Future Generation Computer Systems 134 (2022) 66–77

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An out-of-core method for GPU image mapping on large 3D scenarios of the real world

Juan M. Jurado [a,*], Emilio J. Padrón [b], J. Roberto Jiménez [a], Lidia Ortega [a]

[a] Computer Graphics and Geomatics Group of Jaén, University of Jaén, Spain
[b] CITIC Research & Computer Architecture Group, University of A Coruña, Spain

ARTICLE INFO

Article history:
Received 14 July 2021
Received in revised form 3 February 2022
Accepted 15 March 2022
Available online 24 March 2022

Keywords:
Parallel computing
GPGPU
Image mapping
3D model
Multi-source data fusion

ABSTRACT

Image mapping on 3D huge scenarios of the real world is one of the most fundamental and computational expensive processes for the integration of multi-source sensing data. Recent studies focused on the observation and characterization of Earth have been enhanced by the proliferation of Unmanned Aerial Vehicle (UAV) and sensors able to capture massive datasets with a high spatial resolution. Despite the advances in manufacturing new cameras and versatile platforms, only a few methods have been developed to characterize the study area by fusing heterogeneous data such as thermal, multispectral or hyperspectral images with high-resolution 3D models. The main reason for this lack of solutions is the challenge to integrate multi-scale datasets and high computational efforts required for image mapping on dense and complex geometric models. In this paper, we propose an efficient pipeline for multi-source image mapping on huge 3D scenarios. Our GPU-based solution significantly reduces the run time and allows us to generate enriched 3D models on-site. The proposed method is out-of-core and it uses available resources of the GPU's machine to perform two main tasks: (i) image mapping and (ii) occlusion testing. We deploy highly-optimized GPU-kernels for image mapping and detection of self-hidden geometry in the 3D model, as well as a GPU-based parallelization to manage the 3D model considering several spatial partitions according to the GPU capabilities. Our method has been tested on 3D scenarios with different point cloud densities (66M, 271M, 542M) and two sets of multispectral images collected by two drone flights. We focus on launching the proposed method on three platforms: (i) System on a Chip (SoC), (ii) a user-grade laptop and (iii) a PC. The results demonstrate the method's capabilities in terms of performance and versatility to be computed by commodity hardware. Thus, taking advantage of GPUs, this method opens the door for embedded and edge computing devices for 3D image mapping on large-scale scenarios in near real-time.

1. Introduction

Nowadays, precision agriculture or environmental health diagnostics make widespread use of multi-sensors coupled with drones or UAVs (Unmanned Aerial Vehicles). Some of these devices are thermal sensors, RGB, multispectral or hyperspectral cameras, as well as LiDAR (Light Detection and Ranging or Laser Imaging Detection and Ranging) systems. At present, they all have lightened their weight, improved their performances and lowered their cost. This allows us to monitor large areas of crops or forests remotely, obtaining information in the visible and non-visible spectral ranges. Large areas can be monitored on each flight, depending on the flight altitude and battery life. In any case, a day's flying usually generates large amounts of information that needs high computational requirements to be processed.

Applications of these technologies are very diverse. Thermal sensing, for instance, is useful for detecting the impact of heat waves and drought in crops or ecosystems [1]. However, not always only one sensor is attached to the drone. There is a tendency to use several combined sensors in the so-called UAS (Unmanned Aerial Systems) to obtain diversified information [2]. As a consequence, huge amounts of heterogeneous data must be managed.

An additional objective is to process all this heterogeneous information under the same data model, including the 3D models. In fact, RGB and LiDAR sensors allow us to generate 3D point clouds, which characterize the geometric properties of soil and vegetation. Therefore, an ideal capture and processing mechanism would be able to automatically integrate both geometric and spectral information in the same data model over time. Thus,

* Corresponding author.
E-mail address: jjurado@ujaen.es (J.M. Jurado).

https://doi.org/10.1016/j.future.2022.03.022

this integrated system is, on the one hand, characterized by 3D point clouds representing the geometry of vegetation, soil and the rest of environmental elements. On the other hand, each of these 3D points are mapped with semantic attributes from different remote sensing devices such as multi- or hyper-spectral cameras. In short, both point clouds and multi-source images are processed together in order to map relevant image-based characteristics for each 3D point. Reviewing the scientific literature, we find that this problem is similar to that posed in the technique called projective texture mapping [3–5]. Originally, the aim of that technique was to have additional effects on the realistic image synthesis field of research, to cast shadows or render translucent objects [6]. This process poses a prevalent challenge in the current scenario, considering the high number of high-resolution images and the huge size of point clouds, formed by several hundreds of millions of 3D points. This way of enriching a 3D model is disruptive and allows us to reach a deep knowledge about natural environments. In order to achieve such an objective, multidisciplinary teams must work together and specific hardware with high computational and storage capabilities are required.

The problem at hand is complex, not only because of the computational time required. The main question is: which are the pixels from many different 2D images that correspond with a specific 3D point? Sensors attached to drones obtain images with a dependency on the angle and position of the camera. Moreover, some areas are non-visible because they are self-hidden from the zenith capture position, especially trunks and lower parts. Thus, the problem has two different aspects: (1) the pixels-point matching and (2) the occlusion culling issue.

To summarize, we need to look for a mechanism (1) to process the heterogeneous information coming from different UAV-based sensors, (2) to match semantic and geometric data considering problems associated with occlusion and the viewpoints of cameras. To address the problem, traditionally, raw datasets (images and 3D models) have been stored on hard disks to be post-processed on powerful desktop computers. However, a more dynamic approach would allow us to inspect and analyze the resulting fusion of collected data on site, using portable equipment, e.g., a laptop. Even in the future it could take advantage of 5G technology to transmit to a local device the information during the fly, or even perform this processing on ubiquitous processing systems processes attached to the UAS.

In any case, to deal with the challenge of processing huge amounts of input data from different sources, general-purpose computing on graphics processing units (GPGPU) techniques are used to take advantage of parallel and distributed computing strategies. GPGPU techniques can currently be applied to devices with very different sizes, memory and computational capacities. In contrast to classical sequential methods, the computational resources offered by GPU devices suppose a great opportunity to accelerate image-based operations, 3D projections, geometric transformations and occlusion tests. However, there are still many limitations that should be addressed. GPGPU algorithms must achieve a good balance distribution of compute-intensive tasks considering the available resources and consuming time of data transfers between the CPU and GPU. In parallel, the proliferation of IoT (Internet of Things) devices is transforming almost every industry. This involves the development of embedded systems, System on Chips (SoC), that bring new capabilities to the edge, accelerating product development and deployment at scale. Therefore, new strategies are highly demanded to ensure not only efficient methods but also the processing of large datasets.

In this scope, we propose an automatic and out-of-core pipeline to map remote sensing images on huge 3D point clouds. The main purpose is to combine, under the same data model, the geometric information of point clouds with multispectral data. As a result, enriched 3D models with additional environmental information are generated.

In this paper we address this problem by computing different dataset sizes and using several GPU devices emulating three different scenarios: (1) desktop computer, (2) laptop and (3) SoC platform that might be attached to any capture system. In terms of GPGPU programming, we want to contribute with:

- An automatic approach to take full advantage of parallel programming in GPU for image mapping on huge 3D models.
- The development of an out-of-core method which is not dependent on the available GPU resources.
- The optimization by asynchronous transfers for a parallel execution of mapping and occlusion tasks.
- The proposed solution for on-site data processing using discrete GPUs and embedded systems.

Several datasets have been used to test the applicability and the effectiveness of the proposed method. Our proposal is integrated in the so-called GEU (Geospatial and Environmental tools of University of Jaén) framework [7], which is briefly presented in Section 3. In this platform we have integrated three different GPU configurations to simulate different processing functionalities, addressing this way the possibility of performing on-site computation. Current research in the field of remote sensing, earth observation and big data processing can benefit from our method. The generation of enriched 3D models from spectral and high-resolution data by GPU enables new approaches to study, analyze and assess the conservation and the evolution of our environment.

The paper is organized as follows: Section 2 presents a brief review of related works. Section 3 describes the framework GEU on which this method is integrated. The problem statement, input datasets and the proposed algorithm are included in Section 4. The computational performance is tested in Section 5. Section 6 presents a discussion about the results. Finally, Section 7 summarized the main conclusions derived from this work.

2. Related work

Data management in natural environments is an increasingly interesting topic due to the emergence of disruptive acquisition technologies. Large areas of forests or crops can be remotely monitored using different capturing devices in order to estimate the biometric values of plants, the harvest, as well as to detect different plant diseases [8–10]. Some well-known techniques for the generation of 3D data in real-world scenarios are: Radio Detection And Ranging (RaDAR) [11], Light Detection and Ranging (LiDAR) [12] or structure-from-motion (SfM) [13].

Definitely, the reduction of the size and weight of remote sensing systems bring up the possibility to be coupled on UAVs [14–16]. This brings the opportunity for the monitoring of large extensions and difficult-to-access natural areas. Accordingly, multi-source remote sensing devices may be used together to increase the knowledge of dynamic environments. These diverse capture techniques have in common the huge amount of data recorded, usually given as images or point clouds. This also generates the search for new methodologies to process this data and apply the results to the different areas of interest.

The limitations of 2D zenital images are better overcome by obtaining a three-dimensional representation [17,18]. Genarally, the subsequent treatment of images is usually oriented to generate 3D models of the surveyed areas [19,20]. In the las few years, a wide variety of methods have been proposed for the detection of individual crops and trees by taking advatanges

of using 3D geometric data [21,22]. The resulting millions 3D points are processed to reduce rendering times by calculating occluded surfaces in scanned or synthetic scenarios [23,24]. The geometric modeling of observed entities allow us to study the morphological structure of trees, as well as the volume measure of the vegetation cover according to the shape of reconstructed surfaces [25].

In any case, a natural environment is not only characterized by its geometric representation. On the other hand, different sensing devices are able to provide high-resolution spectral information capable of parameterizing, and therefore, characterizing the vegetation of these spaces. Thus, using a combination of imaging sensors, we generate multi-resolution datasets that should be merged and jointly analyzed [26] in order to reveal hidden and meaningful features of natural environments. In this field, the most commonly used UAV-based sensors are focused on the acquisition of multispectral-, hyperspectral-, and thermal- imaging. For instance, the first of these devices provides an accurate measurement of reflectance in a few number of narrow-bands. In the near-infrared (NIR), vegetation can be recognized because this band is less sensitive to chlorophyll. By contrast, the green and red bands are very useful to study the reflected light by the tree canopy in the visible range. Finally, the red-edge (REG) band captures the reflectance between the Red and NIR and plays a key role to detect a key contrast from the visible to infrared light. In a similar manner, a thermal camera is able to solve many ecological questions regarding the ecosystem metabolism and the impact of heat waves and drought [1,27]. Likewise, similar techniques have been applied for the detection of plant diseases [2,27].

According to the UAV's capabilities for masive data acquisition through the use of different sensors, one of the main challenge is to develop a heterogneous data integration model, in which significant variables of surveyed entities can be correlated. The integration of spatial and spectral information into the same framework allows a more detailed analysis of crops or forests. The use of this framework prevents the comparison of partial results, which have been generated from different data sources, especially when they come from data with different dimensions (2D/3D), resolution or reference systems. Recent work are focused on multi-source data fusion or image mapping in order to provide new tools for data inspection and analysis [28,29]. The foundation of this task involves the search of those pixels, corresponding to multiple images, that have captured a specific 3D point of the point cloud. Each of these images provides an additional knowledge about this point, only if the area represented by the point is directly visible from the camera position.

Image mapping process on dense 3D models requires a high computational effort. This issue arises when multiple 2D images obtained from these sensors, which are taken from different points of view, must be appended to the 3D point cloud as complementary data. This fusion of spectral and structural features of real-world objects (artificial or natural entities) is in fact the problem of mapping pixels from input images to points with (X, Y, Z) coordinates. The result is a 3D information system that can be automatically enriched with spectral or thermal information. Moreover, this system can be fed with the information from other sources and over time to provide multi-temporal monitoring. Nevertheless, not all points are visible for all images. The tree canopy can be observed from many images, but those in the sides or even in the lower parts, are only depicted by the sides. Therefore, for each image, all pixels are mapped on a specific subset of 3D points, those ones that are directly visible from that viewpoint. Many others will be self-hidden by the rest of geometry [30–32]. Again, there is a huge number of combinations that must be processed to determine visibility.

As stated above, the processes to deal with such amounts of remote sensing information are computationally very expensive. One way of reducing this computational cost is to apply any parallelization approach [33–35]. Image mapping can take advantage of the computing capabilities of cloud computing [36] or current graphics cards to manage amounts of images [35,37,38], perform image analysis by quaternion moments [39] or 3D point clouds.

Among all the GPGPU technologies, CUDA (Compute Unified Device Architecture) is one of the most widely used parallel computing platform developed by NVIDIA that includes a suite of tools. This provides the programmer with an abstraction layer to take advantage of the GPU's parallel computing capabilities. Compared to other alternatives such as OpenACC or OpenCL, CUDA is one of the most widely used API's for GPGPU programming. In general, CUDA achieves a significant performance increase compared to these other solutions [40–42]. For instance for extracting features from high resolution imaging [43,44], for fractal mapping [45] or for mapping multispectral aerial images on orthomosaics [46]. For the calculation of visibility, CUDA has also been used in image processing or considering the scene geometry [47]. Also, in the field of natural environments and crops interesting results were found out [33]. In parallel to the rapid development of discrete GPUs that provide more and more cores to accelerate high computing tasks, other platforms based on embedded systems also take advantage of GPU for real-time data processing. Recent researches have used System on a Chip (SOC), which can be coupled to robotic systems, for multi-camera visual SLAM [48] and UAV-based 3D modeling [49] and accelerating image fusion algorithms using CUDA on embedded industrial platforms [50]. Besides efficient GPU development, when the volume of data exceeds the storage capacity we have to resort to out-of-core solutions. However, although there is a vast literature in the computer graphics field about out-of-core problems, to our knowledge, there are no proposals to map remote sensing images on dense 3D point clouds. In general, we found two key issues among out-of-core proposals, on the one hand, developing efficient algorithms, as we have already mentioned, and on the other hand, minimizing the traffic of data between storage devices. To accomplish these goals, most of the solutions use precomputed data structures usually in combination with an appropriate caching scheme. The most common spatial data structure is the octree. Richter et al. [42] generated an octree for change detection in 3D point clouds. [51] used it to load photons in a daylight modeling scenario. [52] proposed a point-based global illumination. [53] presented an algorithm for constructing a sparse voxel octree from extremely large triangle meshes. According to other data structures, Sarton et al. [54] defined a pyramidal mipmap and a page table for an interactive visualization of large volume data.

Other approaches used hierarchical linear bounding volume hierarchies in a path tracer [55]. [56] took advantage of the min-heap data structure to allocate rendering resources beyond the memory capacity. Instead of data structures, other solutions proposed compression schemes [57,58]. [59] presented a library that enables out-of-core implementations on cloud environments of data-parallel kernels for accelerators. In our proposal, the characterized point cloud serves as a valid 3D model for the identification of plant diseases, determination of plant growth, etc. Hence, it is necessary that the characterization process uses the data with maximum accuracy in terms of geometric and spectral resolutions. Consequently, hierarchical data representations or compression schemes are not useful in our case study. Our solution focuses on an adequate arrangement of data and a caching scheme to minimize data transfer as well as on the development of efficient techniques for mapping remote sensing images on 3D point clouds. Thus, we aim at providing a pipeline

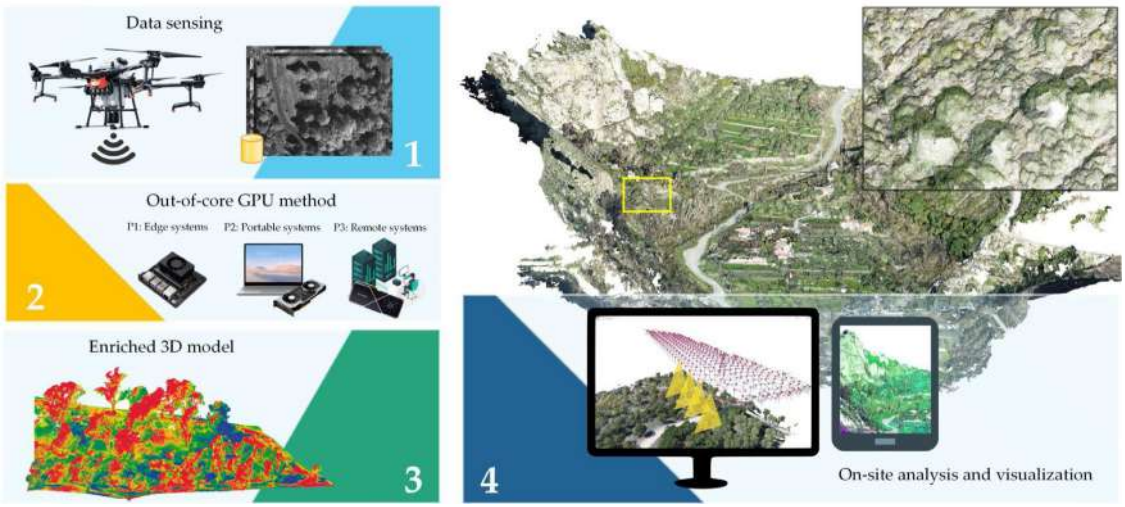


Fig. 1. Overview of the proposed solution considering four main aspects: (1) firstly, remote sensing data collected by drones, (2) secondly, efficient GPU data processing on different platforms, (3) the generation of enriched 3D models with spectral, thermal or other sensing attributes, (4) the on-site visualization and analysis of resulting data.

for the generation of enriched 3D models from multi-sensorial images on-site. To demonstrate the applicability of the proposed solution we have considered both (1) discrete GPUs and (2) embedded systems. On the one hand, a consumer-grade laptop and a PC were used for testing the proposed algorithm. On the other hand, SoC GPUs, which can be mounted on drones, were considered to allow us to generate enriched 3D models during the acquisition process.

3. Overview of GEU

GEU framework [19] is a disruptive solution to generate and process 3D models which can be characterized by multi-source data. Thus, real-world scenarios are highly detailed by the geometry and semantic features mainly extracted from high-resolution and spectral aerial images. Consequently, GEU involves many applications in many research domains in order to analyze the conservation of natural environments, to simulate real and physical phenomenons and to classify different materials or target objects. The use of GEU enhances the development of a multi-disciplinary research related with (1) precision agriculture [25,60,61,61,62], (2) computer graphics [63–65] and (3) computer vision [19,66–69].

GEU takes as input large scale datasets in order to characterize both urban and natural scenarios of the real world. Thus, point clouds can be enriched from sensing data captured by drones. The geometry of real-world environments is characterized by the temperature or spectral reflectance in order to describe the main features of observed objects. A graphical overview of the proposed solution is shown in Fig. 1. Regarding the challenges of this method, most dense 3D models are often subsampled in order to enable the operations of image mapping and occlusion tests. Moreover, high hardware requirements are needed in order to process sensing and massive data. This work aims to overcome this problem by the development of an out-of-core method that focuses on the use of GPU to ensure an efficient performance of time-consuming tasks related with heterogeneous data fusion on huge point clouds.

Table 1
Description of datasets used in this study.

	Nb. of Images/ Nb. of 3D points (M)			
Flights	F1: 180		F2: 1350	
Datasets	D1: 66	D2: 270	D3: 542	D4:1084

4. Materials and methods

4.1. Description of datasets

In this study, a large-scale forest area of ∼10 hectares (ha) was covered. Two different unmanned aerial systems (UAS) were used for the data collection process. On the one hand, the DJI Matrice 210 quadcopter was the flight platform used to carry the multispectral sensor. This device is formed by four lenses and it provides us four different images, for each one the reflected light intensity in a certain band of the electromagnetic spectrum is captured. These are green (530 nm to 570 nm), red (640 nm to 680 nm), near infrared (770 nm to 810 nm) and red-edge (730 nm to 740 nm). In this study, two sets of images were collected at different flight heights. The first one (F1) was planned at 120 m and 180 images were acquired. The second one (F2) was deployed at 40 m and 1350 images were collected.

On the other hand, the Phantom 4 RTK was the flight platform used for the capture of high-resolution RGB images. This drone is equipped with a digital camera (∼20 MP), so that multiple overlapped images were collected in order to generate the 3D model. After applying photogrammetric and filtering techniques [25], four datasets were generated with a different point cloud density. In summary, Table 1 shows the 3D models and sets of images used for testing our GPU-based method varying the geometric complexity (number of millions of points) and the number of spectral images to be mapped on the point clouds. In order to understand the dimensionality of the problem, the specific notation and formulation required to carry out each step is presented in Section 4.2.

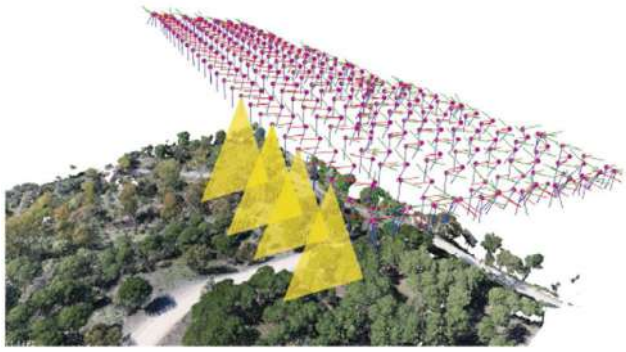


Fig. 2. A spatial correlation is defined to set the point clouds and all cameras in the same reference system. Thus, a view frustum is defined for each camera and a set of 3D points are selected to be mapped.

4.2. Problem statement and notation

Image mapping on dense point clouds presents one of the most prevalent problems to characterize 3D models. The complexity and high size of real-world scenarios pose a challenge for multi-source data fusion. Significant features of observed objects can be measured using different sensors but the integration of this heterogeneous data with the 3D geometry requires a high computational effort. The process to map every pixel on its corresponding 3D surface considering self-occlusion and visibility in the image from the camera viewpoint is not a trivial task.

Formally, the input dataset D is composed by a 3D point cloud P of size N, P=$\{p_0, p_1, \ldots, p_{n-1}\}$. It is also formed by M images or captures, C=$\{c_0, c_1, \ldots, c_{m-1}\}$, each of them with a resolution of R pixels, so that each c_i in C contains the set of pixels $c_i = \{px_{i,0}, px_{i,1}, \ldots, px_{i,R-1}\}$. Only a subset of the points in P is visible for each image, since most points of P are not within the view-frustum of the camera, or these are directly occluded by other closer surfaces to the viewpoint of the camera. Consequently, a selection of those points which are inside of the view-frustum are candidates to be considered for each image (Fig. 2). Next, a second test is necessary to detect self-hidden geometry. It means that a 3D point may be occluded by a part of the geometry of the same model from the camera viewpoint. In this study, multispectral images are used but the procedure is the same for any source or type of images. This mapping will enrich the three-dimensional nature of huge point clouds with significant information related to the reflectance of sunlight at microscale in a huge real-world scenario. In any case, each point p_j in P can be associated with an undefined set of pixels coming from a subset of images in C. Therefore, finding which point, $p \in \boldsymbol{P}$, is visible from a given camera, c_i, where i ∈ [0, M-1] and a given pixel $px_{i,k}$ where k∈ [0, R-1], is a computational complex task. This operation is iterated for all pixels and cameras, besides solving the self-hidden problems. A naive solution is proposed by Jurado et al. [19] and it is presented in Algorithm **1**. This exhaustive method requires O(NMR) to map each point on the images, where N represents the number of points, M the number of cameras and R. the image size. Moreover, a further occlusion detection must be performed to select visible points, with an additional O(MR*T), where R* is the number of pixels which contain one projected point at least and T is the average number of points per pixel. This process, based on a sequential execution on the CPU, cannot be computed on the fly for mapping several thousand images on large-scale models formed by hundreds of millions of 3D points.

The problem at hand can be considered as embarrassingly parallel as it requires applying the same script to multiple data, images and 3D points. The type of operations required for the algorithm are pure matrix algebra, just what graphics card hardware is designed for. Observe that the graphics processors for the so-called GPGPU computing, are SIMD architectures. Therefore, graphics cards expose primarily data parallelism, with thousands of simultaneous threads performing these tasks in parallel on independent data. Likewise, according to the huge amount of sensing data and dense 3D models to be processed, our approach is based on an efficient out-of-core method. In this way, our solution has no dependence on memory (VRAM) requirements allowing us to launch the proposed method using commodity hardware.

Algorithm 1: Naive method for image mapping on 3D point clouds on CPU.

Input:
$P = \{p_0, p_1, \ldots p_{n-1}\}$ a point cloud in R^3
$C = \{c_0, c_1, \ldots c_{m-1}\}$ captures or images such as each c_i contains the set of pixels $c_i = \{px_{i,0}, px_{i,1}, \ldots px_{i,R-1}\}$
Output:
S: an array of matrices with the size of input images. For each pixel ($px_{i,j}$), the id of the closest projected point (p_i) is stored.
begin
for all points $p_j \in P$, $j \subset [0..n-1]$ **do** (points in the cloud)
___ **for all images** $c_i \in C$, $i \subset [0..m-1]$ **do** (all images)
_____ **for all pixels** $px_{i,k} \in c_i, i \subset [0..R-1]$ **do** (all pixels)
if the pixel $px_{i,k}$ maps the area of p_j
and not $\exists\, p_s \in P, \forall s \neq j$ such as p_s occludes p_j
then $S[c_i][px_{i,k}] = p_j$
end

Algorithm 2: Mapping of a 3D point in a CUDA thread

Input:
$P = \{p_0, p_1, \ldots p_{n-1}\}$ a section of the point cloud in R^3
I : an image to be projected formed by its rotation matrix and position vector..
Metadata of an image.
Output:
$PointId$: index of a 3d point in the dataset being processed.
$Dist$: distance 3d point - camera (depth or z value)
$PixelId$: image coordinates once projected.
begin
*task_id = blockIdx.x * blockDim.x + threadIdx.x;*
r = sqrt(pow(x, 2) + pow(y, 2));
*theta = (2 / π) * atan(r / z);*
*p = theta + p[1] * theta * theta + p[2] * theta * theta * theta + p[3] * theta * theta * theta * theta;*
x_h *= (p * X) / r;*
y_h *= (p * Y) / r;*
*xd = (C * x_h + D * y_h + cy);*
*yd = (E * x_h + F * y_h + cx);*
if *xd < 0 or xd > I.rows or yd < 0 or yd > I.columns* **then**
___*return; // point out of viewing volume*
else
___*PixelId = xd * I.columns + yd;*
___*Dist = position$_{ci}$ - p_i|*
___*PointId = task_id;*
end

4.3. GPU kernels

The proposed solution aims to accelerate the procedure for fusion UAV-based imagery with dense point clouds of real-world scenarios. For this purpose, two main tasks have to be developed: (i) image mapping and (ii) occlusion. Our out-of-core method enables the processing of high-resolution 3D models and a large number of images, even for both user-grade GPUs and embedded systems with a lower GPU memory capacity. The size of the 3D model is divided into n partitions to be independently processed on the GPU. Two CUDA kernels perform the mapping and occlusion operations for every point of its corresponding block. As many GPU threads as points in the block are spawned, each thread computes the mapping and the occlusion for just one 3D point. These operations are described in more detail in Sections 4.3.1 and 4.3.2.

4.3.1. Image mapping

This stage is carried out in the first part of the main CUDA kernel and the aim is to get the image coordinates for each 3D point. To ensure a correct mapping of aerial images on the 3D model, both cameras and the point cloud must be located in the same reference system. In this study, input data were co-registered using the ICP algorithm [70] as a step of the preprocessing phase.

According to the image deformation due to fish-eye lenses of the multispectral sensor, a polynomial distortion (ρ) and a fisheye affine matrix are calculated to transform every 3D point (X, Y, Z) to the corresponding image coordinates ($x_d y_d$) To this end, Eqs. (1) and (2) are used as follows:

$$\rho = \theta + p_2\theta^2 + p_3\theta^3 + p_4\theta^4 \tag{1}$$

where: $\theta = \frac{2}{\pi} \arctan\left(\frac{\sqrt{X^2+Y^2}}{Z}\right)$; $\theta \epsilon [0, 1]$ and p_2, p_3 and p_4 are the coefficients of polynomial fisheye, provided by the sensor.

$$[x_d y_d] = [CDEF]\,[x_h y_h] + \left[c_x c_y\right] \tag{2}$$

where C, D, E and F are the parameters of the affine deformation, c_x and c_y are the principal point in pixel coordinates and $[x_h y_h] = \left[\frac{\rho X}{\sqrt{X^2+Y^2}} \frac{\rho Y}{\sqrt{X^2+Y^2}}\right]$.

According to this formulation, Algorithm **2** is carried out to map every 3D point on captured images.

The output data of the mapping method are three values: point-camera distance, point id and the pixel coordinates. Each CUDA thread is responsible for calculating the projection of every 3D point. The id of every thread is assigned considering the dimensions of the CUDA block (32, 64 and 128). Then, the polynomial fisheye vector (ρ) and the coefficients of the fisheye affine matrix (C,D,E,F) are used to calculate the projection for each 3D point to the distorted image plane, as indicated in Eqs. (1) and (2). Finally, if the projected point is inside the image range, the id of the pixel is calculated considering its corresponding pixel coordinates on the image, the id of the thread is considered as the id of the projected 3D point and the distance from the position of the camera to the point is also calculated. Thus, every 3D point is mapped on the image plane by obtaining the pixel position ($x_d y_d$). A graphical example of this computation, but only considering the mapping of one block on one image, is presented in Fig. 3.

4.3.2. Occlusion

The image resolution captured by spectral sensors is significantly lower than the 3D point cloud, which was modeled by using overlapped RGB images. Consequently, multiple 3D points are mapped on the same pixel (Fig. 3). In addition, points which are behind others should be discarded considering the occlusion for each image. To overcome this problem, occluded points are

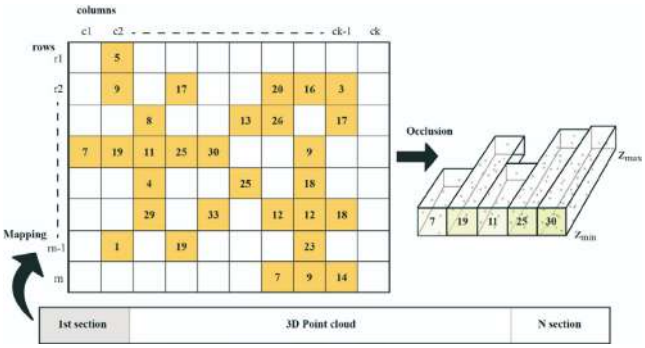


Fig. 3. The scheme of image mapping on the 3D point cloud. Yellow pixels contain one projected point at least. If multiple points are mapped on the same pixel, these are ordered by the distance between the 3D point and the camera positions.

managed by using a depth buffer for each pixel, as shown by Algorithm **3**. The distance from the 3D point to the camera position was previously calculated by the mapping method, so a comparison with the closest point in the block projected to the same pixel so far is enough to discard the current point or update the depth buffer. Thus, an array of size the resolution of the camera image is kept during the processing of a pair dataset block-image. This depth buffer is updated by an *atomicMin()* operation, to protect each array value from concurrent accesses. Finally, once the block has been fully processed, a snapshot (2D matrix) of the occlusion test is saved for each image. This array of snapshots is used later to check the occlusion for the following blocks of the point cloud.

Algorithm 3: Occlusion test in a CUDA thread

Input:
PointId*: Index of 3d point in dataset*
PixelId*: Image pixel where 3d point has been mapped*
Dist*: Distance to camera from 3d point*
Output:
Z-buffer: Updated depth buffer
pixel = PointId | (Dist << 32);
atomicMin(Z-buffer[PixelId], pixel);

4.4. Out-of-core method in GPU

GEU exploits the described mapping & occlusion kernel (Algorithms **2** and **3**) mainly in a post hoc scenario, once one or more drone flights are completed; either in-situ, working in the field with a laptop, or in the office, using a potentially more powerful system such as a workstation. In both use cases, it is necessary to execute the kernels for every image on the whole dataset to have accurate occlusion results. Hence, an out-of-core approach that allows the processing of arbitrarily large point clouds is needed. In Section 4.4.1 we describe an efficient out-of-core method for handling point clouds in CUDA on a discrete GPU independently of their size and the amount of VRAM available on the platform.

An alternative scenario is described in Section 4.4.2, moving from a post hoc to an online processing for an edge computing approach. In this solution, the mapping & occlusion kernel in Algorithms **2** and **3** is now executed by a SoC equipped in the drone to obtain the real-time mapping for each image on the fly, every time a new shot is taken by the camera.

4.4.1. Discrete GPUs

This method makes a partition on the dataset according to the usable GPU memory and transparently processes the different

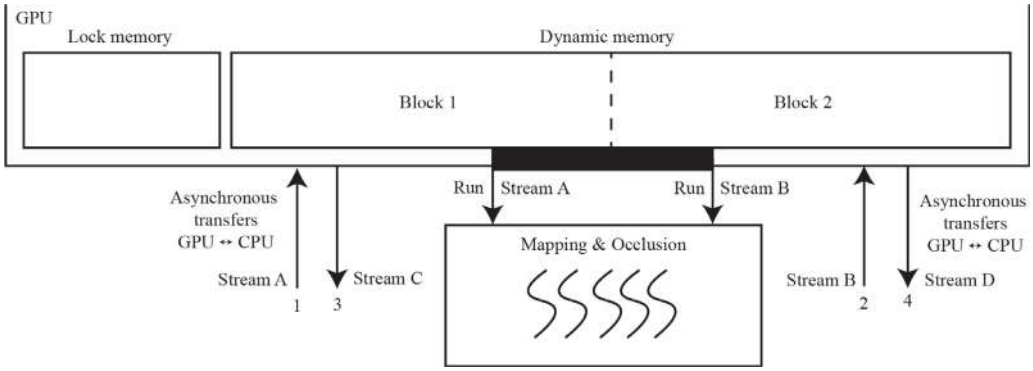


Fig. 4. Workflow of the proposed out-of-core method for both operations: 3D mapping and occlusion tasks.

blocks for every image, obtaining the mapping of all images on the point cloud at the end of the whole process.

One of the main challenges in a GPGPU algorithm is to hide the CPU<->GPU transfers by overlapping computation and communication as much as possible. Moreover, it is also essential to get an effective overlap of CPU and GPU computation, minimizing the bubbles produced when one of them is stalled waiting for a computation from the other. Finally, another basic rule is to use as few transferences as possible, exploiting every data previously moved to GPU before loading new data, so avoiding to send the same data multiple times.

In our approach, multiple CUDA streams exploit asynchronous transfer operations and kernel executions to achieve an efficient result. Hence, while points in a block are being processed in a CUDA kernel, the next block is being transferred to GPU. Likewise, while a camera is being processed in the current kernel, the results from the previous camera are being transferred to CPU, where those data are merged with previous results for that camera concurrently with the GPU computation of the next one. Fig. 4 shows the workflow of the proposed method. According to the available memory in the GPU, this is divided into two parts: (i) lock memory and (ii) dynamic memory. In the first one, temporal and constant variables, as well as the position vector $\mathbf{p} = (x, y, z)$ and rotation matrix for each image, are stored for the entire program execution. The second section of memory is occupied by pairs of blocks of the 3D point cloud. The size for each block is automatically adjusted considering the half of the available memory after storing data in the lock memory. Firstly, the first block is transferred from CPU to GPU (*Stream A*). Secondly, while the first block is processed by the mapping and occlusion kernels (performing the computation on the points of that block for every image), the second block is transferred to the GPU (*Stream B*). Finally, when the memory space for a block is released, the next block is transferred to the GPU asynchronously. While a dataset block is in the GPU memory, all the cameras are processed for that block by the same CUDA stream that previously loaded that block in GPU. Once each image is computed for that block, it is transferred to the CPU by a different stream, to overlap this communication with the computation of the following image (*Stream C* is in charge of GPU->CPU transfers for data computed by *Stream A*, and *Stream D* for data computed by *Stream B*). This workflow is presented in Algorithm **4**.

Algorithm 4: Main tasks for computing the method on GPUs.

Input:
$P = \{p_0, p_1, \ldots p_{n-1}\}$ a point cloud in R^3
$C = \{c_0, c_1, \ldots c_{m-1}\}$ captures or images such as each c_i contains the set of pixels $c_i = \{px_{i,0}, px_{i,1}, \ldots px_{i,r-1}\}$

Output:
S: an array of matrices with the size of input images. For each pixel (px_i), the id of the closest projected point (p_j) is stored.

```
begin
Transfer of c_i from host to device
for each pair of blocks (P_k, P_{k+1}) in P do
  Async. transfer of block k from host to device <Stream A>;
  Async. transfer of block k+1 from host to device<Stream B>;
  for each c_i in C do
    kernel<<..., Stream A>>(block k)Mapping&Occlusion;
    Async. transfer of image from device to host <Stream C>;
  for each c_i in C do
    kernel<<..., Stream B>>(block k+1)Mapping&Occlusion;
    Async. transfer of image from device to host <Stream D>;
end
```

4.4.2. GPU-based embedded systems

Systems on a Chip (SoCs) include on an embedded system several components: CPU, GPU, memory, power management, high-speed interface, and more. This hardware solution brings to develop novel applications for energy-efficient autonomous machines by reducing the size, weight and energy consumption. In terms of performance capabilities and power supply requirements, it delivers an adequate balance offering CUDA cores for NVIDIA platforms, a high-speed I/O and around 100–200 TOPS for multiple concurrent interfaces. The power consumption of these devices is around 15 W whereas a desktop PC consumes 180–200 W on average, i.e., an order of magnitude lower. These devices open the door for embedded and edge computing approaches that demand increased performance but are constrained by size, weight, and power budgets.

According to the use case of this study, these systems become ideal to be equipped in a drone and take advantage of its computational capabilities during a flight. Fig. 5 illustrates the system we propose. The drone is equipped with a smart module that includes the embedded system (SoC), an external hard disk and a power battery. This allows processing the collected images during the drone flight and transmitting the results to the client machine.

In comparison to platforms with discrete GPUs, despite its hardware limitations, these embedded systems have the advantage of ubiquitous processing of the captured information. On

Table 2
Description of hardware platforms and compilers used to test the proposed method.

	Platform P1 SoC: Jetson Xavier NX	Platform P2 Laptop	Platform P3	
			Workstation A	Workstation B
GPU	384-core Nvidia Volta Compute Capab. 7.2 SoC RAM: 8 GB	GeForce GTX 1050 Compute Capab. 6.1 VRAM: 4 GB	Tesla K20 m Compute Capab. 3.5 VRAM: 5 GB	GeForce RTX 2080Ti Compute Capab. 7.5 VRAM: 11 GB
CPU	Nvidia Carmel ARM v8.2 6 cores SoC RAM: 8 GiB	Intel i7-7700HQ 2.8 GHz 4 cores (8 SMT) RAM: 32 GB	Intel Xeon E5-2660 2.2 GHz 8 cores (16 SMT) RAM: 64 GB	AMD Ryzen Threadripper 1950X 16 cores (32 SMT) RAM: 64 GB
CUDA & Driver	CUDA 11.0.194 in Jetpack 4.6	11.5 & 495.46	11.0.194 & 460.56	11.2 & 460.32.03
C/C++ Compiler	GNU 7.5.0	GNU 11.2.1	GNU 8.3.0	GNU 8.4.0

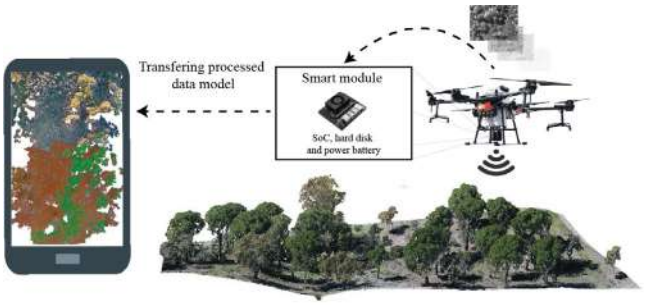


Fig. 5. A graphical representation of the proposed solution using our method on a SoC platform.

the one hand, it avoids the generation of large volumes of data after the capture process, synthesizing the relevant information. On the other hand, throughout the data collection process, an information model is ready to be analyzed. The computational capabilities of SoCs allow us to deploy the proposed out-of-core pipeline. Initially, the geometric 3D model, as a point cloud, is stored on the external hard disk of the smart module. This one is loaded on the SoC at the beginning of the drone flight. During the drone flight, each image is automatically sent to the smart module, in which the SoC executes the kernels described in Section 4.3. Algorithm **5** aims to perform the mapping and occlusion tasks for a specific image on the SoC.

Algorithm 5: Main tasks for computing the method on SoC.

Input:
$P = \{p_0, p_1, \ldots p_{n-1}\}$ a point cloud in R^3
C one image which contains the set of pixels
$C = \{px_0, px_1, \ldots px_{q-1}\}$
Output:
M: a matrix where for each pixel (px_i), the id of the closest projected point (p_j) is stored.
begin
if (P > Size of SoC memory)
___Transfer of P_k from hard disk to the SoC
else
___Transfer of P from hard disk to the SoC
Async. transfer of C from drone camera to the SoC
kernel<<..., Stream A>>(P) Mapping&Occlusion;
Async. transfer of results from the SoC to the hard disk;
end

Regarding usual point cloud densification and memory capabilities of current SoCs, our method assumes that the one block of the point cloud for a set of images can be fully stored on the SoC memory. Firstly, according to the drone position, one block of the point cloud is transferred from an external disk to the SoC memory. This one is maintained in memory whereas the drone is located into its bounding box and all collected images are processed. Secondly, after the capture of one image, this is sent to the SoC and computed there. Finally, the resulting matrix which stores the correlation between 3D points and pixels is stored on the external disk, and then, the results can be sent to the client machine. The evaluation of this last step is out of the scope of this study. Therefore, image processing is performed in accordance with the in-flight image capture process. Likewise, the results obtained can be evaluated and decisions can be made during the data capture session, thus reducing the number of flights and, consequently, lowering time and costs.

Table 3
Execution time for mapping one image on the 3D model using the Platform P1.

Datasets	Time (ms) per image on average
D1 (66M)	60
D2 (271M)	237
D3 (542M)	474
D4 (1084M, $P_k = 271$M)	237

5. Experimental results and performance evaluation

In order to evaluate the impact of the presented work, several 3D scenarios of the real world were tested, considering different spatial resolutions. The execution time, the cover area and the size of the input dataset are analyzed to demonstrate the effectiveness of the proposed solution.

Four different platforms were used as a testbed for our method, covering different NVIDIA GPU microarchitectures and compute capabilities. Table 2 summarizes their main features. An NVIDIA Jetson Xavier NX (Platform P1) was used to cover the edge computing scenario described in Section 4.4.2, whereas three different discrete NVIDIA GPUs (Platforms P2, P3-A and P3-B) were employed to test the main post hoc approach, depicted in Section 4.4.1.

Tables 3 and 4 depict the experimental results obtained for the two scenarios described in Section 4. Namely, the total execution time is shown, just discarding the file loading time from disk. Three runs were launched for each setup, and the best times were included in the table. Results for 128 threads per CUDA block are shown in all cases. 32, 64 and 256 threads per block were also tested, with similar execution times in most cases, only slightly better for 128 threads/block.

Table 3 shows the average time Platform P1 needs to compute the mapping of the point cloud on an image every time a new shot is taken by a camera in the drone. Datasets D1, D2 and D3 can be completely loaded in Jetson's memory, whereas for

Table 4
Platforms P2 (laptop) and P3 (workstations A & B). All times in seconds.

Flights	Datasets	Platform P2		Platform P3-A		Platform P3-B	
F1 180 images	D1 (66M)	15.30	1	11.03	1	0.64	1
	D2 (271M)	62.29	1	44.82	1	1.87	1
	D3 (542M)	125.04	4	88.14	3	3.71	1
	D4 (1084M)	536.82	7	174.84	6	6.85	3
F2 1350 images	D1 (66M)	379.93	1	80.55	1	4.46	1
	D2 (271M)	1570.96	1	327.99	1	12.67	1
	D3 (542M)	2945.30	4	654.35	3	25.27	1
	D4 (1084M)	5817.19	7	1306.65	6	49.99	3
		Time	Partitions	Time	Partitions	Time	Partitions

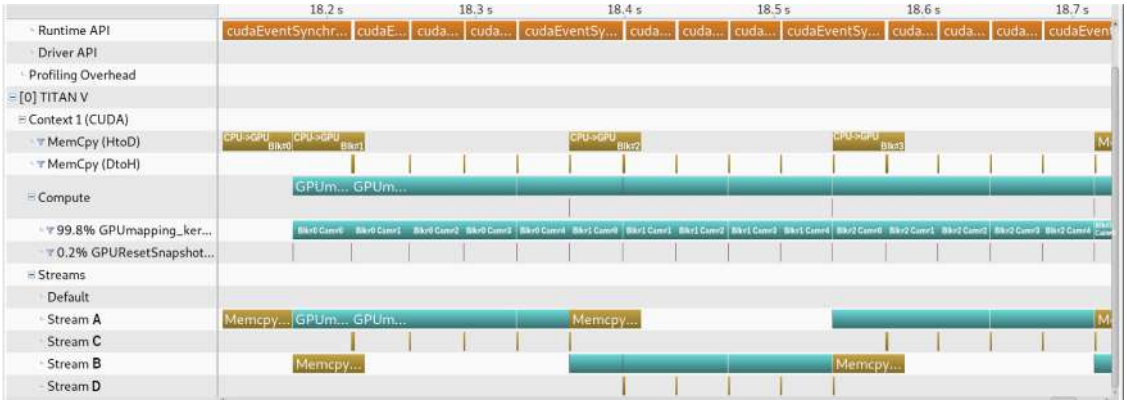


Fig. 6. Overlapping of transference and computation: detail from the nVIDIA Visual Profiler on discrete GPUs.

D4 the system needs to work with a block of ∼271M of points spanning the camera field-of-view, reserving space to overlap the loading process of the next block in memory from disk with the computation of the images for the current block.

Table 4 presents the experimental results for the post hoc scenario, with platforms P2, P3-A and P3-B computing the mapping on the point cloud for every image in the two used flights. For each platform and dataset, the execution time and the number of partitions needed are shown.

Fig. 6 depicts a time diagram from the GPU perspective of the execution of the 271M points dataset, with only 5 images for the sake of clarity. This diagram, obtained with the NVIDIA Visual Profiler, shows the effective overlapping of CPU->GPU transferences (blocks of the point cloud loaded in VRAM), GPU computation (kernels processing each block for each image) and GPU->CPU transferences (snapshots obtained for each block-image computation that are transferred to CPU to be merged with previous blocks) by means of four CUDA Streams.

The lower part of the figure shows the timeline for each of these four Streams: *Stream A* and *Stream B* are in charge of both the execution of the CUDA kernels and the CPU->GPU transfers, overlapping each other to avoid waiting for the transferences: a new block is uploaded to GPU by *Stream B* while the previous one is being processed by *Stream A*, and vice versa. Streams C and D communicate the results for each block-image to the CPU, again concurrently with GPU processing in *Stream A* or *Stream B*.

According to the resulting execution time for the most efficient platform just 50 s are required to process the large-scale scenario. Taking advantage of this method, researchers can analyze on-site the surveyed scenario in a 3D environment in which the geometry and multi-source data are represented.

6. Discussion

From the experiment results under different problem sizes and platforms, it can be seen that our out-of-core approach enables the generation of enriched 3D geometry from multi-source data by an efficient GPU-based parallelization. Memory limitations are overcome considering the spatial subdivision of the 3D model into blocks. Thus, the point cloud is mapped on captured images and then, an efficient occlusion test is performed to discard self-occluded points.

The experimental results demonstrate the good adaptation of our solution to the GPU as shown in Tables 3 and 4. Table 3 depicts the embedded system equipped in drones is adapted to the needs of the capture process and enables processing the datasets, image by image, while they are being acquired. Likewise, as can be observed in the experimental results presented in Table 4, the achieved performance is close to linear in most cases, i.e. the execution time linearly scales with the increment in data (points and images). A few exceptions present an overhead of the expected performance in the more memory demanding setups for platform P2. On one hand, dataset D4 is exactly twice the size of dataset D3, but platform P2 needs more than 4x the time in order to process D3 considering 180 images from flight F1. On the other hand, flight F2 has about 7 times the number of images in flight F1, but platform P2 needs about 25x the time to process the several datasets for F2. Therefore, it scales when increasing the number of points, but with an extra offset. This anomaly, exclusive of platform P2, is related with the RAM requirements of our out-of-core approach and the amount of RAM available in platform P2: 32 GB (P3-A and P3-B have 64 GB of RAM). The dataset D4 has 1084 million of points, which means about 13 GB (1084 M. Points $\times$ 12B per/point) of pinned memory allocated in RAM, as non-pageable memory is used for the dataset to overlap computation and communication in CUDA. The 1350 images in flight F2 need an allocation of about 12 GB of RAM (8 B/pixel $\times$ 960 $\times$ 1280 pix/image $\times$ 1350 images). Even though the system is able to effectively manage this demanding scenario, the performance is clearly worse due to the virtual memory management with an important amount of memory pages pinned.

Table 5
CPU baseline in GEU (fastest CPU, all times in seconds) where F1 is the first flight (180 images) and F2 is the second flight (1350 images).

P3-B - AMD Ryzen Threadripper 1950X 16 cores (32 SMT)				
		Sequential	OpenMP 32 thrs	Out-of-core CUDA
F1	D1 (66M)	363.9	19.3	0.64
	D2 (271M)	1492.8	76.4	1.87
	D3 (542M)	2973.6	152.0	3.71
	D4 (1084M)	5950.0	303.6	6.85
F2	D1 (66M)	2667.1	142.3	4.46
	D2 (271M)	10829.7	562.2	12.67
	D3 (542M)	21665.7	1122.9	25.27
	D4 (1084M)	43309.8	2233.6	49.99

As shown in Fig. 6, two blocks are simultaneously being managed to overlap transfer and computing operations. While the first block is under processing in the GPU, the second block is being transferred from the CPU to the GPU. This latency hiding allows us to boost the performance of the method significantly on those platforms with low memory capacity. According to the results, we can conclude that the proposed out-of-core method makes it possible to deal with arbitrary large datasets independently of the hardware platform. Thus, the experimental results show that huge datasets can be processed while maintaining a low GPU memory footprint, partitioning the point cloud with a really small penalty: the overhead introduced is relatively low in most cases, and the out-of-core processing is even faster in some cases when only 2 blocks are enough to address the memory constraints. This is possible due to the good overlapping achieved among the different tasks that can run concurrently: GPU and CPU computation and CPU<->GPU transferences.

Our results demonstrate a significant improvement considering current solutions in GEU that only support CPU-based computing. Table 5 shows the time required by the fastest platform (P3-B) to process the eight datasets on CPU sequentially (left column) and with an OpenMP parallel implementation exploiting 64 CPU threads (middle column). Comparing the results in the CPU with those obtained with our proposal (GPU) (left column), we observe that the problem is better suited to the GPU architecture and achieves more than 30 times better results than the parallel CPU solution. In this way, on-site analysis can be carried out by following our GPU-based solution and also enabling multitemporal characterization of large-scale scenarios of the real world.

In this context, novel applications are currently being developed to tackle the challenges of processing huge datasets by a high throughput computing [71–73]. Undoubtedly, the characterization of real-world scenarios implies the generation of a dense geometry to represent both dynamic and static objects and to capture other features also. Most of them are obtained from aerial images that provide meaningful information about the class or state of the observed entities. The processing and integration of multi-source data in a 3D collaborative environment by commodity hardware, using the GPU, is one of the main goals of this research. In comparison to previous work, our method provided fast results in short times allowing us to review them on-site. To our knowledge, existing methods are only focused on a sequential computation of mapping and operation tasks [25]. Therefore, several hours are required to integrate multi-source data with 3D models. Moreover, high-computational requirements must be satisfied with an expensive workstation to load in memory and process huge datasets. Our method overcomes the mentioned limitations by following an out-of-the box method. Independently of the size of 3D point cloud and set of images, our solution works properly so that even enabling remote computing by applying existing solutions such as rCUDA [74]. Thus, the observation of real-world scenarios is offered to researchers in a 3D virtual environment in a short time, less than one minute for the largest dataset and optimal platform. This opens new possibilities to explore huge scenarios that can be monitored under a centimetric accuracy and also share the results between researchers in a collaborative environment.

7. Conclusion and future work

According to existing literature, there are not any methods to allow the efficient generation of enriched 3D models using remote sensing imagery. In parallel to current advances in Earth observation and the proliferation of IoT systems, we have proposed a software solution to integrate multisensorial data and dense 3D models, which are extracted from the real world, providing near real-time results. In fact, future generation computer systems may take advatanges according to the capabilities of the pipeline proposed in this study. Thus, enabling the characterization of 3D geometry with drone data opens new opportunities to understand and analyze our environment. Be aware of the huge size of collected datasets, the proposed out-of-core method is based on GPU capabilities of heterogeneous platforms, from workstations and portable devices to embedded systems.

The proposed solution is part of the framework GEU focused on supporting research related to precision agriculture, forestry and Earth observation. The proposed advances involve the enhancement of the study and analysis of real-world scenarios by exploiting the GPU computing. In terms of the overall performance for image mapping and occlusion tasks, considering high dense 3D models, our method presents a good balance between problem size and complexity.

Through an exhaustive evaluation, considering different platforms and datasets, we have validated the results for getting an optimal spatial subdivision of the 3D model and optimal overlap of time-consuming tasks. Accordingly, the proposed GPU-based solution improves the naive version by getting an execution time on the workstation of 0.64 s for a medium-scale scenario (66 M points, 180 images) and less than 50 s for the largest dataset (1084 M points, 1350 images). These figures represent a speed-up of more than 560 times in comparison to the CPU solution and are at least 30x faster than the parallel CPU alternative. As a result, it has been demonstrated that the problem is perfectly suited for a GPU parallelization. In fact, regarding the execution times of our method on the SoC, for each image, an acceleration by 30 times is also achieved with respect to the CPU sequential solution.

As future work, we will exploit our method focusing on edge computing approaches on SoCs that integrate a GPU for data filtering, fusion and processing. This method brings the chance for the development of future generation systems that can be mounted on drones to allow us the 3D monitoring of environmental scenarios by integrating multi-source data in real-time.

CRediT authorship contribution statement

Juan M. Jurado: Ideas, Development or design of methodology, Programming, Specifically performing the experiments, or data/evidence collection, Writing – review & editing. **Emilio J. Padrón:** Development or design of methodology, Programming, Specifically performing the experiments, or data/evidence collection, Writing – review & editing, Testing of existing code components, Computing resources, Supervision. **J. Roberto Jiménez:** Ideas, Writing – review & editing, Supervision. **Lidia Ortega:** Ideas, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partially supported through the research projects TIN2017-84968-R, PID2019-104184RB-I00 funded by MCIN/AEI/10.13039/501100011033 and ERDF funds "A way of doing Europe", as well as by ED431C 2021/30, ED431F 2021/11 funded by Xunta de Galicia and 1381202 by Junta de Andalucía.

References

[1] C. Still, R. Powell, D. Aubrecht, Y. Kim, B. Helliker, D. Roberts, A.D. Richardson, M. Goulden, Thermal imaging in plant and ecosystem ecology: applications and challenges, Ecosphere 10 (2019) e02768, http://dx.doi.org/10.1002/ecs2.2768.

[2] R. Calderón, J.A. Navas-Cortés, P.J. Zarco-Tejada, Early detection and quantification of verticillium wilt in olive using hyperspectral and thermal imagery over large areas, Remote Sens. 7 (2015) 5584–5610, http://dx.doi.org/10.3390/rs70505584.

[3] P. Debevec, Y. Yu, G. Borshukov, Efficient view-dependent image-based rendering with projective texture-mapping, in: Eurographics Workshop on Rendering Techniques, Springer, 1998, pp. 105–116.

[4] C. Everitt, Interactive order-independent transparency. White Pap. NVIDIA 2, 7, 2001.

[5] P.S. Heckbert, Survey of texture mapping, IEEE Comput. Graph. Appl. 6 (1986) 56–67, http://dx.doi.org/10.1109/MCG.1986.276672.

[6] C. Dachsbacher, M. Stamminger, Translucent shadow maps, Render. Tech. 2003 (2003) 197–201.

[7] J.M. Jurado, Spectral characterization and semantic segmentation of complex 3D models in natural environments, 2020.

[8] R. Ali, A.K. Pal, S. Kumari, M. Karuppiah, M. Conti, A secure user authentication and key-agreement scheme using wireless sensor networks for agriculture monitoring, Future Gener. Comput. Syst. 84 (2018) 200–215, http://dx.doi.org/10.1016/j.future.2017.06.018.

[9] F.B.J.R. Dallaqua, Á.L. Fazenda, F.A. Faria, ForestEyes project: Conception, enhancements, and challenges, Future Gener. Comput. Syst. 124 (2021) 422–435, http://dx.doi.org/10.1016/j.future.2021.06.002.

[10] H. Zhao, C. Yang, W. Guo, L. Zhang, D. Zhang, Correction: Automatic estimation of crop disease severity levels based on vegetation index normalization, Remote Sens. 12 (12) (2020) 1, http://dx.doi.org/10.3390/rs12223761.

[11] Z. Feng, Y. Chen, T. Hakala, J. Hyyppä, Range calibration of airborne profiling radar used in forest inventory, in: 2016 IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2016 - Proceedings, IEEE International Geoscience and Remote Sensing Symposium Proceedings, IEEE, United States, 2016, pp. 6672–6675, http://dx.doi.org/10.1109/IGARSS.2016.7730742.

[12] Y. Su, Q. Guo, B. Xue, T. Hu, O. Alvarez, S. Tao, J. Fang, Spatial distribution of forest aboveground biomass in China: Estimation through combination of spaceborne lidar, optical imagery, and forest inventory data, Remote Sens. Environ. 173 (2016) 187–199, http://dx.doi.org/10.1016/j.rse.2015.12.002.

[13] J. Rahlf, J. Breidenbach, S. Solberg, E. Næsset, R. Astrup, Digital aerial photogrammetry can efficiently support large-area forest inventories in Norway, For. Int. J. For. Res. 90 (2017) 710–718, http://dx.doi.org/10.1093/forestry/cpx027.

[14] Y. Hu, Y. Yao, Q. Ren, X. Zhou, 3D multi-UAV cooperative velocity-aware motion planning, Future Gener. Comput. Syst. 102 (2020) 762–774, http://dx.doi.org/10.1016/j.future.2019.09.030.

[15] S. Samiappan, L. Casagrande, G.M. MacHado, G. Turnage, L. Hathcock, R. Moorhead, J. Ball, Texture classification of very high resolution UAS imagery using a graphics processing unit, in: International Geoscience and Remote Sensing Symposium, IGARSS, 2018, pp. 6476–6479, http://dx.doi.org/10.1109/IGARSS.2018.8519298.

[16] C. Torresan, A. Berton, F. Carotenuto, S.F.D. Gennaro, B. Gioli, A. Matese, F. Miglietta, C. Vagnoli, A. Zaldei, L. Wallace, Forestry applications of UAVs in Europe: a review, Int. J. Remote Sens. 38 (2017) 2427–2447, http://dx.doi.org/10.1080/01431161.2016.1252477.

[17] R.A. Díaz-Varela, R. De la Rosa, L. León, P.J. Zarco-Tejada, High-resolution airborne UAV imagery to assess olive tree crown parameters using 3D photo reconstruction: Application in breeding trials, Remote Sens. 7 (2015) 4213–4232, http://dx.doi.org/10.3390/rs70404213.

[18] A. Miranda-Fuentes, J. Llorens, J.L. Gamarra-Diezma, J.A. Gil-Ribes, E. Gil, Towards an optimized method of olive tree crown volume measurement, Sensors 15 (2015) 3671–3687, http://dx.doi.org/10.3390/s150203671.

[19] Juan M. Jurado, J.L. Cárdenas, C.J. Ogayar, L. Ortega, F.R. Feito, Semantic segmentation of natural materials on a point cloud using spatial and multispectral features, Sensors 20 (2244) (2020).

[20] D. Zhang, J. Shao, X. Li, H.T. Shen, Remote sensing image super-resolution via mixed high-order attention network, IEEE Trans. Geosci. Remote Sens. (2020) 1–14, http://dx.doi.org/10.1109/TGRS.2020.3009918.

[21] Juan M. Jurado, L. Pádua, F.R. Feito, J.J. Sousa, Automatic grapevine trunk detection on UAV-based point cloud, Remote Sens. 12 (3043) (2020).

[22] J.M. Jurado, M.I. Ramos, C. Enríquez, F.R. Feito, The impact of canopy reflectance on the 3D structure of individual trees in a mediterranean forest, Remote Sens. 12 (2020) http://dx.doi.org/10.3390/rs12091430.

[23] L.Q. Campagnolo, W. Celes, Interactive directional ambient occlusion and shadow computations for volume ray casting, Comput. Graph. 84 (2019) 66–76, http://dx.doi.org/10.1016/j.cag.2019.08.009.

[24] P. Shanmugam, O. Arikan, Hardware accelerated ambient occlusion techniques on GPUs, in: Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games, I3D '07, Association for Computing Machinery, New York, NY, USA, 2007, pp. 73–80, http://dx.doi.org/10.1145/1230100.1230113.

[25] J.M. Jurado, L. Ortega, J.J. Cubillas, F.R. Feito, Multispectral mapping on 3D models and multi-temporal monitoring for individual characterization of olive trees, Remote Sens. 12 (1106) (2020) http://dx.doi.org/10.3390/rs12071106.

[26] M. Imani, Adaptive signal representation and multi-scale decomposition for panchromatic and multispectral image fusion, Future Gener. Comput. Syst. 99 (2019) 410–424, http://dx.doi.org/10.1016/j.future.2019.05.004.

[27] G. Sepulcre-Cantó, P.J. Zarco-Tejada, J.C. Jiménez-Muñoz, J.A. Sobrino, E. de Miguel, F.J. Villalobos, Detection of water stress in an olive orchard with thermal remote sensing imagery, Agric. for. Meteorol. 136 (2006) 31–44, http://dx.doi.org/10.1016/j.agrformet.2006.01.008.

[28] M. Alonzo, B. Bookhagen, D.A. Roberts, Urban tree species mapping using hyperspectral and lidar data fusion, Remote Sens. Environ. 148 (2014) 70–83, http://dx.doi.org/10.1016/j.rse.2014.03.018.

[29] O. Nevalainen, E. Honkavaara, S. Tuominen, N. Viljanen, T. Hakala, X. Yu, J. Hyyppä, H. Saari, I. Pölönen, N.N. Imai, A.M.G. Tommaselli, Individual tree detection and classification with UAV-based photogrammetric point clouds and hyperspectral imaging, Remote Sens. 9 (2017) http://dx.doi.org/10.3390/rs9030185.

[30] A. Cavagna, S. Melillo, L. Parisi, F. Ricci-Tersenghi, SpaRTA tracking across occlusions via partitioning of 3D clouds of points, IEEE Trans. Pattern Anal. Mach. Intell. 43 (2021) 1394–1403, http://dx.doi.org/10.1109/TPAMI.2019.2946796.

[31] V.A. Debelov, I. Sevastianov, Light mesh: soft shadows as interpolation of visibility, Future Gener. Comput. Syst. 20 (2004) 1299–1315, http://dx.doi.org/10.1016/j.future.2004.05.027.

[32] J. Xu, J. Shan, G. Wang, Hierarchical modeling of street trees using mobile laser scanning, Remote Sens. 12 (2020) http://dx.doi.org/10.3390/rs12142321.

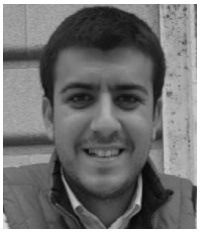
[33] L. Li, J. Gu, A. Song, H. Zheng, J. Cao, D. Zhu, Parallelization on model of ecological environment remote sensing evaluation based on GPU, Nongye Jixie Xuebao/Trans. Chin. Soc. Agric. Mach. 48 (2017) 135–141, http://dx.doi.org/10.6041/j.issn.1000-1298.2017.05.016.

[34] B. Zhao, M. Liu, J. Wu, X. Liu, M. Liu, L. Wu, Parallel computing for obtaining regional scale rice growth conditions based on WOFOST and satellite images, IEEE Access 8 (2020) 223675–223685, http://dx.doi.org/10.1109/ACCESS.2020.3043003.

[35] X. Zuo, T. Qi, B. Qiao, Z. Deng, Q. Ge, Fast parallel extraction method of normalized vegetation index, in: 15th International Conference on Computer Science and Education, ICCSE 2020, 2020, pp. 433–437, http://dx.doi.org/10.1109/ICCSE49874.2020.9201851.

[36] J. Yan, Y. Ma, L. Wang, K.-K.R. Choo, W. Jie, A cloud-based remote sensing data production system, Future Gener. Comput. Syst. 86 (2018) 1154–1166, http://dx.doi.org/10.1016/j.future.2017.02.044.

[37] D.C. de Andrade, L.G. Trabasso, An OpenCL framework for high performance extraction of image features, J. Parallel Distrib. Comput. 109 (2017) 75–88, http://dx.doi.org/10.1016/j.jpdc.2017.05.011.
[38] A. Casella, I.D. Falco, A.D. Cioppa, U. Scafuri, E. Tarantino, Exploiting multi-core and GPU hardware to speed up the registration of range images by means of Differential Evolution, J. Parallel Distrib. Comput. 133 (2019) 307–318, http://dx.doi.org/10.1016/j.jpdc.2018.07.002.
[39] A. Salah, K. Li, K.M. Hosny, M.M. Darwish, Q. Tian, Accelerated CPU–GPUs implementations for quaternion polar harmonic transform of color images, Future Gener. Comput. Syst. 107 (2020) 368–382, http://dx.doi.org/10.1016/j.future.2020.01.051.
[40] J. Kim, T.T. Dao, J. Jung, J. Joo, J. Lee, Bridging OpenCL and CUDA: a comparative analysis and translation, in: SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2015, pp. 1–12.
[41] S. Memeti, L. Li, S. Pllana, J. Kołodziej, C. Kessler, Benchmarking OpenCL, OpenACC, OpenMP, and CUDA: programming productivity, performance, and energy consumption, in: Proceedings of the 2017 Workshop on Adaptive Resource Management and Scheduling for Cloud Computing, 2017, pp. 1–6.
[42] R. Richter, J.E. Kyprianidis, J. Döllner, Out-of-core GPU-based change detection in massive 3D point clouds, Trans. GIS 17 (2013) 724–741.
[43] S.A.E. Al, Real-time parallel image processing applications on multicore CPUs with OpenMP and GPGPU with CUDA, J. Supercomput. 225 (2018) 5–2275.
[44] Y. Yuan, X. Yang, W. Wu, H. Li, Y. Liu, K. Liu, A fast single-image super-resolution method implemented with CUDA, J. Real-Time Image Process. 16 (2019) http://dx.doi.org/10.1007/s11554-018-0774-z.
[45] C.A. Navarro, F.A. Quezada, N. Hitschfeld, R. Vega, B. Bustos, Efficient GPU thread mapping on embedded 2D fractals, Future Gener. Comput. Syst. 113 (2020) 158–169, http://dx.doi.org/10.1016/j.future.2020.07.006.
[46] I. Sa, M. Popović, R. Khanna, Z. Chen, P. Lottes, F. Liebisch, J. Nieto, C. Stachniss, A. Walter, R. Siegwart, WeedMap: A large-scale semantic weed mapping framework using aerial multispectral imaging and deep neural network for precision farming, Remote Sens. 10 (2018) http://dx.doi.org/10.3390/rs10091423.
[47] W. Li, W. Randolph Franklin, S.V.G. de Magalhães, M.V.A. Andrade, GPU-accelerated multiple observer siting, Photogramm. Eng. Remote Sens. 83 (2017) 439–446, http://dx.doi.org/10.14358/PERS.83.6.439.
[48] J. Li, G. Deng, W. Zhang, C. Zhang, F. Wang, Y. Liu, Realization of CUDA-based real-time multi-camera visual SLAM in embedded systems, J. Real-Time Image Process. 17 (2020) 713–727, http://dx.doi.org/10.1007/s11554-019-00924-4.
[49] B. Ruf, J. Mohrs, M. Weinmann, S. Hinz, J. Beyerer, ReS2tAC—UAV-Borne real-time SGM stereo optimized for embedded ARM and CUDA devices, Sensors 21 (3938) (2021) http://dx.doi.org/10.3390/s21113938.
[50] A. Kaczmarczyk, W. Zatorska, Accelerating image fusion algorithms using CUDA on embedded industrial platforms dedicated to UAV and UGV, in: R. Szewczyk, C. Zieliński, M. Kaliczyńska (Eds.), Automation 2019, in: Advances in Intelligent Systems and Computing, Springer International Publishing, Cham, 2020, pp. 697–706, http://dx.doi.org/10.1007/978-3-030-13273-6_65.
[51] R. Schregle, L.O. Grobe, S. Wittkopf, An out-of-core photon mapping approach to daylight coefficients, J. Build. Perform. Simul. 9 (2016) 620–632, http://dx.doi.org/10.1080/19401493.2016.1177116.
[52] J. Kontkanen, E. Tabellion, R.S. Overbeck, Coherent out-of-core point-based global illumination, Comput. Graph. Forum 30 (2011) 1353–1360, http://dx.doi.org/10.1111/j.1467-8659.2011.01995.x.
[53] J. Baert, A. Lagae, P. Dutr, Out-of-core construction of sparse voxel octrees, in: Proceedings of the 5th High-Performance Graphics Conference, HPG '13, Association for Computing Machinery, Anaheim, California, 2013, pp. 27–32, http://dx.doi.org/10.1145/2492045.2492048.
[54] J. Sarton, N. Courilleau, Y. Rémion, L. Lucas, Interactive visualization and on-demand processing of large volume data: A fully GPU-based out-of-core approach, IEEE Trans. Vis. Comput. Graphics 26 (2020) 3008–3021.
[55] M. Zeidan, T. Nazmy, M. Aref, GPU-based out-of-core HLBVH construction, in: Eurographics Symp. Render. - Exp. Ideas Implement, 2015, p. 10, http://dx.doi.org/10.2312/SRE.20151165.
[56] K. Meenrattanakorn, C. Chantrapornchai, Expanding video memory through texture migration for out-of-core shading, in: 2018 22nd International Computer Science and Engineering Conference (ICSEC). Presented At the 2018 22nd International Computer Science and Engineering Conference, ICSEC, 2018, pp. 1–4, http://dx.doi.org/10.1109/ICSEC.2018.8712680.
[57] J. Elseberg, D. Borrmann, A. Nüchter, One billion points in the cloud – an octree for efficient processing of 3D laser scans, ISPRS J. Photogramm. Remote Sens. Terr. 3D Model. 76 (2013) 76–88, http://dx.doi.org/10.1016/j.isprsjprs.2012.10.004.
[58] J. Shen, Y. Wu, M. Okita, F. Ino, Accelerating GPU-based out-of-core stencil computation with on-the-fly compression, 2021, arXiv:2109.05410 Cs.
[59] H. Khaleghzadeh, Z. Zhong, R.R. Manumachu, A. Lastovetsky, Out-of-core implementation for accelerator kernels on heterogeneous clouds, J. Supercomput. 74 (2018) 551–568, http://dx.doi.org/10.1007/s11227-017-2141-4.
[60] J.M. Jurado, L.M. Ortega, F.R. Feito, 3D mapping approach to analyze the evolution of vegetation using multispectral imagery, in: CEIG, 2018, pp. 129–132.
[61] S. Khanal, J. Fulton, S. Shearer, An overview of current and potential applications of thermal remote sensing in precision agriculture, Comput. Electron. Agric. 139 (2017) 22–32, http://dx.doi.org/10.1016/j.compag.2017.05.001.
[62] L. Pádua, J. Vanko, J. Hruška, T. Adão, J.J. Sousa, E. Peres, R. Morais, UAS, sensors, and data processing in agroforestry: a review towards practical applications, Int. J. Remote Sens. 38 (2017) 2349–2391, http://dx.doi.org/10.1080/01431161.2017.1297548.
[63] M. Apel, From 3d geomodelling systems towards 3d geoscience information systems: Data model, query functionality, and data management, Comput. Geosci. 32 (2006) 222–229, http://dx.doi.org/10.1016/j.cageo.2005.06.016.
[64] J. Guo, Z. Cheng, S. Xu, X. Zhang, Realistic procedural plant modeling guided by 3D point cloud, in: ACM SIGGRAPH 2017 Posters, 2017, pp. 1–2.
[65] M. Makowski, T. Hädrich, J. Scheffczyk, D.L. Michels, S. Pirk, W. Pałubicki, Synthetic silviculture: multi-scale modeling of plant ecosystems, ACM Trans. Graph. 38 (2019) 131:1–131:14, http://dx.doi.org/10.1145/3306346.3323039.
[66] R. Feng, Q. Du, X. Li, H. Shen, Robust registration for remote sensing images by combining and localizing feature- and area-based methods, ISPRS J. Photogramm. Remote Sens. 151 (2019) 15–26, http://dx.doi.org/10.1016/j.isprsjprs.2019.03.002.
[67] L. Landrieu, M. Simonovsky, Large-scale point cloud semantic segmentation with superpoint graphs, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4558–4567.
[68] A.L. Ruiz, J.M.J. Rodríguez, C.J.O. Anguita, F.R.F. Higueruela, Multispectral registration, undistortion and tree detection for precision agriculture, 2019.
[69] J. Zhang, X. Zhao, Z. Chen, Z. Lu, A review of deep learning-based semantic segmentation for point cloud, IEEE Access 7 (2019) 179118–179133.
[70] X. Wang, X. Zhu, S. Ying, C. Shen, An accelerated and robust partial registration algorithm for point clouds, IEEE Access 8 (2020) 156504–156518, http://dx.doi.org/10.1109/ACCESS.2020.3019209.
[71] G. Heidsieck, D.de. Oliveira, E. Pacitti, C. Pradal, F. Tardieu, P. Valduriez, Cache-aware scheduling of scientific workflows in a multisite cloud, Future Gener. Comput. Syst. 122 (2021) 172–186, http://dx.doi.org/10.1016/j.future.2021.03.012.
[72] Y. Kirsal, Y.Kirsal. Ever, G.E. Mapp, M. Raza, 3D analytical modelling and iterative solution for high performance computing clusters, IEEE Trans. Cloud Comput. (2021) 1, http://dx.doi.org/10.1109/TCC.2021.3055119.
[73] J.C. Romero, A. Navarro, A. Vilches, A. Rodríguez, F. Corbera, R. Asenjo, Efficient heterogeneous matrix profile on a CPU + high performance FPGA with integrated HBM, Future Gener. Comput. Syst. 125 (2021) 10–23, http://dx.doi.org/10.1016/j.future.2021.06.025.
[74] F. Silla, S. Iserte, C. Reaño, J. Prades, On the benefits of the remote GPU virtualization mechanism: The rCUDA case, Concurr. Comput. Pract. Exp. 29 (2017) e4072, http://dx.doi.org/10.1002/cpe.4072.



Juan M. Jurado received the Ph.D. degree in computer science from University of Jaén, Jaén, Spain, in 2020. He is currently a professor of the Department of Computer Science at the University of Jaén. In 2017, he received the award of the best master's thesis of the University of Jaén. His research area focuses on the generation and processing of 3D models, as well as the fusion of geometric, spatial and spectral variables of real-world environments. The results of his research derive in significant advances in fields such as Remote Sensing, Computer Vision and Computer Graphics through the development of novel methods for the integration of multi-source data, the unsupervised classification of real scenarios and the modeling of the material appearance.