: 10.48550/arXiv.2302.14801. URL: http://arxiv.org/abs/2302.14801 (visited on 03/01/2023) (cited on pages 41, 44, 50, 274, 297).

[112] Peter van Oosterom et al. 'Organizing and visualizing point clouds with continuous levels of detail'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 194 (Dec. 2022), pp. 119–131. DOI: 10.1016/j.isprsjprs.2022.10.004 (cited on pages 41, 45).

[113] José L. Cárdenas et al. 'Reconstruction of tree branching structures from UAV-LiDAR data'. In: *Frontiers in Environmental Science* 10 (2022) (cited on page 42).

[114] NVIDIA. *NVIDIA RTX A6000*. URL: https://www.nvidia.com/es-es/design-visualization/rtx-a6000/ (visited on 02/28/2023) (cited on page 42).

[115] NVIDIA. *NVIDIA GeForce RTX 40 Series*. URL: https://www.nvidia.com/en-us/geforce/graphics-cards/40-series/ (visited on 02/28/2023) (cited on page 42).

[116] NVIDIA. *NVIDIA GeForce RTX 30 Series*. URL: https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/ (visited on 02/28/2023) (cited on page 42).

[117] Mai Bui et al. 'Comparative Study of 3D Point Cloud Compression Methods'. In: *2021 IEEE International Conference on Big Data (Big Data)*. Dec. 2021, pp. 5859–5861. DOI: 10.1109/BigData52589.2021.9671822 (cited on page 42).

[118] Zizheng Que, Guo Lu, and Dong Xu. 'VoxelContext-Net: An Octree Based Framework for Point Cloud Compression'. In: 2021, pp. 6042–6051 (cited on page 42).

[119] Alejandro Graciano et al. 'QuadStack: An Efficient Representation and Direct Rendering of Layered Datasets'. In: *IEEE Transactions on Visualization and Computer Graphics* 27.9 (Sept. 2021), pp. 3733–3744. DOI: 10.1109/TVCG.2020.2981565 (cited on pages 43, 151, 156, 161).

[120] Tomas Akenine-Möller et al. *Real-Time Rendering 4th Edition*. Boca Raton, FL, USA: A K Peters/CRC Press, 2018 (cited on pages 43, 45, 134, 172, 196).

[121] Markus Schutz and Michael Wimmer. 'Rendering Point Clouds with Compute Shaders'. In: *SIGGRAPH Asia 2019 Posters*. SA '19. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 1–2. DOI: 10.1145/3355056.3364554 (cited on pages 43, 45).

[122] Markus Schütz, Bernhard Kerbl, and Michael Wimmer. 'Rendering Point Clouds with Compute Shaders and Vertex Order Optimization'. In: *Computer Graphics Forum* 40.4 (2021), pp. 115–126. DOI: 10.1111/cgf.14345 (cited on pages 43–45, 50, 110, 132, 146, 157, 274, 297).

[123] Markus Schütz, Bernhard Kerbl, and Michael Wimmer. 'Software Rasterization of 2 Billion Points in Real Time'. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 5.3 (July 2022), 24:1–24:17. DOI: 10.1145/3543863 (cited on pages 45, 50).

[124] Steven White et al. *Cascaded Shadow Maps*. Apr. 2021. URL: https://learn.microsoft.com/en-us/windows/win32/dxtecharts/cascaded-shadow-maps (visited on 11/09/2022) (cited on page 45).

[125] Juan M. Jurado et al. 'An out-of-core method for GPU image mapping on large 3D scenarios of the real world'. In: *Future Generation Computer Systems* 134 (Sept. 2022), pp. 66–77. DOI: 10.1016/j.future.2022.03.022 (cited on page 45).

[126] J. M. Jurado et al. 'Multispectral Mapping on 3D Models and Multi-Temporal Monitoring for Individual Characterization of Olive Trees'. In: *Remote Sensing* 12.7 (Jan. 2020). Number: 7 Publisher: Multidisciplinary Digital Publishing Institute, p. 1106. DOI: 10.3390/rs12071106 (cited on pages 46, 56, 64).

[127] Sebastian Schneider et al. 'Fusing vision and LIDAR - Synchronization, correction and occlusion reasoning'. In: *2010 IEEE Intelligent Vehicles Symposium*. IEEE, June 2010. DOI: 10.1109/ivs.2010.5548079 (cited on page 46).

[128] Yasutaka Furukawa and Carlos Hernández. *Multi-View Stereo: A Tutorial*. Boston Delft, May 2015 (cited on page 47).

[129] Haichao Zheng et al. 'A Thermal Performance Detection Method for Building Envelope Based on 3D Model Generated by UAV Thermal Imagery'. In: *Energies* 13.24 (Jan. 2020), p. 6677. DOI: 10.3390/en13246677 (cited on pages 47, 60, 61).

[130] Ghassan Al Lafi et al. '3D Thermal and Spatial Modeling of a Subway Tunnel: A Case Study'. In: (June 2017), pp. 386–394. DOI: 10.1061/9780784480823.046 (cited on pages 47, 60, 64).

[131] San Jiang, Cheng Jiang, and Wanshou Jiang. 'Efficient structure from motion for large-scale UAV images: A review and a comparison of SfM tools'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 167 (Sept. 2020), pp. 230–251. DOI: 10.1016/j.isprsjprs.2020.04.016 (cited on pages 47, 48, 132).

[132] Ludwig Hoegner, Sebastian Tuttas, and Uwe Stilla. '3D building reconstruction and construction site monitoring from RGB and TIR image sets'. In: *2016 12th IEEE International Symposium on Electronics and Telecommunications (ISETC)*. Oct. 2016, pp. 305–308. DOI: 10.1109/ISETC.2016.7781118 (cited on pages 48, 57).

[133] Humberto Bustince et al. 'On the definition of penalty functions in data aggregation'. In: *Fuzzy Sets and Systems* 323 (Sept. 2017). Publisher: Elsevier BV, pp. 1–18. DOI: 10.1016/j.fss.2016.09.011 (cited on page 48).

[134] Humberto Bustince, Javier Fernandez, and Pedro Burillo. 'Penalty Function in Optimization Problems: A Review of Recent Developments'. In: *Soft Computing Based Optimization and Decision Models*. Springer International Publishing, Aug. 2017, pp. 275–287. DOI: 10.1007/978-3-319-64286-4_17 (cited on page 48).

[135] Daniel Paternain, Aranzazu Jurio, and Gleb Beliakov. 'Color image reduction by minimizing penalty functions'. In: *2012 IEEE International Conference on Fuzzy Systems*. IEEE, June 2012, pp. 1–7. DOI: 10.1109/fuzz-ieee.2012.6250794 (cited on pages 48, 118).

[136] D. Paternain et al. 'Construction of image reduction operators using averaging aggregation functions'. In: *Fuzzy Sets and Systems*. Theme: Aggregation operators 261 (Feb. 2015), pp. 87–111. DOI: 10.1016/j.fss.2014.03.008 (cited on page 48).

[137] J. Jakob and M. Guthe. 'Optimizing LBVH-Construction and Hierarchy-Traversal to accelerate kNN Queries on Point Clouds using the GPU'. In: *Computer Graphics Forum* 40.1 (2021), pp. 124–137. DOI: https://doi.org/10.1111/cgf.14177 (cited on pages 49, 119).

[138] Jan Eric Lenssen, Christian Osendorfer, and Jonathan Masci. 'Deep Iterative Surface Normal Estimation'. In: 2020, pp. 11247–11256 (cited on page 49).

[139] Jin Zeng et al. 'Deep Surface Normal Estimation With Hierarchical RGB-D Fusion'. In: 2019, pp. 6153–6162 (cited on page 49).

[140] Alexandre Boulch and Renaud Marlet. 'Deep Learning for Robust Normal Estimation in Unstructured Point Clouds'. In: *Computer Graphics Forum* 35.5 (2016), pp. 281–290. DOI: 10.1111/cgf.12983 (cited on page 49).

[141] Josef Stumpfegger et al. 'GPU accelerated scalable parallel coordinates plots'. In: *Computers & Graphics* 109 (Dec. 2022), pp. 111–120. DOI: 10.1016/j.cag.2022.10.008 (cited on page 50).

[142] Nakhoon Baek and Kwan-Hee Yoo. 'An accelerated rendering scheme for massively large point cloud data'. In: *The Journal of Supercomputing* 76.10 (Oct. 2020), pp. 8313–8323. DOI: 10.1007/s11227-019-03114-y (cited on page 50).

[143] Diego Royo et al. 'Non-line-of-sight transient rendering'. In: *Computers & Graphics* 107 (Oct. 2022), pp. 84–92. DOI: 10.1016/j.cag.2022.07.003 (cited on page 50).

[144] Daniel Meister et al. 'A Survey on Bounding Volume Hierarchies for Ray Tracing'. In: *Computer Graphics Forum* 40.2 (2021), pp. 683–712. DOI: 10.1111/cgf.142662 (cited on pages 51, 52).

[145] Morgan McGuire. *Computer Graphics Archive*. July 2017 (cited on pages 51, 52).

[146] Daniel Meister and Jiri Bittner. 'Parallel Locally-Ordered Clustering for Bounding Volume Hierarchy Construction'. In: *IEEE Transactions on Visualization and Computer Graphics* 24.3 (Mar. 2018), pp. 1345–1353. DOI: 10.1109/tvcg.2017.2669983 (cited on pages 51, 52, 116, 170).

[147] Hubert Nguyen. *GPU Gems 3*. First. Addison-Wesley Professional, 2007 (cited on pages 52, 137).

[148] J. Hendrich et al. 'Ray Classification for Accelerated BVH Traversal'. In: *Computer Graphics Forum* 38.4 (July 2019), pp. 49–56. DOI: 10.1111/cgf.13769 (cited on page 52).

[149] J. Revelles, C. Ureña, and M. Lastra. 'An Efficient Parametric Algorithm for Octree Traversal'. In: *Journal of WSCG*. 2000, pp. 212–219 (cited on page 53).

[150] Artur L. dos Santos et al. 'kD-Tree Traversal Implementations for Ray Tracing on Massive Multiprocessors: A Comparative Study'. In: *2009 21st International Symposium on Computer Architecture and High Performance Computing*. ISSN: 1550-6533. Oct. 2009, pp. 41–48. DOI: 10.1109/SBAC-PAD.2009.25 (cited on page 53).

[151] John Amanatides and Andrew Woo. 'A Fast Voxel Traversal Algorithm for Ray Tracing'. In: (1987). Accepted: 2015-10-05T07:55:23Z Publisher: Eurographics Association. DOI: 10.2312/egtp.19871000 (cited on page 53).

[152] Tero Karras. 'Maximizing parallelism in the construction of BVHs, octrees, and k-d trees'. In: *Proceedings of the Fourth ACM SIGGRAPH / Eurographics conference on High-Performance Graphics*. EGGH-HPG'12. Goslar, DEU: Eurographics Association, June 2012, pp. 33–37 (cited on page 53).

[153] T. Poblete et al. 'Discriminating Xylella fastidiosa from Verticillium dahliae infections in olive trees using thermal- and hyperspectral-based plant traits'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 179 (Sept. 2021), pp. 133–144. DOI: 10.1016/j.isprsjprs.2021.07.014 (cited on page 56).

[154] Danilo Roberti Alves de Almeida et al. 'Monitoring restored tropical forest diversity and structure through UAV-borne hyperspectral and lidar fusion'. In: *Remote Sensing of Environment* 264 (Oct. 2021), p. 112582. DOI: 10.1016/j.rse.2021.112582 (cited on page 56).

[155] Nathalie Guimarães et al. 'Forestry Remote Sensing from Unmanned Aerial Vehicles: A Review Focusing on the Data, Processing and Potentialities'. In: *Remote Sensing* 12.6 (Jan. 2020). Number: 6 Publisher: Multidisciplinary Digital Publishing Institute, p. 1046. DOI: 10.3390/rs12061046 (cited on page 56).

[156] Kai Heckel et al. 'Predicting Forest Cover in Distinct Ecosystems: The Potential of Multi-Source Sentinel-1 and -2 Data Fusion'. In: *Remote Sensing* 12.2 (Jan. 2020), p. 302. DOI: 10.3390/rs12020302 (cited on page 56).

[157] Felix Schiefer et al. 'Mapping forest tree species in high resolution UAV-based RGB-imagery by means of convolutional neural networks'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 170 (Dec. 2020), pp. 205–215. DOI: 10.1016/j.isprsjprs.2020.10.015 (cited on page 56).

[158] Maitiniyazi Maimaitijiang et al. 'Crop Monitoring Using Satellite/UAV Data Fusion and Machine Learning'. In: *Remote Sensing* 12.9 (Jan. 2020), p. 1357. DOI: 10.3390/rs12091357 (cited on page 56).

[159] André Vong et al. 'How to Build a 2D and 3D Aerial Multispectral Map?—All Steps Deeply Explained'. In: *Remote Sensing* 13.16 (Jan. 2021), p. 3227. DOI: 10.3390/rs13163227 (cited on page 56).

[160] Yu Hou et al. 'Fusing tie points' RGB and thermal information for mapping large areas based on aerial images: A study of fusion performance under different flight configurations and experimental conditions'. In: *Automation in construction* 124 (2021), p. 103554. DOI: 10.1016/j.autcon.2021.103554 (cited on page 57).

[161] D. Stojcsics et al. 'High Resolution 3D Thermal Imaging Using FLIR DUO R Sensor'. In: 2018, pp. 000311–000316. DOI: 10.1109/INES.2018.8523914 (cited on pages 57, 60).

[162] L. Hoegner et al. 'Evaluation of methods for coregistration and fusion of RPAS-based 3D point clouds and thermal infrared images'. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XLI-B3. Copernicus GmbH, June 2016, pp. 241–246. DOI: 10.5194/isprs-archives-XLI-B3-241-2016 (cited on pages 57, 58).

[163] M. Landmann et al. 'Multimodal sensor: High-speed 3D and thermal measurement'. In: vol. 11144. 2019. DOI: 10.1117/12.2531950 (cited on page 58).

[164] Ludwig Hoegner et al. 'Fusion of TLS and RGB point clouds with TIR images for indoor mobile mapping'. In: Berlin, Jan. 2018. DOI: 10.21611/qirt.2018.019 (cited on pages 58, 60).

[165] Xiaoyong Shen et al. 'Multi-modal and Multi-spectral Registration for Natural Images'. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 309–324. DOI: 10.1007/978-3-319-10593-2_21 (cited on page 59).

[166] Jyun-Ping Jhan, Jiann-Yeou Rau, and Cho-Ying Huang. 'Band-to-band registration and ortho-rectification of multilens/multispectral imagery: A case study of MiniMCA-12 acquired by a fixed-wing UAS'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 114 (2016). Publisher: Elsevier, pp. 66–77 (cited on page 59).

[167] JP Jhan et al. 'Investigation of parallax issues for multi-lens multispectral camera band co-registration'. In: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 42 (2017). Publisher: Copernicus GmbH, p. 157 (cited on page 59).

[168] Amin Sedaghat and Nazila Mohammadi. 'High-resolution image registration based on improved SURF detector and localized GTM'. In: *International Journal of Remote Sensing* 40.7 (Apr. 2019), pp. 2576–2601. DOI: 10.1080/01431161.2018.1528402 (cited on page 59).

[169] Sajid Saleem and Robert Sablatnig. 'A robust SIFT descriptor for multispectral images'. In: *IEEE signal processing letters* 21.4 (2014), pp. 400–403 (cited on page 59).

[170] Chung-Hsien Tsai and Yu-Ching Lin. 'An accelerated image matching technique for UAV orthoimage registration'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 128 (June 2017), pp. 130–145. DOI: 10.1016/j.isprsjprs.2017.03.017 (cited on page 59).

[171] Yoseline Angel et al. 'Automated Georectification and Mosaicking of UAV-Based Hyperspectral Imagery from Push-Broom Sensors'. In: *Remote Sensing* 12.1 (Jan. 2020). Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, p. 34. DOI: 10.3390/rs12010034 (cited on page 60).

[172] Jingwei Zhu et al. 'Fusion of urban 3D point clouds with thermal attributes using MLS data and TIR image sequences'. In: *Infrared Physics & Technology* 113 (Mar. 2021), p. 103622. DOI: 10.1016/j.infrared.2020.103622 (cited on pages 60, 62, 63).

[173] Antonio Adán et al. 'Towards the Use of 3D Thermal Models in Constructions'. In: *Sustainability* 12.20 (Jan. 2020), p. 8521. DOI: 10.3390/su12208521 (cited on pages 60, 63, 81).

[174] Mitra Dahaghin, Farhad Samadzadegan, and Farzaneh Dadrass Javan. 'Precise 3D extraction of building roofs by fusion of UAV-based thermal and visible images'. In: *International Journal of Remote Sensing* 42.18 (Sept. 2021), pp. 7002–7030. DOI: 10.1080/01431161.2021.1951875 (cited on pages 60, 61).

[175] M. Dahaghin, F. Samadzadegan, and F. Dadras Javan. '3D thermal mapping of building roofs based on fusion of thermal and visible point clouds in uav imagery'. In: vol. 42. 2019, pp. 271–277. DOI: 10.5194/isprs-archives-XLII-4-W18-271-2019 (cited on pages 60, 63).

[176] V. V. Kniaz and V. A. Mizginov. 'Thermal Texture Generation and 3d Model Reconstruction Using SFM and Gan'. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 422 (May 2018), pp. 519–524. DOI: 10.5194/isprs-archives-XLII-2-519-2018 (cited on page 60).

[177] L. Hoegner and U. Stilla. 'Mobile thermal mapping for matching of infrared images with 3D building models and 3D point clouds'. In: *Quantitative InfraRed Thermography Journal* 15.2 (July 2018), pp. 252–270. DOI: 10.1080/17686733.2018.1455129 (cited on pages 60, 63).

[178] H. Macher et al. 'Combination of thermal and geometric information for BIM enrichment'. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XLII-2-W15. Copernicus GmbH, Aug. 2019, pp. 719–725. DOI: 10.5194/isprs-archives-XLII-2-W15-719-2019 (cited on pages 60, 64).

[179] E. Maset et al. 'PHOTOGRAMMETRIC 3D BUILDING RECONSTRUCTION FROM THERMAL IMAGES'. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. IV-2-W3. Copernicus GmbH, Aug. 2017, pp. 25–32. DOI: 10.5194/isprs-annals-IV-2-W3-25-2017 (cited on page 60).

[180] Gregory Clarkson, Shan Luo, and Raul Fuentes. 'Thermal 3D modelling'. In: *ISARC Proceedings* (July 2017), pp. 493–499 (cited on pages 60, 63).

[181] Yingying Kong et al. '3-D Point Cloud Reconstruction of Infrared Images Based on Improved Structure from Motion'. In: *2018 2nd European Conference on Electrical Engineering and Computer Science (EECS)*. Dec. 2018, pp. 307–310. DOI: 10.1109/EECS.2018.00063 (cited on page 61).

[182] Abdul Nishar et al. 'Thermal infrared imaging of geothermal environments and by an unmanned aerial vehicle (UAV): A case study of the Wairakei – Tauhara geothermal field, Taupo, New Zealand'. In: *Renewable Energy* 86 (Feb. 2016), pp. 1256–1264. DOI: 10.1016/j.renene.2015.09.042 (cited on page 61).

[183] Wouter H. Maes, Alfredo R. Huete, and Kathy Steppe. 'Optimizing the Processing of UAV-Based Thermal Imagery'. In: *Remote Sensing* 9.5 (May 2017), p. 476. DOI: 10.3390/rs9050476 (cited on page 61).

[184] F. Hosoi, S. Umeyama, and K. Kuo. 'Estimating 3D chlorophyll content distribution of trees using an image fusion method between 2D camera and 3D portable scanning lidar'. In: *Remote Sensing* 11.18 (2019). DOI: 10.3390/rs11182134 (cited on page 62).

[185] J. Zhu et al. 'Direct co-registration of TIR images and MLS point clouds by corresponding keypoints'. In: vol. 4. 2019, pp. 235–242. DOI: 10.5194/isprs-annals-IV-2-W7-235-2019 (cited on pages 62, 63).

[186] Trong Phuc Truong et al. 'Registration of RGB and Thermal Point Clouds Generated by Structure From Motion'. In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. Oct. 2017, pp. 419–427. DOI: 10.1109/ICCVW.2017.57 (cited on page 63).

[187] Haiping Wang et al. 'You Only Hypothesize Once: Point Cloud Registration with Rotation-equivariant Descriptors'. In: *Proceedings of the 30th ACM International Conference on Multimedia*. MM '22. New York, NY, USA: Association for Computing Machinery, Oct. 2022, pp. 1630–1641. DOI: 10.1145/3503161.3548023 (cited on page 63).

[188] Jan Maria Juszczyk et al. 'Wound 3D Geometrical Feature Estimation Using Poisson Reconstruction'. In: *IEEE Access* 9 (2021), pp. 7894–7907. DOI: 10.1109/ACCESS.2020.3035125 (cited on page 63).

[189] Yishuo Huang, Chih-Hung Chiang, and Keng-Tsang Hsu. 'Combining the 3D model generated from point clouds and thermography to identify the defects presented on the facades of a building'. In: *Nondestructive Characterization and Monitoring of Advanced Materials, Aerospace, Civil Infrastructure, and Transportation XII*. Vol. 10599. International Society for Optics and Photonics, Mar. 2018, 105990G. DOI: 10.1117/12.2297656 (cited on page 64).

[190] Douglas K. Bolton et al. 'Optimizing Landsat time series length for regional mapping of lidar-derived forest structure'. In: *Remote Sensing of Environment* 239 (Mar. 2020), p. 111645. DOI: 10.1016/j.rse.2020.111645 (cited on page 64).

[191] Alex M. Lechner, Giles M. Foody, and Doreen S. Boyd. 'Applications in Remote Sensing to Forest Ecology and Management'. In: *One Earth* 2.5 (May 2020), pp. 405–412. DOI: 10.1016/j.oneear.2020.05.001 (cited on page 64).

[192] Krishna Karthik Gadiraju et al. 'Multimodal Deep Learning Based Crop Classification Using Multispectral and Multitemporal Satellite Imagery'. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '20. New York, NY, USA: Association for Computing Machinery, Aug. 2020, pp. 3234–3242. DOI: 10.1145/3394486.3403375 (cited on page 64).

[193] Muhammad Usman Qadeer et al. 'Spatio-Temporal Crop Classification On Volumetric Data'. In: *2021 IEEE International Conference on Image Processing (ICIP)*. ISSN: 2381-8549. Sept. 2021, pp. 3812–3816. DOI: 10.1109/ICIP42928.2021.9506046 (cited on page 64).

[194] Zouhair Ballouch, Rafika Hajji, and Mohamed Ettarid. 'Toward a Deep Learning Approach for Automatic Semantic Segmentation of 3D Lidar Point Clouds in Urban Areas'. In: *Geospatial Intelligence: Applications and Future Trends*. Ed. by Fatimazahra Barramou et al. Advances in Science, Technology & Innovation. Cham: Springer International Publishing, 2022, pp. 67–77. DOI: 10.1007/978-3-030-80458-9_6 (cited on page 64).

[195] Ekrem Saralioglu and Oguz Gungor. 'Semantic segmentation of land cover from high resolution multispectral satellite images by spectral-spatial convolutional neural network'. In: *Geocarto International* 0.0 (Mar. 2020), pp. 1–21. DOI: 10.1080/10106049.2020.1734871 (cited on page 64).

[196] Shengxi Gui and Rongjun Qin. 'Automated LoD-2 model reconstruction from very-high-resolution satellite-derived digital surface model and orthophoto'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 181 (Nov. 2021), pp. 1–19. DOI: 10.1016/j.isprsjprs.2021.08.025 (cited on page 64).

[197] Rongjun Qin. 'A critical analysis of satellite stereo pairs for digital surface model generation and a matching quality prediction model'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 154 (Aug. 2019), pp. 139–150. DOI: 10.1016/j.isprsjprs.2019.06.005 (cited on page 64).

[198] Alvaro Gómez et al. 'An Experimental Comparison of Multi-View Stereo Approaches on Satellite Images'. In: 2022, pp. 844–853 (cited on page 64).

[199] Corinne Stucker and Konrad Schindler. 'ResDepth: A deep residual prior for 3D reconstruction from high-resolution satellite images'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 183 (Jan. 2022), pp. 560–580. DOI: 10.1016/j.isprsjprs.2021.11.009 (cited on page 64).

[200] Y. Han et al. 'State of the art in digital surface modelling from multi-view high resolution satellite images'. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. V-2-2020. Copernicus GmbH, Aug. 2020, pp. 351–356. DOI: 10.5194/isprs-annals-V-2-2020-351-2020 (cited on page 64).

[201] Mathias Rothermel et al. 'Photometric multi-view mesh refinement for high-resolution satellite images'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 166 (Aug. 2020), pp. 52–62. DOI: 10.1016/j.isprsjprs.2020.05.001 (cited on page 64).

[202] Huajian Liu, Sang-Heon Lee, and Javaan Singh Chahl. 'Registration of multispectral 3D points for plant inspection'. In: *Precision Agriculture* 19.3 (June 2018), pp. 513–536. DOI: 10.1007/s11119-017-9536-3 (cited on page 64).

[203] K. Zainuddin et al. '3D modeling for rock art documentation using lightweight multispectral camera'. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XLII-2-W9. ISSN: 1682-1750. Copernicus GmbH, Jan. 2019, pp. 787–793. DOI: 10.5194/isprs-archives-XLII-2-W9-787-2019 (cited on page 64).

[204] Xin Shen et al. 'Estimation of Forest Structural Attributes Using Spectral Indices and Point Clouds from UAS-Based Multispectral and RGB Imageries'. In: *Remote Sensing* 11.7 (Jan. 2019), p. 800. DOI: 10.3390/rs11070800 (cited on page 64).

[205] Juan Villacrés and Fernando A. Auat Cheein. 'Construction of 3D maps of vegetation indices retrieved from UAV multispectral imagery in forested areas'. In: *Biosystems Engineering* 213 (Jan. 2022), pp. 76–88. DOI: 10.1016/j.biosystemseng.2021.11.025 (cited on page 64).

[206] Lorenzo Comba et al. 'Unsupervised detection of vineyards by 3D point-cloud UAV photogrammetry for precision agriculture'. In: *Computers and Electronics in Agriculture* 155 (Dec. 2018), pp. 84–95. DOI: 10.1016/j.compag.2018.10.005 (cited on pages 64, 65).

[207] Thibault Clamens et al. 'Real-time Multispectral Image Processing and Registration on 3D Point Cloud for Vineyard Analysis'. In: *16th International Conference on Computer Vision Theory and Applications*. Vienna, Austria, Feb. 2021. DOI: 10.5220/0010266203880398 (cited on page 64).

[208] Rubén Valbuena. 'Integrating Airborne Laser Scanning with Data from Global Navigation Satellite Systems and Optical Sensors'. In: *Forestry Applications of Airborne Laser Scanning: Concepts and Case Studies*. Ed. by Matti Maltamo, Erik Næsset, and Jari Vauhkonen. Managing Forest Ecosystems. Dordrecht: Springer Netherlands, 2014, pp. 63–88. DOI: 10.1007/978-94-017-8663-8_4 (cited on page 65).

[209] Joel B. Sankey et al. 'Quantifying plant-soil-nutrient dynamics in rangelands: Fusion of UAV hyperspectral-LiDAR, UAV multispectral-photogrammetry, and ground-based LiDAR-digital photography in a shrub-encroached desert grassland'. In: *Remote Sensing of Environment* 253 (Feb. 2021), p. 112223. DOI: 10.1016/j.rse.2020.112223 (cited on page 65).

[210] YanFeng Gu et al. 'UAV-based integrated multispectral-LiDAR imaging system and data processing'. In: *Science China Technological Sciences* 63.7 (July 2020), pp. 1293–1301. DOI: 10.1007/s11431-019-1571-0 (cited on page 65).

[211] Qinan Lin et al. 'Detection of Pine Shoot Beetle (PSB) Stress on Pine Forests at Individual Tree Level using UAV-Based Hyperspectral Imagery and Lidar'. In: *Remote Sensing* 11.21 (Jan. 2019). Number: 21 Publisher: Multidisciplinary Digital Publishing Institute, p. 2540. DOI: 10.3390/rs11212540 (cited on page 65).

[212] Ali Zia et al. '3D Reconstruction from Hyperspectral Images'. In: *2015 IEEE Winter Conference on Applications of Computer Vision*. Jan. 2015, pp. 318–325. DOI: 10.1109/WACV.2015.49 (cited on page 65).

[213] Min H. Kim et al. '3D imaging spectroscopy for measuring hyperspectral patterns on solid objects'. In: *ACM Transactions on Graphics* 31.4 (July 2012), 38:1–38:11. DOI: 10.1145/2185520.2185534 (cited on page 65).

[214] Jan Behmann et al. 'Calibration of hyperspectral close-range pushbroom cameras for plant phenotyping'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 106 (Aug. 2015), pp. 172–182. DOI: 10.1016/j.isprsjprs.2015.05.010 (cited on page 65).

[215] Jan Behmann et al. 'Generation and application of hyperspectral 3D plant models: methods and challenges'. In: *Machine Vision and Applications* 27.5 (July 2016), pp. 611–624. DOI: 10.1007/s00138-015-0716-8 (cited on page 66).

[216] Huajian Liu et al. 'Hyperspectral imaging and 3D technologies for plant phenotyping: From satellite to close-range sensing'. In: *Computers and Electronics in Agriculture* 175 (Aug. 2020), p. 105621. DOI: 10.1016/j.compag.2020.105621 (cited on page 66).

[217] I. Kalisperakis et al. 'Leaf area index estimation in vineyards from UAV hyperspectral data, 2D image mosaics and 3D canopy surface models'. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XL-1-W4. ISSN: 1682-1750. Copernicus GmbH, Aug. 2015, pp. 299–303. DOI: 10.5194/isprsarchives-XL-1-W4-299-2015 (cited on page 66).

[218] Juan I. Nieto, Sildomar T. Monteiro, and Diego Viejo. '3D geological modelling using laser and hyperspectral data'. In: *2010 IEEE International Geoscience and Remote Sensing Symposium*. ISSN: 2153-7003. July 2010, pp. 4568–4571. DOI: 10.1109/IGARSS.2010.5651553 (cited on page 66).

[219] Maxime Ferrera et al. 'Hyperspectral 3D Mapping of Underwater Environments'. In: *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. ISSN: 2473-9944. Oct. 2021, pp. 3696–3705. DOI: 10.1109/ICCVW54120.2021.00413 (cited on page 66).

[220] Luís Pádua et al. 'UAS, sensors, and data processing in agroforestry: a review towards practical applications'. In: *International Journal of Remote Sensing* 38.8-10 (May 2017). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/01431161.2017.1297548, pp. 2349–2391. DOI: 10.1080/01431161.2017.1297548 (cited on page 66).

[221] Brooke Bruning et al. 'Approaches, applications, and future directions for hyperspectral vegetation studies: An emphasis on yield-limiting factors in wheat'. In: *The Plant Phenome Journal* 3.1 (2020). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/ppj2.20007, e20007. DOI: 10.1002/ppj2.20007 (cited on page 66).

[222] Olli Nevalainen et al. 'Individual Tree Detection and Classification with UAV-Based Photogrammetric Point Clouds and Hyperspectral Imaging'. In: *Remote Sensing* 9.3 (Mar. 2017). Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, p. 185. DOI: 10.3390/rs9030185 (cited on page 66).

[223] Roberto Garzonio et al. 'Surface Reflectance and Sun-Induced Fluorescence Spectroscopy Measurements Using a Small Hyperspectral UAS'. In: *Remote Sensing* 9.5 (May 2017). Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, p. 472. DOI: 10.3390/rs9050472 (cited on page 66).

[224] Thomas Astor et al. 'Vegetable Crop Biomass Estimation Using Hyperspectral and RGB 3D UAV Data'. In: *Agronomy* 10.10 (Oct. 2020). Number: 10 Publisher: Multidisciplinary Digital Publishing Institute, p. 1600. DOI: 10.3390/agronomy10101600 (cited on page 66).

[225] Pablo Horstrand et al. 'A UAV Platform Based on a Hyperspectral Sensor for Image Capturing and On-Board Processing'. In: *IEEE Access* 7 (2019). Conference Name: IEEE Access, pp. 66919–66938. DOI: 10.1109/ACCESS.2019.2913957 (cited on page 66).

[226] Heikki Saari et al. 'Visible, very near IR and short wave IR hyperspectral drone imaging system for agriculture and natural water applicationS: Frontiers in Spectral imaging and 3D Technologies for Geospatial Solutions, ISPRS SPEC3D'. In: *Frontiers in Spectral imaging and 3D Technologies for Geospatial Solutions*. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (Oct. 2017). Ed. by E. Honkavaara et al. Publisher: International Society for Photogrammetry and Remote Sensing ISPRS, pp. 165–170. DOI: 10.5194/isprs-archives-XLII-3-W3-165-2017 (cited on page 66).

[227] Jinkang Cai et al. 'A Survey on Data-Driven Scenario Generation for Automated Vehicle Testing'. In: *Machines* 10.11 (Nov. 2022), p. 1101. DOI: 10.3390/machines10111101 (cited on pages 67, 68).

[228] Xing Yi et al. 'Cycle Generative Adversarial Network Based on Gradient Normalization for Infrared Image Generation'. In: *Applied Sciences* 13.1 (Jan. 2023). Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, p. 635. DOI: 10.3390/app13010635 (cited on pages 67, 275, 297).

[229] Mehmet Akif Özkanoğlu and Sedat Ozer. 'InfraGAN: A GAN architecture to transfer visible images to infrared domain'. In: *Pattern Recognition Letters* 155 (Mar. 2022), pp. 69–76. DOI: 10.1016/j.patrec.2022.01.026 (cited on pages 67, 275, 297).

[230] Rishi Mohan and Bram de Jager. 'Robust Optimal Sensor Planning for Occlusion Handling in Dynamic Robotic Environments'. In: *IEEE Sensors Journal* 19.11 (June 2019), pp. 4259–4270. DOI: 10.1109/JSEN.2019.2899929 (cited on page 67).

[231] Dongsheng Li et al. '3D model-based scan planning for space frame structures considering site conditions'. In: *Automation in Construction* 140 (Aug. 2022), p. 104363. DOI: 10.1016/j.autcon.2022.104363 (cited on pages 67, 72, 207).

[232] G. Lee, J. Cheon, and I. Lee. 'VALIDATION OF LIDAR CALIBRATION USING A LIDAR SIMULATOR'. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLIII-B1-2020 (Aug. 2020), pp. 39–44. DOI: 10.5194/isprs-archives-XLIII-B1-2020-39-2020 (cited on page 67).

[233] Jawad Iqbal et al. 'Simulation of an Autonomous Mobile Robot for LiDAR-Based In-Field Phenotyping and Navigation'. In: *Robotics* 9.2 (June 2020), p. 46. DOI: 10.3390/robotics9020046 (cited on page 67).

[234] Fredrik Westling, Mitch Bryson, and James Underwood. 'SimTreeLS: Simulating aerial and terrestrial laser scans of trees'. In: *arXiv:2011.11954 [cs, eess]* (Nov. 2020) (cited on pages 67–69).

[235] Niklas Peinecke, Thomas Lueken, and Bernd R. Korn. 'Lidar simulation using graphics hardware acceleration'. In: *2008 IEEE/AIAA 27th Digital Avionics Systems Conference*. St. Paul, MN, USA: IEEE, Oct. 2008, pp. 4.D.4–1–4.D.4–8. DOI: 10.1109/DASC.2008.4702838 (cited on pages 67, 70).

[236] Jin Fang et al. 'Augmented LiDAR Simulator for Autonomous Driving'. In: *IEEE Robotics and Automation Letters* 5.2 (Apr. 2020), pp. 1931–1938. DOI: 10.1109/LRA.2020.2969927 (cited on pages 67, 70, 186).

[237] Ying Li et al. 'Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review'. In: *IEEE Transactions on Neural Networks and Learning Systems* (2020), pp. 1–21. DOI: 10.1109/TNNLS.2020.3015992 (cited on page 67).

[238] Sivabalan Manivasagam et al. 'LiDARsim: Realistic LiDAR Simulation by Leveraging the Real World'. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2020, pp. 11164–11173. DOI: 10.1109/CVPR42600.2020.01118 (cited on pages 67, 69, 70, 186).

[239] Ting Yun et al. 'Simulation of multi-platform LiDAR for assessing total leaf area in tree crowns'. In: *Agricultural and Forest Meteorology* 276-277 (Oct. 2019), p. 107610. DOI: 10.1016/j.agrformet.2019.06.009 (cited on page 68).

[240] Peng Chen et al. 'OLE: A Novel Oceanic Lidar Emulator'. In: *IEEE Transactions on Geoscience and Remote Sensing* (2020), pp. 1–15. DOI: 10.1109/TGRS.2020.3035381 (cited on page 68).

[241] T. I. Zohdi. 'Rapid simulation-based uncertainty quantification of flash-type time-of-flight and Lidar-based body-scanning processes'. In: *Computer Methods in Applied Mechanics and Engineering* 359 (Feb. 2020), p. 112386. DOI: 10.1016/j.cma.2019.03.056 (cited on pages 68, 70, 171, 176, 192).

[242] LG Electronics R&D Lab. *LGSVL Simulator*. 2021. (Visited on 07/01/2021) (cited on pages 68, 186).

[243] Siemens. *Simcenter*. 2021. (Visited on 07/01/2021) (cited on page 68).

[244] Alexey Dosovitskiy et al. 'CARLA: An Open Urban Driving Simulator'. In: *Proceedings of the 1st Annual Conference on Robot Learning*. PMLR, Oct. 2017, pp. 1–16 (cited on pages 68, 69, 186).

[245] Xieyuanli Chen et al. 'Automatic Labeling to Generate Training Data for Online LiDAR-Based Moving Object Segmentation'. In: *IEEE Robotics and Automation Letters* 7.3 (July 2022), pp. 6107–6114. DOI: 10.1109/LRA.2022.3166544 (cited on page 68).

[246] J. Behley et al. 'Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: The SemanticKITTI Dataset'. In: *The International Journal on Robotics Research* 40.8-9 (2021), pp. 959–967. DOI: 10.1177/02783649211006735 (cited on pages 68, 184, 188, 275, 297).

[247] Weikai Tan et al. 'Toronto-3D: A large-scale mobile lidar dataset for semantic segmentation of urban roadways'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 202–203 (cited on page 68).

[248] Yukyung Choi et al. 'KAIST Multi-Spectral Day/Night Data Set for Autonomous and Assisted Driving'. In: *IEEE Transactions on Intelligent Transportation Systems* 19.3 (Mar. 2018). Conference Name: IEEE Transactions on Intelligent Transportation Systems, pp. 934–948. DOI: 10.1109/TITS.2018.2791533 (cited on page 68).

[249] U.S. Geological Survey. *Lidar Base Specification: Collection Requirements*. Techniques and Methods. U.S. Geological Survey, 2012 (cited on page 68).

[250] Instituto Geográfico de Información Geográfica. *PNOA LiDAR*. Publication Title: Plan Nacional de Ortografía Aérea. URL: https://pnoa.ign.es/ (visited on 11/18/2022) (cited on page 68).

[251] Nina Varney, Vijayan K. Asari, and Quinn Graehling. *DALES: A Large-scale Aerial LiDAR Data Set for Semantic Segmentation*. arXiv:2004.11985 [cs, stat]. Apr. 2020. DOI: 10.48550/arXiv.2004.11985. URL: http://arxiv.org/abs/2004.11985 (visited on 03/15/2023) (cited on page 68).

[252] Yancheng Pan et al. *SemanticPOSS: A Point Cloud Dataset with Large Quantity of Dynamic Instances*. 2020 (cited on pages 68, 184, 188).

[253] Bichen Wu et al. 'SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a LiDAR Point Cloud'. In: *2019 International Conference on Robotics and Automation (ICRA)*. May 2019, pp. 4376–4382. DOI: 10.1109/ICRA.2019.8793495 (cited on page 68).

[254] Holger Caesar et al. 'nuScenes: A multimodal dataset for autonomous driving'. In: *arXiv:1903.11027 [cs, stat]* (May 2020) (cited on pages 68, 184, 188).

[255] Timo Hackel et al. 'SEMANTIC3D: A new large-scale point cloud classification benchmark'. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. IV-1-W1. 2017, pp. 91–98 (cited on pages 68, 185, 188, 275, 297).

[256] Yurii Piadyk et al. 'StreetAware: A High-Resolution Synchronized Multimodal Urban Scene Dataset'. In: *Sensors* 23.7 (Jan. 2023). Number: 7 Publisher: Multidisciplinary Digital Publishing Institute, p. 3710. DOI: 10.3390/s23073710 (cited on page 68).

[257] Michael Gschwandtner et al. 'BlenSor: Blender Sensor Simulation Toolbox'. In: *Advances in Visual Computing*. Ed. by George Bebis et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 199–208. DOI: 10.1007/978-3-642-24031-7_20 (cited on pages 69, 70, 186, 192).

[258] Natalya Ahn et al. 'Real-time Simulation of Physical Multi-sensor Setups'. In: *ATZelectronics worldwide* 15.6 (June 2020), pp. 8–11. DOI: 10.1007/s38314-020-0207-1 (cited on pages 69, 192).

[259] Jian Zhao et al. 'Method and Applications of Lidar Modeling for Virtual Testing of Intelligent Vehicles'. In: *IEEE Transactions on Intelligent Transportation Systems* 22.5 (May 2021), pp. 2990–3000. DOI: 10.1109/TITS.2020.2978438 (cited on pages 69, 192).

[260] Arsalan Haider et al. 'Development of High-Fidelity Automotive LiDAR Sensor Model with Standardized Interfaces'. In: *Sensors* 22.19 (Jan. 2022), p. 7556. DOI: 10.3390/s22197556 (cited on pages 69, 70, 193).

[261] Timo Hanke et al. 'Generation and validation of virtual point cloud data for automated driving systems'. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. Oct. 2017, pp. 1–6. DOI: 10.1109/ITSC.2017.8317864 (cited on page 69).

[262] Andreas Ullrich and Martin Pfennigbauer. 'Advances in lidar point cloud processing'. In: *Laser Radar Technology and Applications XXIV*. Vol. 11005. SPIE, May 2019, pp. 157–166. DOI: 10.1117/12.2518856 (cited on page 69).

[263] Haopeng Chen and Steffen Müller. 'Analysis of Real-Time LiDAR Sensor Simulation for Testing Automated Driving Functions on a Vehicle-in-the-Loop Testbench'. In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. June 2022, pp. 1605–1614. DOI: 10.1109/IV51971.2022.9827048 (cited on pages 69, 70, 192, 193).

[264] Andrey Dimitrov and Mani Golparvar-Fard. 'Segmentation of building point cloud models including detailed architectural/structural features and MEP systems'. In: *Automation in Construction* 51 (Mar. 2015), pp. 32–45. DOI: 10.1016/j.autcon.2014.12.015 (cited on page 69).

[265] A. Isheil et al. 'Systematic error correction of a 3D laser scanning measurement device'. In: *Optics and Lasers in Engineering* 49.1 (Jan. 2011), pp. 16–24. DOI: 10.1016/j.optlaseng.2010.09.006 (cited on page 69).

[266] Lei Fan et al. 'Error in target-based georeferencing and registration in terrestrial laser scanning'. In: *Computers & Geosciences* 83 (Oct. 2015), pp. 54–64. DOI: 10.1016/j.cageo.2015.06.021 (cited on page 69).

[267] Jelena Pandžić et al. 'Error model of direct georeferencing procedure of terrestrial laser scanning'. In: *Automation in Construction* 78 (June 2017), pp. 13–23. DOI: 10.1016/j.autcon.2017.01.003 (cited on pages 69, 70).

[268] Aoran Xiao et al. 'SynLiDAR: Learning From Synthetic LiDAR Sequential Point Cloud for Semantic Segmentation'. In: *arXiv preprint arXiv:2107.05399* (2021) (cited on pages 69, 70, 184, 186, 188, 193).

[269] Benoit Guillard et al. 'Learning to Simulate Realistic LiDARs'. In: (Sept. 2022). DOI: 10.48550/arXiv.2209.10986 (cited on pages 69, 193).

[270] Patrik Vacek et al. 'Learning to Predict Lidar Intensities'. In: *IEEE Transactions on Intelligent Transportation Systems* 23.4 (Apr. 2022). Conference Name: IEEE Transactions on Intelligent Transportation Systems, pp. 3556–3564. DOI: 10.1109/TITS.2020.3037980 (cited on pages 70, 193).

[271] Hu Su et al. 'A Simulation Method for LIDAR of Autonomous Cars'. In: *IOP Conference Series: Earth and Environmental Science* 234 (Mar. 2019), p. 012055. DOI: 10.1088/1755-1315/234/1/012055 (cited on pages 70, 184, 186, 188).

[272] F. Banfi. 'The integration of a scan-to-HBIM process in BIM application: the development of an add-in to guide users in Autodesk Revit'. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2/W11 (May 2019), pp. 141–148. DOI: 10.5194/isprs-archives-XLII-2-W11-141-2019 (cited on pages 71, 206).

[273] Namhyuk Ham, Baek-Il Bae, and Ok-Kyung Yuh. 'Phased Reverse Engineering Framework for Sustainable Cultural Heritage Archives Using Laser Scanning and BIM: The Case of the Hwanggungwoo (Seoul, Korea)'. In: *Sustainability* 12.19 (Oct. 2020), p. 8108. DOI: 10.3390/su12198108 (cited on pages 71, 206).

[274] Florent Poux. 'The Smart Point Cloud: Structuring 3D intelligent point data'. PhD Thesis. Liège, Belgique: Université de Liège, Liège, Belgique, June 2019 (cited on page 71).

[275] A. Warchoł. 'THE CONCEPT OF LIDAR DATA QUALITY ASSESSMENT IN THE CONTEXT OF BIM MODELING'. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-1/W2 (Sept. 2019), pp. 61–66. DOI: 10.5194/isprs-archives-XLII-1-W2-61-2019 (cited on page 71).

[276] S. Soudarissanane and R. Lindenbergh. 'Optimizing Terrestrial Laser Scanning measurement set-up'. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XXXVIII-5-W12. Copernicus GmbH, Sept. 2012, pp. 127–132. DOI: 10.5194/isprsarchives-XXXVIII-5-W12-127-2011 (cited on pages 71, 73, 206, 220).

[277] Hélène Macher, Tania Landes, and Pierre Grussenmeyer. 'From Point Clouds to Building Information Models: 3D Semi-Automatic Reconstruction of Indoors of Existing Buildings'. In: *Applied Sciences* 7.10 (Oct. 2017), p. 1030. DOI: 10.3390/app7101030 (cited on pages 71, 205).

[278] Viorica Pătrăucean et al. 'State of research in automatic as-built modelling'. In: *Advanced Engineering Informatics* 29.2 (Apr. 2015), pp. 162–171. DOI: 10.1016/j.aei.2015.01.001 (cited on page 71).

[279] Rocha et al. 'A Scan-to-BIM Methodology Applied to Heritage Buildings'. In: *Heritage* 3.1 (Feb. 2020), pp. 47–67. DOI: 10.3390/heritage3010004 (cited on pages 71, 206).

[280] Juan Moyano et al. 'Bringing BIM to archaeological heritage: Interdisciplinary method/strategy and accuracy applied to a megalithic monument of the Copper Age'. In: *Journal of Cultural Heritage* 45 (Sept. 2020), pp. 303–314. DOI: 10.1016/j.culher.2020.03.010 (cited on pages 71, 206).

[281] Christoph Gollob, Tim Ritter, and Arne Nothdurft. 'Comparison of 3D Point Clouds Obtained by Terrestrial Laser Scanning and Personal Laser Scanning on Forest Inventory Sample Plots'. In: *Data* 5.4 (Dec. 2020), p. 103. DOI: 10.3390/data5040103 (cited on pages 71, 206).

[282] Liwen Li et al. 'Probability learning based tabu search for the budgeted maximum coverage problem'. In: *Expert Systems with Applications* 183 (Nov. 2021), p. 115310. DOI: 10.1016/j.eswa.2021.115310 (cited on pages 71, 73, 205).

[283] Houssem Eddine Mohamadi, Nadjia Kara, and Mohand Lagha. 'Efficient algorithms for decision making and coverage deployment of connected multi-low-altitude platforms'. In: *Expert Systems with Applications* 184 (Dec. 2021), p. 115529. DOI: 10.1016/j.eswa.2021.115529 (cited on pages 71, 73, 205).

[284] Vahid Roostapour et al. 'Pareto optimization for subset selection with dynamic cost constraints'. In: *Artificial Intelligence* 302 (Jan. 2022), p. 103597. DOI: 10.1016/j.artint.2021.103597 (cited on pages 71, 73, 205).

[285] Christian Potthast and Gaurav S. Sukhatme. 'A probabilistic framework for next best view estimation in a cluttered environment'. In: *Journal of Visual Communication and Image Representation*. Visual Understanding and Applications with RGB-D Cameras 25.1 (Jan. 2014), pp. 148–164. DOI: 10.1016/j.jvcir.2013.07.006 (cited on page 71).

[286] Mikhail Giorgini et al. 'Sensor-Based Optimization of Terrestrial Laser Scanning Measurement Setup on GPU'. In: *IEEE Geoscience and Remote Sensing Letters* 16.9 (Sept. 2019), pp. 1452–1456. DOI: 10.1109/LGRS.2019.2899681 (cited on pages 71–74, 207).

[287] Afrooz Aryan, Frédéric Bosché, and Pingbo Tang. 'Planning for terrestrial laser scanning in construction: A review'. In: *Automation in Construction* 125 (May 2021), p. 103551. DOI: 10.1016/j.autcon.2021.103551 (cited on pages 71, 72).

[288] Eisuke Wakisaka, Satoshi Kanai, and Hiroaki Date. 'Optimal Laser Scan Planning for As-Built Modeling of Plant Renovations Using Mathematical Programming'. In: *ISARC Proceedings* (May 2019), pp. 91–98 (cited on pages 72, 206, 207).

[289] Gilles Rougeron, Jérémie Le Garrec, and Claude Andriot. 'Optimal positioning of terrestrial LiDAR scanner stations in complex 3D environments with a multiobjective optimization method based on GPU simulations'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 193 (Nov. 2022), pp. 60–76. DOI: 10.1016/j.isprsjprs.2022.08.023 (cited on page 72).

[290] Zhiping Chen et al. '3D model-based terrestrial laser scanning (TLS) observation network planning for large-scale building facades'. In: *Automation in Construction* 144 (Dec. 2022), p. 104594. DOI: 10.1016/j.autcon.2022.104594 (cited on page 72).

[291] Michael J. Starek et al. 'Viewshed simulation and optimization for digital terrain modelling with terrestrial laser scanning'. In: *International Journal of Remote Sensing* 41.16 (Aug. 2020), pp. 6409–6426. DOI: 10.1080/01431161.2020.1752952 (cited on pages 72, 73, 207).

[292] Cheng Zhang, Vamsi Sai Kalasapudi, and Pingbo Tang. 'Rapid data quality oriented laser scan planning for dynamic construction environments'. In: *Advanced Engineering Informatics* 30.2 (Apr. 2016), pp. 218–232. DOI: 10.1016/j.aei.2016.03.004 (cited on page 72).

[293] Morteza Heidari Mozaffar and Masood Varshosaz. 'Optimal Placement of a Terrestrial Laser Scanner with an Emphasis on Reducing Occlusions'. In: *The Photogrammetric Record* 31.156 (2016), pp. 374–393. DOI: 10.1111/phor.12162 (cited on page 72).

[294] Hao Chen et al. 'Indoor high precision three-dimensional positioning system based on visible light communication using modified genetic algorithm'. In: *Optics Communications* 413 (Apr. 2018), pp. 103–120. DOI: 10.1016/j.optcom.2017.12.045 (cited on pages 72, 73, 207).

[295] F. Jia and D. Lichti. 'A comparison of simulated annealing, genetic algorithm and particle swarm optimization in optimal first-order design of indoor TLS networks'. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. IV-2-W4. Copernicus GmbH, Sept. 2017, pp. 75–82. DOI: 10.5194/isprs-annals-IV-2-W4-75-2017 (cited on pages 72, 73, 207).

[296] E. Latimer et al. 'Sensor space planning with applications to construction environments'. In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004.* Vol. 5. Apr. 2004, 4454–4460 Vol.5. DOI: 10.1109/ROBOT.2004.1302419 (cited on page 72).

[297] Jaehong Ahn and KwangYun Wohn. 'Interactive scan planning for heritage recording'. In: *Multimedia Tools and Applications* 75.7 (Apr. 2016), pp. 3655–3675. DOI: 10.1007/s11042-015-2473-0 (cited on page 72).

[298] Y.V. Pehlivanoglu and P. Pehlivanoglu. 'An enhanced genetic algorithm for path planning of autonomous UAV in target coverage problems'. In: *Applied Soft Computing* 112 (2021). DOI: 10.1016/j.asoc.2021.107796 (cited on page 73).

[299] Vincent Roberge and Mohammed Tarbouchi. 'Parallel Algorithm on GPU for Wireless Sensor Data Acquisition Using a Team of Unmanned Aerial Vehicles'. In: *Sensors* 21.20 (Jan. 2021), p. 6851. DOI: 10.3390/s21206851 (cited on pages 73, 74).

[300] Tae-Hyeong Kim and Tae-Hyoung Park. 'Placement Optimization of Multiple Lidar Sensors for Autonomous Vehicles'. In: *IEEE Transactions on Intelligent Transportation Systems* 21.5 (May 2020), pp. 2139–2145. DOI: 10.1109/TITS.2019.2915087 (cited on page 73).

[301] Marcelo Pereira et al. 'Self calibration of multiple LIDARs and cameras on autonomous vehicles'. In: *Robotics and Autonomous Systems* 83 (Sept. 2016), pp. 326–337. DOI: 10.1016/j.robot.2016.05.010 (cited on page 73).

[302] Lucas de Paula Veronese et al. 'An Accurate and Computational Efficient System for Detecting and Classifying Ego and Sides Lanes Using LiDAR'. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. June 2018, pp. 1476–1483. DOI: 10.1109/IVS.2018.8500434 (cited on page 73).

[303] Zi-Jia Wang, Zhi-Hui Zhan, and Jun Zhang. 'Solving the Energy Efficient Coverage Problem in Wireless Sensor Networks: A Distributed Genetic Algorithm Approach with Hierarchical Fitness Evaluation'. In: *Energies* 11.12 (Dec. 2018), p. 3526. DOI: 10.3390/en11123526 (cited on pages 73, 74).

[304] R. Islambouli and S. Sharafeddine. 'Optimized 3D Deployment of UAV-Mounted Cloudlets to Support Latency-Sensitive Services in IoT Networks'. In: *IEEE Access* 7 (2019), pp. 172860–172870. DOI: 10.1109/ACCESS.2019.2956150 (cited on page 73).

[305] Salvatore Filippo Di Gennaro and Alessandro Matese. 'Evaluation of novel precision viticulture tool for canopy biomass estimation and missing plant detection based on 2.5D and 3D approaches using RGB images acquired by UAV platform'. In: *Plant Methods* 16.1 (July 2020), p. 91. DOI: 10.1186/s13007-020-00632-2 (cited on page 74).

[306] L. G. Santesteban et al. 'High-resolution UAV-based thermal imaging to estimate the instantaneous and seasonal variability of plant water status within a vineyard'. In: *Agricultural Water Management*. Special Issue: Advances on ICTs for Water Management in Agriculture 183 (Mar. 2017), pp. 49–59. DOI: 10.1016/j.agwat.2016.08.026 (cited on page 74).

[307] Salvador Gutiérrez et al. 'Assessing and mapping vineyard water status using a ground mobile thermal imaging platform'. In: *Irrigation Science* 39.4 (July 2021), pp. 457–468. DOI: 10.1007/s00271-021-00735-1 (cited on page 74).

[308] Xuelian Peng et al. 'Prediction of the Nitrogen, Phosphorus and Potassium Contents in Grape Leaves at Different Growth Stages Based on UAV Multispectral Remote Sensing'. In: *Remote Sensing* 14.11 (Jan. 2022), p. 2659. DOI: 10.3390/rs14112659 (cited on page 74).

[309] R. G. V. Bramley. '12 - Precision Viticulture: managing vineyard variability for improved quality outcomes'. In: *Managing Wine Quality*. Ed. by Andrew G. Reynolds. Woodhead Publishing Series in Food Science, Technology and Nutrition. Woodhead Publishing, Jan. 2010, pp. 445–480. DOI: 10.1533/9781845699284.3.445 (cited on page 74).

[310] Javier Campos et al. 'Development of canopy vigour maps using UAV for site-specific management during vineyard spraying process'. In: *Precision Agriculture* 20.6 (Dec. 2019), pp. 1136–1156. DOI: 10.1007/s11119-019-09643-z (cited on page 74).

[311] Jorge Mendes et al. 'VineInspector: The Vineyard Assistant'. In: *Agriculture* 12.5 (May 2022), p. 730. DOI: 10.3390/agriculture12050730 (cited on pages 74, 78).

[312] I. Soubry, P. Patias, and V. Tsioukas. 'Monitoring vineyards with UAV and multi-sensors for the assessment of water stress and grape maturity'. In: *Journal of Unmanned Vehicle Systems* 5.2 (June 2017), pp. 37–50. DOI: 10.1139/juvs-2016-0024 (cited on page 74).

[313] Amirhossein Hassanzadeh et al. 'Broadacre Crop Yield Estimation Using Imaging Spectroscopy from Unmanned Aerial Systems (UAS): A Field-Based Case Study with Snap Bean'. In: *Remote Sensing* 13.16 (Jan. 2021), p. 3241. DOI: 10.3390/rs13163241 (cited on page 74).

[314] Abhaya Pal Singh et al. 'A Bibliometric Review of the Use of Unmanned Aerial Vehicles in Precision Agriculture and Precision Viticulture for Sensing Applications'. In: *Remote Sensing* 14.7 (Jan. 2022), p. 1604. DOI: 10.3390/rs14071604 (cited on pages 74, 131).

[315] Marco Ammoniaci et al. 'State of the Art of Monitoring Technologies and Data Processing for Precision Viticulture'. In: *Agriculture* 11.3 (Mar. 2021), p. 201. DOI: 10.3390/agriculture11030201 (cited on page 74).

[316] Alberto Sassu et al. 'Advances in Unmanned Aerial System Remote Sensing for Precision Viticulture'. In: *Sensors* 21.3 (Jan. 2021), p. 956. DOI: 10.3390/s21030956 (cited on page 74).

[317] Luís Pádua et al. 'Vineyard classification using OBIA on UAV-based RGB and multispectral data: A case study in different wine regions'. In: *Computers and Electronics in Agriculture* 196 (May 2022), p. 106905. DOI: 10.1016/j.compag.2022.106905 (cited on pages 74, 78, 131).

[318] Raymond F. Kokaly et al. *USGS Spectral Library Version 7*. USGS Numbered Series 1035. Reston, VA: U.S. Geological Survey, 2017, p. 68. DOI: 10.3133/ds1035 (cited on page 75).

[319] Debsunder Dutta et al. 'Characterizing Vegetation Canopy Structure Using Airborne Remote Sensing Data'. In: *IEEE Transactions on Geoscience and Remote Sensing* 55.2 (Feb. 2017), pp. 1160–1178. DOI: 10.1109/TGRS.2016.2620478 (cited on page 75).

[320] Freek van der Meer. 'The effectiveness of spectral similarity measures for the analysis of hyperspectral imagery'. In: *International Journal of Applied Earth Observation and Geoinformation* 8.1 (Jan. 2006), pp. 3–17. DOI: 10.1016/j.jag.2005.06.001 (cited on page 75).

[321] Zhongliang Ren, Qiuping Zhai, and Lin Sun. 'A Novel Method for Hyperspectral Mineral Mapping Based on Clustering-Matching and Nonnegative Matrix Factorization'. In: *Remote Sensing* 14.4 (Jan. 2022), p. 1042. DOI: 10.3390/rs14041042 (cited on page 75).

[322] Mirko Agarla et al. 'An analysis of spectral similarity measures'. In: *Color and Imaging Conference* 29 (Nov. 2021), pp. 300–305. DOI: 10.2352/issn.2169-2629.2021.29.300 (cited on page 75).

[323] Chandan Kumar et al. 'New effective spectral matching measures for hyperspectral data analysis'. In: *International Journal of Remote Sensing* 42.11 (June 2021), pp. 4126–4156. DOI: 10.1080/01431161.2021.1890265 (cited on page 75).

[324] Armando M. Fernandes et al. 'Grapevine variety identification using "Big Data" collected with miniaturized spectrometer combined with support vector machines and convolutional neural networks'. In: *Computers and Electronics in Agriculture* 163 (Aug. 2019), p. 104855. DOI: 10.1016/j.compag.2019.104855 (cited on page 75).

[325] Shara Ahmed et al. 'Applied aerial spectroscopy: A case study on remote sensing of an ancient and semi-natural woodland'. In: *PLOS ONE* 16.11 (Nov. 2021), e0260056. DOI: 10.1371/journal.pone.0260056 (cited on page 75).

[326] Luís Pádua et al. 'Monitoring of Chestnut Trees Using Machine Learning Techniques Applied to UAV-Based Multispectral Data'. In: *Remote Sensing* 12.18 (Jan. 2020), p. 3032. DOI: 10.3390/rs12183032 (cited on pages 75, 78).

[327] S. Shanmugam and P. Srinivasa Perumal. 'Spectral matching approaches in hyperspectral image processing'. In: *International Journal of Remote Sensing* 35.24 (Dec. 2014). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/01431161.2014.980922, pp. 8217–8251. DOI: 10.1080/01431161.2014.980922 (cited on page 75).

[328] Ricardo Augusto Borsoi et al. 'Spectral Variability in Hyperspectral Data Unmixing: A comprehensive review'. In: *IEEE Geoscience and Remote Sensing Magazine* 9.4 (Dec. 2021), pp. 223–270. DOI: 10.1109/MGRS.2021.3071158 (cited on page 76).

[329] Jignesh S. Bhatt and M. V. Joshi. 'Deep Learning in Hyperspectral Unmixing: A Review'. In: *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*. Sept. 2020, pp. 2189–2192. DOI: 10.1109/IGARSS39084.2020.9324546 (cited on page 76).

[330] Jonáš Hruška et al. 'Machine learning classification methods in hyperspectral data processing for agricultural applications'. In: *Proceedings of the International Conference on Geoinformatics and Data Analysis*. ICGDA '18. New York, NY, USA: Association for Computing Machinery, Apr. 2018, pp. 137–141. DOI: 10.1145/3220228.3220242 (cited on page 76).

[331] Chen Shi and Le Wang. 'Incorporating spatial information in spectral unmixing: A review'. In: *Remote Sensing of Environment* 149 (June 2014), pp. 70–87. DOI: 10.1016/j.rse.2014.03.034 (cited on page 76).

[332] Shaoquan Zhang et al. 'Spectral-Spatial Hyperspectral Unmixing Using Nonnegative Matrix Factorization'. In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–13. DOI: 10.1109/TGRS.2021.3074364 (cited on page 76).

[333] H. Jiang et al. 'Rapid and non-destructive detection of natural mildew degree of postharvest Camellia oleifera fruit based on hyperspectral imaging'. In: *Infrared Physics and Technology* 123 (2022). DOI: 10.1016/j.infrared.2022.104169 (cited on page 76).

[334] Q. Shenming, L. Xiang, and G. Zhihua. 'A new hyperspectral image classification method based on spatial-spectral features'. In: *Scientific Reports* 12.1 (2022). DOI: 10.1038/s41598-022-05422-5 (cited on pages 76, 77).

[335] Yuzhen Lu et al. 'Hyperspectral Imaging With Machine Learning to Differentiate Cultivars, Growth Stages, Flowers, and Leaves of Industrial Hemp (Cannabis sativa L.)' In: *Frontiers in Plant Science* 12 (2022) (cited on pages 76, 251, 254).

[336] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. arXiv:1802.03426 [cs, stat]. Sept. 2020. DOI: 10.48550/arXiv.1802.03426. URL: http://arxiv.org/abs/1802.03426 (visited on 04/09/2023) (cited on pages 77, 256).

[337] Hui Liu, Chenming Li, and Lizhong Xu. 'Dimension reduction and classification of hyperspectral images based on neural network sensitivity analysis and multi-instance learning'. In: *Computer Science and Information Systems* 16.2 (2019), pp. 443–468 (cited on page 77).

[338] Loganathan Agilandeeswari et al. 'Crop Classification for Agricultural Applications in Hyperspectral Remote Sensing Images'. In: *Applied Sciences* 12.3 (Jan. 2022), p. 1670. DOI: 10.3390/app12031670 (cited on page 77).

[339] Antonio Santos-Rufo et al. 'Wavelength Selection Method Based on Partial Least Square from Hyperspectral Unmanned Aerial Vehicle Orthomosaic of Irrigated Olive Orchards'. In: *Remote Sensing* 12.20 (Jan. 2020), p. 3426. DOI: 10.3390/rs12203426 (cited on page 77).

[340] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. 'Regularization Paths for Generalized Linear Models via Coordinate Descent'. In: *Journal of Statistical Software* 33.1 (2010), pp. 1–22 (cited on page 77).

[341] Tahir Mehmood et al. 'A review of variable selection methods in Partial Least Squares Regression'. In: *Chemometrics and Intelligent Laboratory Systems* 118 (Aug. 2012), pp. 62–69. DOI: 10.1016/j.chemolab.2012.07.010 (cited on page 77).

[342] Salvador Gutiérrez et al. 'On-The-Go Hyperspectral Imaging Under Field Conditions and Machine Learning for the Classification of Grapevine Varieties'. In: *Frontiers in Plant Science* 9 (July 2018), p. 1102. DOI: 10.3389/fpls.2018.01102 (cited on pages 77, 239, 240).

[343] F. Fuentes-Peñailillo et al. 'Using clustering algorithms to segment UAV-based RGB images'. In: *2018 IEEE International Conference on Automation/XXIII Congress of the Chilean Association of Automatic Control (ICA-ACCA)*. Oct. 2018, pp. 1–5. DOI: 10.1109/ICA-ACCA.2018.8609822 (cited on page 78).

[344] Georgios D. Karatzinis et al. 'Towards an Integrated Low-Cost Agricultural Monitoring System with Unmanned Aircraft System'. In: *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. Sept. 2020, pp. 1131–1138. DOI: 10.1109/ICUAS48674.2020.9213900 (cited on page 78).

[345] Chantal Hajjar et al. 'Vine Identification and Characterization in Goblet-Trained Vineyards Using Remotely Sensed Images'. In: *Remote Sensing* 13.15 (Jan. 2021), p. 2992. DOI: 10.3390/rs13152992 (cited on page 78).

[346] Carlos Poblete-Echeverría et al. 'Detection and Segmentation of Vine Canopy in Ultra-High Spatial Resolution RGB Imagery Obtained from Unmanned Aerial Vehicle (UAV): A Case Study in a Commercial Vineyard'. In: *Remote Sensing* 9.3 (Mar. 2017), p. 268. DOI: 10.3390/rs9030268 (cited on page 78).

[347] André Silva Aguiar et al. 'Localization and Mapping on Agriculture Based on Point-Feature Extraction and Semiplanes Segmentation From 3D LiDAR Data'. In: *Frontiers in Robotics and AI* 9 (2022) (cited on page 78).

[348] Juan M. Jurado et al. 'Automatic Grapevine Trunk Detection on UAV-Based Point Cloud'. In: *Remote Sensing* 12.18 (Jan. 2020), p. 3043. DOI: 10.3390/rs12183043 (cited on page 78).

[349] Mohamed Kerkech, Adel Hafiane, and Raphael Canals. 'Vine disease detection in UAV multispectral images using optimized image registration and deep learning segmentation approach'. In: *Computers and Electronics in Agriculture* 174 (July 2020), p. 105446. DOI: 10.1016/j.compag.2020.105446 (cited on page 78).

[350] A. P. Nolan et al. 'Automated detection and segmentation of vine rows using high resolution UAS imagery in a commercial vineyard'. In: *Proceedings - 21st International Congress on Modelling and Simulation, MODSIM 2015*. Modelling, Simulation Society of Australia, and New Zealand Inc. (MSSANZ), 2015, pp. 1406–1412 (cited on page 78).

[351] Yanan Li et al. 'Performance Evaluation of Crop Segmentation Algorithms'. In: *IEEE Access* 8 (2020), pp. 36210–36225. DOI: 10.1109/ACCESS.2020.2969451 (cited on page 78).

[352] Anna Kicherer et al. 'Phenoliner: A New Field Phenotyping Platform for Grapevine Research'. In: *Sensors* 17.7 (July 2017), p. 1625. DOI: 10.3390/s17071625 (cited on page 78).

[353] Canh Nguyen et al. 'Early Detection of Plant Viral Disease Using Hyperspectral Imaging and Deep Learning'. In: *Sensors (Basel, Switzerland)* 21.3 (Jan. 2021), p. 742. DOI: 10.3390/s21030742 (cited on page 78).

[354] Nele Bendel et al. 'Detection of Grapevine Leafroll-Associated Virus 1 and 3 in White and Red Grapevine Cultivars Using Hyperspectral Imaging'. In: *Remote Sensing* 12.10 (Jan. 2020), p. 1693. DOI: 10.3390/rs12101693 (cited on page 78).

[355] N. Bendel et al. 'Evaluating the suitability of hyper- and multispectral imaging to detect foliar symptoms of the grapevine trunk disease Esca in vineyards'. In: *Plant Methods* 16.1 (2020). DOI: 10.1186/s13007-020-00685-3 (cited on page 78).

[356] Ana Cláudia Teixeira et al. 'A Systematic Review on Automatic Insect Detection Using Deep Learning'. In: *Agriculture* 13.3 (Mar. 2023). Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, p. 713. DOI: 10.3390/agriculture13030713 (cited on page 78).

[357] Xiang Zhang, Huiyi Gao, and Li Wan. 'Classification of Fine-Grained Crop Disease by Dilated Convolution and Improved Channel Attention Module'. In: *Agriculture* 12.10 (Oct. 2022), p. 1727. DOI: 10.3390/agriculture12101727 (cited on page 78).

[358] Keng-Hao Liu et al. 'Plant Species Classification Based on Hyperspectral Imaging via a Lightweight Convolutional Neural Network Model'. In: *Frontiers in Plant Science* 13 (2022) (cited on pages 78, 243, 250, 251, 262).

[359] Somayeh Nezami et al. 'Tree Species Classification of Drone Hyperspectral and RGB Imagery with Deep Learning Convolutional Neural Networks'. In: *Remote Sensing* 12.7 (Jan. 2020). Number: 7 Publisher: Multidisciplinary Digital Publishing Institute, p. 1070. DOI: 10.3390/rs12071070 (cited on pages 79, 239, 250, 263).

[360] M. Graña, M.A. Veganzons, and B. Ayerdi. *Hyperspectral Remote Sensing Scenes*. Type: Hyperspectral Remote Sensing Scenes. URL: https://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes (visited on 11/02/2022) (cited on pages 79, 240).

[361] Jaime Moraga and H. Sebnem Duzgun. *JigsawHSI: a network for Hyperspectral Image classification*. June 2022. DOI: 10.48550/arXiv.2206.02327. URL: http://arxiv.org/abs/2206.02327 (visited on 09/25/2022) (cited on pages 79, 240, 250, 251, 258, 264).

[362] Tanmay Chakraborty and Utkarsh Trehan. *SpectralNET: Exploring Spatial-Spectral WaveletCNN for Hyperspectral Image Classification*. Apr. 2021. DOI: 10.48550/arXiv.2104.00341. URL: http://arxiv.org/abs/2104.00341 (visited on 09/25/2022) (cited on pages 79, 240, 243, 250, 251, 265).

[363] Swalpa Kumar Roy et al. 'HybridSN: Exploring 3D-2D CNN Feature Hierarchy for Hyperspectral Image Classification'. In: *IEEE Geoscience and Remote Sensing Letters* 17.2 (Feb. 2020), pp. 277–281. DOI: 10.1109/LGRS.2019.2918719 (cited on pages 79, 250, 251, 267).

[364] Swalpa Kumar Roy et al. 'Attention-Based Adaptive Spectral–Spatial Kernel ResNet for Hyperspectral Image Classification'. In: *IEEE Transactions on Geoscience and Remote Sensing* 59.9 (Sept. 2021), pp. 7831–7843. DOI: 10.1109/TGRS.2020.3043267 (cited on pages 79, 240, 243, 246).

[365] Zhaohui Xue et al. 'Attention-Based Second-Order Pooling Network for Hyperspectral Image Classification'. In: *IEEE Transactions on Geoscience and Remote Sensing* 59.11 (Nov. 2021), pp. 9600–9615. DOI: 10.1109/TGRS.2020.3048128 (cited on pages 79, 240, 246, 248, 250, 251, 261).

[366] Qiqi Zhu et al. 'A Spectral-Spatial-Dependent Global Learning Framework for Insufficient and Imbalanced Hyperspectral Image Classification'. In: *IEEE Transactions on Cybernetics* (2021), pp. 1–15. DOI: 10.1109/TCYB.2021.3070577 (cited on page 79).

[367] Ziqi Xin et al. 'Convolution Enhanced Spatial-Spectral Unified Transformer Network for Hyperspectral Image Classification'. In: *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium*. July 2022, pp. 2267–2270. DOI: 10.1109/IGARSS46834.2022.9884124 (cited on page 79).

[368] Peiyan Guan and Edmund Y. Lam. 'Spatial-Spectral Contrastive Learning for Hyperspectral Image Classification'. In: *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium*. July 2022, pp. 1372–1375. DOI: 10.1109/IGARSS46834.2022.9883226 (cited on pages 79, 275, 298).

[369] Susan Meerdink et al. 'Multitarget Multiple-Instance Learning for Hyperspectral Target Detection'. In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–14. DOI: 10.1109/TGRS.2021.3060966 (cited on pages 79, 275, 298).

[370] S. Campana. 'Drones in Archaeology. State-of-the-art and Future Perspectives'. In: *Archaeological Prospection* 24.4 (2017). _eprint: https://bit.ly/3bXrH5t, pp. 275–296. DOI: 10.1002/arp.1569 (cited on page 79).

[371] J Waagen. 'New technology and archaeological practice. Improving the primary archaeological recording process in excavation by means of UAS photogrammetry'. In: *Journal of Archaeological Science* 101 (2019), pp. 11–20. DOI: http://doi.org/gfpfw9 (cited on page 79).

[372] Antonio Pecci and Nicola Masini. 'Archaeology, historical site risk assessment and monitoring by UAV: approaches and case studies'. In: *EGU General Assembly Conference Abstracts*. EGU General Assembly Conference Abstracts. Apr. 2016, EPSC2016–17424 (cited on page 80).

[373] Marco Dubbini, Lucia Irene Curzio, and Alessandro Campedelli. 'Digital elevation models from unmanned aerial vehicle surveys for archaeological interpretation of terrain anomalies: case study of the Roman castrum of Burnum (Croatia)'. In: *Journal of Archaeological Science: Reports* 8 (2016), pp. 121–134. DOI: 10.1016/j.jasrep.2016.05.054 (cited on page 80).

[374] Jesse Casana et al. 'Archaeological Aerial Thermography in Theory and Practice'. In: *Advances in Archaeological Practice* 5.4 (Nov. 2017), pp. 310–327. DOI: 10.1017/aap.2017.23 (cited on pages 80, 229).

[375] Christopher Brooke. 'Thermal Imaging for the Archaeological Investigation of Historic Buildings'. In: *Remote Sensing* 10.9 (Sept. 2018), p. 1401. DOI: 10.3390/rs10091401 (cited on page 80).

[376] Madeleine McLeester et al. 'Detecting prehistoric landscape features using thermal, multispectral, and historical imagery analysis at Midewin National Tallgrass Prairie, Illinois'. In: *Journal of Archaeological Science: Reports* 21 (Oct. 2018), pp. 450–459. DOI: 10.1016/j.jasrep.2018.08.016 (cited on page 80).

[377] José Angel Salgado Carmona et al. 'Assessing the potential of multispectral and thermal UAV imagery from archaeological sites. A case study from the Iron Age hillfort of Villasviejas del Tamuja (Cáceres, Spain)'. In: *Journal of Archaeological Science: Reports* 31 (June 2020), p. 102312. DOI: 10.1016/j.jasrep.2020.102312 (cited on pages 80, 229).

[378] Giacomo Patrucco et al. '3D Data Fusion for Historical Analyses of Heritage Buildings Using Thermal Images: The Palacio de Colomina as a Case Study'. In: *Remote Sensing* 14.22 (2022). DOI: 10.3390/rs14225699 (cited on page 80).

[379] G. Teza and A. Pesci. 'Evaluation of the temperature pattern of a complex body from thermal imaging and 3D information: A method and its MATLAB implementation'. In: *Infrared Physics and Technology* 96 (2019), pp. 228–237. DOI: 10.1016/j.infrared.2018.11.029 (cited on page 91).

[380] J. Park et al. 'pi-SIFT: A photometric and Scale Invariant Feature Transform'. In: *ICPR 2008 19th International Conference on Pattern Recognition*. ISSN: 1051-4651. IEEE Computer Society, Dec. 2008. DOI: 10.1109/ICPR.2008.4761181 (cited on page 96).

[381] G. D. Evangelidis and E. Z. Psarakis. 'Parametric Image Alignment Using Enhanced Correlation Coefficient Maximization'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.10 (2008), pp. 1858–1865. DOI: 10.1109/TPAMI.2008.113 (cited on page 96).

[382] Jack Sklansky. 'Finding the convex hull of a simple polygon'. In: *Pattern Recognition Letters* 1.2 (1982), pp. 79–83. DOI: https://doi.org/10.1016/0167-8655(82)90016-2 (cited on page 100).

[383] Haekyung Park and Dongkun Lee. 'Comparison between point cloud and mesh models using images from an unmanned aerial vehicle'. In: *Measurement* 138 (May 2019), pp. 461–466. DOI: 10.1016/j.measurement.2019.02.023 (cited on page 110).

[384] Krishna Ribeiro-Gomes et al. 'Uncooled Thermal Camera Calibration and Optimization of the Photogrammetry Process for UAV Applications in Agriculture'. In: *Sensors* 17.10 (Oct. 2017), p. 2173. DOI: 10.3390/s17102173 (cited on page 110).

[385] J. Behley, V. Steinhage, and A. B. Cremers. 'Efficient radius neighbor search in three-dimensional point clouds'. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 3625–3630. DOI: 10.1109/ICRA.2015.7139702 (cited on page 113).

[386] David Cohen-Steiner and Frank Da. 'A Greedy Delaunay-Based Surface Reconstruction Algorithm'. In: *Vis. Comput.* 20.1 (Apr. 2004). Place: Berlin, Heidelberg Publisher: Springer-Verlag, pp. 4–16. DOI: 10.1007/s00371-003-0217-z (cited on page 115).

[387] Julia Sanchez et al. 'Robust normal vector estimation in 3D point clouds through iterative principal component analysis'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 163 (May 2020), pp. 18–35. DOI: 10.1016/j.isprsjprs.2020.02.018 (cited on page 119).

[388] Patricio Martínez-Carricondo et al. 'Assessment of UAV-photogrammetric mapping accuracy based on variation of ground control points'. In: *International Journal of Applied Earth Observation and Geoinformation* 72 (Oct. 2018). Publisher: Elsevier BV, pp. 1–10. DOI: 10.1016/j.jag.2018.05.015 (cited on page 122).

[389] Jonathan Richard Shewchuk. 'Delaunay refinement algorithms for triangular mesh generation'. In: *Computational Geometry* 22.1 (2002), pp. 21–74. DOI: https://doi.org/10.1016/S0925-7721(01)00047-5 (cited on page 122).

[390] Sung-Hyuk Cha. 'Comprehensive Survey on Distance/Similarity Measures Between Probability Density Functions'. In: *Int. J. Math. Model. Meth. Appl. Sci.* 1 (Jan. 2007) (cited on page 126).

[391] Sen Jia et al. 'A survey: Deep learning for hyperspectral image classification with few labeled samples'. In: *Neurocomputing* 448 (Aug. 2021), pp. 179–204. DOI: 10.1016/j.neucom.2021.03.035 (cited on page 131).

[392] Xing Hu et al. 'Hyperspectral Anomaly Detection Using Deep Learning: A Review'. In: *Remote Sensing* 14.9 (Jan. 2022), p. 1973. DOI: 10.3390/rs14091973 (cited on page 131).

[393] Mohd Nasir Mohamad, Mohd Nadzri Md Reba, and Mohammad Shawkat Hossain. 'A screening approach for the correction of distortion in UAV data for coral community mapping'. In: *Geocarto International* 0.0 (July 2021), pp. 1–33. DOI: 10.1080/10106049.2021.1958066 (cited on page 131).

[394] Bernhard Kerbl et al. 'Effective static bin patterns for sort-middle rendering'. In: *Proceedings of High Performance Graphics*. HPG '17. New York, NY, USA: Association for Computing Machinery, July 2017, pp. 1–10. DOI: 10.1145/3105762.3105777 (cited on page 146).

[395] Mohammad Norouzi, David J Fleet, and Russ R Salakhutdinov. 'Hamming Distance Metric Learning'. In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012 (cited on page 153).

[396] Yubal Barrios et al. 'SHyLoC 2.0: A Versatile Hardware Solution for On-Board Data and Hyperspectral Image Compression on Future Space Missions'. In: *IEEE Access* 8 (2020), pp. 54269–54287. DOI: 10.1109/ACCESS.2020.2980767 (cited on page 156).

[397] Yubal Barrios et al. 'Performance Assessment of the CCSDS-123 Standard for Panchromatic Video Compression on Space Missions'. In: *IEEE Geoscience and Remote Sensing Letters* 19 (2022), pp. 1–5. DOI: 10.1109/LGRS.2021.3099032 (cited on page 156).

[398] Guantao Xuan et al. 'Early diagnosis and pathogenesis monitoring of wheat powdery mildew caused by blumeria graminis using hyperspectral imaging'. In: *Computers and Electronics in Agriculture* 197 (June 2022), p. 106921. DOI: 10.1016/j.compag.2022.106921 (cited on page 156).

[399] Oscar Ferraz, Vitor Silva, and Gabriel Falcao. 'Hyperspectral Parallel Image Compression on Edge GPUs'. In: *Remote Sensing* 13.6 (Jan. 2021), p. 1077. DOI: 10.3390/rs13061077 (cited on page 156).

[400] *Hyperspectral Remote Sensing Scenes - Grupo de Inteligencia Computacional (GIC)*. Publication Title: Hyperspectral Remote Sensing Scenes. URL: http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes#Pavia_Centre_and_University (visited on 05/11/2022) (cited on page 156).

[401] Guillaume Cordonnier et al. 'Authoring landscapes by combining ecosystem and terrain erosion simulation'. In: *ACM Transactions on Graphics* 36.4 (July 2017), pp. 1–12. DOI: 10.1145/3072959.3073667 (cited on page 167).

[402] Roland Fischer et al. 'AutoBiomes: procedural generation of multi-biome landscapes'. In: *The Visual Computer* 36.10-12 (July 2020), pp. 2263–2272. DOI: 10.1007/s00371-020-01920-7 (cited on page 167).

[403] Mi\textbackslashlosz Makowski et al. 'Synthetic silviculture'. In: *ACM Transactions on Graphics* 38.4 (July 2019), pp. 1–14. DOI: 10.1145/3306346.3323039 (cited on page 167).

[404] Alexander Majercik et al. 'A Ray-Box Intersection Algorithm and Efficient Dynamic Voxel Rendering'. In: *Journal of Computer Graphics Techniques (JCGT)* 7.3 (Sept. 2018), pp. 66–81 (cited on page 170).

[405] Tomas Möller and Ben Trumbore. 'Fast, Minimum Storage Ray-Triangle Intersection'. In: *Journal of Graphics Tools* 2.1 (Jan. 1997), pp. 21–28. DOI: 10.1080/10867651.1997.10487468 (cited on page 170).

[406] Paul F. McManamon. *LiDAR Technologies and Systems*. SPIE, July 2019 (cited on page 172).

[407] Hesai. *PandaSet*. 2021. (Visited on 12/14/2021) (cited on pages 173, 186, 187).

[408] David H Douglas and Thomas K Peucker. 'Algorithms for the reduction of the number of points required to represent a digitized line or its caricature'. In: *Cartographica: The International Journal for Geographic Information and Geovisualization* 10.2 (Dec. 1973), pp. 112–122. DOI: 10.3138/FM57-6770-U75U-7727 (cited on page 176).

[409] Michael E. Hodgson and Patrick Bresnahan. 'Accuracy of Airborne Lidar-Derived Elevation'. In: *Photogrammetric Engineering & Remote Sensing* 70.3 (Mar. 2004), pp. 331–339. DOI: 10.14358/pers.70.3.331 (cited on page 179).

[410] Emmanuel P. Baltsavias. 'A comparison between photogrammetry and laser scanning'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 54.2-3 (July 1999), pp. 83–94. DOI: 10.1016/s0924-2716(99)00014-3 (cited on page 179).

[411] W Boehler and M. Bordas Vicent Marbs A. *Investigating LASER Scanner Accuracy*. Tech. rep. i3mainz, Institute for Spatial Information and Surveying Technology, FH Mainz, Holzstrasse 36, 55116 Mainz, Germany, 2018 (cited on page 181).

[412] Xiangyu Yue et al. 'A lidar point cloud generator: from a virtual world to autonomous driving'. In: *ICMR*. ACM, 2018, pp. 458–464 (cited on pages 184, 186, 188).

[413] DJI. *Zenmuse L1 Specs*. 2020. (Visited on 12/06/2021) (cited on pages 184, 188).

[414] Shital Shah et al. 'AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles'. In: *arXiv:1705.05065 [cs]* (July 2017) (cited on page 186).

[415] Julián Tachella et al. 'Real-time 3D reconstruction from single-photon lidar data using plug-and-play point cloud denoisers'. In: *Nature Communications* 10.1 (Nov. 2019), p. 4984. DOI: 10.1038/s41467-019-12943-7 (cited on pages 189, 201).

[416] Alexander Buslaev et al. 'Albumentations: Fast and Flexible Image Augmentations'. In: *Information* 11.2 (Feb. 2020). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 125. DOI: 10.3390/info11020125 (cited on page 193).

[417] Mikhail N. Polyanskiy. *Refractive index database*. Publication Title: Refractive index database. URL: https://refractiveindex.info (visited on 12/15/2022) (cited on page 193).

[418] Pablo Rodríguez-Gonzálvez et al. 'Mobile LiDAR System: New Possibilities for the Documentation and Dissemination of Large Cultural Heritage Sites'. In: *Remote Sensing* 9.3 (Mar. 2017), p. 189. DOI: 10.3390/rs9030189 (cited on page 206).

[419] Anne Bienert et al. 'Comparison and Combination of Mobile and Terrestrial Laser Scanning for Natural Forest Inventories'. In: *Forests* 9.7 (July 2018), p. 395. DOI: 10.3390/f9070395 (cited on page 206).

[420] Ricardo Marques, Christian Bouville, and Kadi Bouatouch. 'Optimal Sample Weights for Hemispherical Integral Quadratures'. In: *Computer Graphics Forum* 38.1 (2019), pp. 59–72. DOI: 10.1111/cgf.13392 (cited on page 212).

[421] John B Burkardt. *The Halton Quasi Monte Carlo (QMC) Sequence*. Dec. 2010. URL: https://people.math.sc.edu/Burkardt/f_src/halton_advanced/halton_advanced.html (visited on 01/09/2022) (cited on page 212).

[422] Jakub Więckowski, Bartłomiej Kizielewicz, and Joanna Kołodziejczyk. 'Finding an Approximate Global Optimum of Characteristic Objects Preferences by Using Simulated Annealing'. In: *Intelligent Decision Technologies*. Ed. by Ireneusz Czarnowski, Robert J. Howlett, and Lakhmi C. Jain. Smart Innovation, Systems and Technologies. Singapore: Springer, 2020, pp. 365–375. DOI: 10.1007/978-981-15-5925-9_31 (cited on page 214).

[423] Marco Vannucci, Valentina Colla, and Silvia Cateni. 'Genetic Operators Impact on Genetic Algorithms Based Variable Selection'. In: *Intelligent Decision Technologies*. Ed. by Ireneusz Czarnowski, Robert J. Howlett, and Lakhmi C. Jain. Smart Innovation, Systems and Technologies. Singapore: Springer, 2020, pp. 211–221. DOI: 10.1007/978-981-15-5925-9_18 (cited on page 215).

[424] Enrique Fernández Hervás. 'El castillo de Arenas, fortaleza nazarita del Reino de Granada'. In: *Castillos de España: publicación de la Asociación Española de Amigos de los Castillos* 91 (1986). Publisher: Asociación Española de Amigos de los Castillos Section: Castillos de España: publicación de la Asociación Española de Amigos de los Castillos, pp. 41–44 (cited on pages 227, 228, 234).

[425] Francisco Olivares. *Castillos de la provincia de Jaén*. Instituto de Estudios Giennenses, 1992 (cited on page 228).

[426] Roque Modrego-Fernández and José-María Martín-Civantos. 'Propuesta metodológica para análisis arqueológicos de altitud: aplicación al Castillo de Arenas (Campillo de Arenas, Jaén)'. In: *Cuadernos de Arqueología de la Universidad de Navarra* (Dec. 2022), pp. 273–300. DOI: 10.15581/012.30.2.012 (cited on page 236).

[427] Deborah L. Bandalos. *Measurement Theory and Applications for the Social Sciences*. Guilford Publications, Jan. 2018 (cited on page 242).

[428] Pascal Bertolino. 'Sensarea: An authoring tool to create accurate clickable videos'. In: *2012 10th International Workshop on Content-Based Multimedia Indexing (CBMI)*. June 2012, pp. 1–4. DOI: 10.1109/CBMI.2012.6269804 (cited on page 243).

[429] Francois Chollet. *Deep Learning with Python, Second Edition*. Google-Books-ID: mjVKEAAAQBAJ. Simon and Schuster, Dec. 2021 (cited on page 245).

[430] Teja Kattenborn et al. 'Review on Convolutional Neural Networks (CNN) in vegetation remote sensing'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 173 (Mar. 2021), pp. 24–49. DOI: 10.1016/j.isprsjprs.2020.12.010 (cited on page 245).

[431] Erting Pan et al. 'Spectral-spatial classification for hyperspectral image based on a single GRU'. In: *Neurocomputing* 387 (Apr. 2020), pp. 150–160. DOI: 10.1016/j.neucom.2020.01.029 (cited on page 247).

[432] Guandong Li and Chunju Zhang. *Faster hyperspectral image classification based on selective kernel mechanism using deep convolutional networks*. Feb. 2022. DOI: 10.48550/arXiv.2202.06458. URL: http://arxiv.org/abs/2202.06458 (visited on 09/25/2022) (cited on page 248).

[433] Christian Szegedy et al. *Going Deeper with Convolutions*. arXiv:1409.4842 [cs] version: 1. Sept. 2014. DOI: 10.48550/arXiv.1409.4842. URL: http://arxiv.org/abs/1409.4842 (visited on 04/09/2023) (cited on pages 248, 253).

[434] Anirudh Ravikumar et al. 'Hyperspectral Image Classification Using Deep Matrix Capsules'. In: *2022 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI)*. Vol. 01. Dec. 2022, pp. 1–7. DOI: 10.1109/ICDSAAI55433.2022.10028853 (cited on page 258).

[435] Vincent Roberge, Mohammed Tarbouchi, and Gilles Labonté. 'Fast Genetic Algorithm Path Planner for Fixed-Wing Military UAV Using GPU'. In: *IEEE Transactions on Aerospace and Electronic Systems* 54.5 (Oct. 2018), pp. 2105–2117. DOI: 10.1109/TAES.2018.2807558 (cited on pages 275, 297).

[436] Facundo Pessacg et al. 'Simplifying UAV-Based Photogrammetry in Forestry: How to Generate Accurate Digital Terrain Model and Assess Flight Mission Settings'. In: *Forests* 13.2 (Feb. 2022). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 173. DOI: 10.3390/f13020173 (cited on pages 275, 297).

[437] Lei Li et al. 'Multi-branch Semantic GAN for Infrared Image Generation from Optical Image'. In: *Intelligence Science and Big Data Engineering. Visual Data Engineering*. Ed. by Zhen Cui et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 484–494. DOI: 10.1007/978-3-030-36189-1_40 (cited on pages 275, 297).

[438] Bing Li et al. 'I-GANs for Infrared Image Generation'. In: *Complexity* 2021 (Mar. 2021). Publisher: Hindawi, e6635242. DOI: 10.1155/2021/6635242 (cited on pages 275, 297).

[439] Vladimir V. Kniaz et al. 'ThermalGAN: Multimodal Color-to-Thermal Image Translation for Person Re-identification in Multispectral Dataset'. In: *Computer Vision – ECCV 2018 Workshops*. Ed. by Laura Leal-Taixé and Stefan Roth. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 606–624. DOI: 10.1007/978-3-030-11024-6_46 (cited on pages 275, 297).

This dissertation was typeset using the kaobook template: https://github.com/fmarotta/kaobook/. The changes made to the default template can be found in the following repository: https://github.com/AlfonsoLRz/PhD-Dissertation.