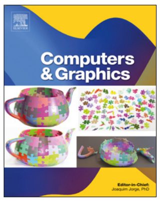
Computers & Graphics 106 (2022) 267–276

Contents lists available at ScienceDirect

Computers & Graphics

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



Special Section on CEIG 2022

Generation of hyperspectral point clouds: Mapping, compression and rendering

Alfonso López [a,*], Juan M. Jurado [b], J. Roberto Jiménez-Pérez [a], Francisco R. Feito [a]

[a] *Department of Computer Science, University of Jaén, Spain*
[b] *Department of Software Engineering, University of Granada, Spain*

A R T I C L E   I N F O

Article history:
Received 12 May 2022
Received in revised form 20 June 2022
Accepted 21 June 2022
Available online 24 June 2022

Keywords:
Hyperspectral
Massively parallel algorithms
Compression
Rendering

A B S T R A C T

Hyperspectral data are being increasingly used for the characterization and understanding of real-world scenarios. In this field, UAV-based sensors bring the opportunity to collect multiple samples from different viewpoints. Thus, light-material interactions of real objects may be observed in outdoor scenarios with a significant spatial resolution (5 cm/pixel). Nevertheless, the generation of hyperspectral 3D data still poses challenges in post-processing due to the high geometric deformation of images. Most of the current solutions use both LiDAR (Light Detection and Ranging) and hyperspectral sensors, which are integrated into the same acquisition system. However, these present several limitations due to errors derived from inertial measurements for data fusion and the spatial resolution according to the LiDAR capabilities. In this work, a method is proposed for the generation of hyperspectral point clouds. Input data are formed by push-broom hyperspectral images and 3D point clouds. On the one hand, the point clouds may be obtained by applying a typical photogrammetric workflow or LiDAR technology. Then, hyperspectral images are geometrically corrected and aligned with the RGB orthomosaic. Accordingly, hyperspectral data are ready to be mapped on the 3D point cloud. This procedure is implemented on the GPU by testing which points are visible for each pixel of the hyperspectral imagery. This work also provides a novel solution to generate, compress and render 3D hyperspectral point clouds, enabling the study of geometry and the hyperspectral response of natural and artificial materials in the real world.

1. Introduction

The realistic modeling of material appearance has been widely studied in Computer Graphics. Accurate and reliable acquisition of either Bidirectional Texture Function (BTF) or even Bidirectional Reflectance Distribution Function (BRDF) is a challenging task, and only a few measurement systems exist. The exploration of hyperspectral imaging opens the possibility to study the material's behavior beyond the human visible range. Recent work is presented for the acquisition of hyperspectral data through a goniophotometer and modeling the BRDF of measured objects [1]. Another method was proposed for the efficient acquisition of spectral BRDFs in outdoor scenarios using a UAV-based (Unmanned Aerial Vehicle) hyperspectral camera [2]. These advances lead us toward the generation of real-world material databases, which are highly 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.

The use of hyperspectral sensors mounted in UAVs brings up new opportunities to explore high-resolution spectral analysis. The light-material interactions can be observed from multiple viewpoints through a wide range of bands to characterize the spectral behavior of surveyed entities in real-world scenarios. The spectral imaging technology was originally used in Earth remote sensing applications, mainly in aerial surveillance. It represented a true revolution in satellite-based remote sensing, allowing first the acquisition of multispectral images, a group of few bands belonging to the visible and near-infrared (VNIR) spectral region. Recent technological developments have enabled the arrival of novel platforms capable of overcoming the major issues associated with both manned aircraft and satellites while simultaneously improving spectral and spatial resolutions. UAV plays an important role in allowing a better understanding of the earth's system phenomena through 3D modeling and hyperspectral spectral characterization.

In the last few years, novel solutions are proposed that rely on the mechanical integration of LiDAR sensors and hyperspectral

* Corresponding author.
E-mail addresses: allopezr@ujaen.es (A. López), jjurado@ugr.es (J.M. Jurado), rjimenez@ujaen.es (J.R. Jiménez-Pérez), ffeito@ujaen.es (F.R. Feito).

https://doi.org/10.1016/j.cag.2022.06.011
0097-8493/© 2022 Elsevier Ltd. All rights reserved.

cameras for the generation of 3D spectral data. Despite the progress made in the fusion of multi-sensor data acquired with drones [3–5], to our knowledge, there is no software solution capable of automatically enriching 3D point clouds with hyperspectral imagery. In addition to the geometric features, the study of hyperspectral data distribution along the reconstructed surfaces helps us to approach non-trivial tasks such as the modeling of complex surface reflectance characteristics, including specularities, interreflections, transparencies, or subsurface scattering under variable and general observation conditions.

In this paper, we propose a pipeline for the generation, compression, and rendering of dense hyperspectral point clouds. The proposed methodology aims to enable hyperspectral analysis in 3D space. Our method is capable to integrate hyperspectral images into 3D models acquired from the real world using LiDAR sensors or photogrammetric techniques. The main contributions of this paper can be summarized as:

1. Hyperspectral image mapping on a high-resolution point cloud.
2. The development of a spatial data structure for hyperspectral orthomosaic compression.
3. The hyperspectral point cloud rendering.

The remainder of this paper is organized as follows: related work is summarized in Section 2, which allows understanding the current state of development. In Section 3, the proposed pipeline for the generation of hyperspectral point clouds on the GPU is explained. Experimental results in terms of performance analysis are described in Section 4. The main conclusions drawn by the development of this work are presented in Section 5.

2. Related work

Hyperspectral data acquisition. 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 of any real-world material is the use of a goniophotometer [6,7]. 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, or other characteristics that allow the sample to be modeled synthetically [8–10]. Nevertheless, these acquisition setups are usually expensive and require to be mounted considering a list of laboratory constraints.

On the other hand, image-based acquisition techniques are a valuable alternative in which the use of a camera together with a suitable procedure to give validity to the results has also extended its use over the last years [11,12]. Moreover, these techniques are particularly suited to data acquisition in real environments. Jurado et al. [2] proposed a novel method for acquiring the spectral reflectance of materials using UAVs equipped with both an RGB camera and a hyperspectral sensor. In terms of the contribution of UAV-based hyperspectral imaging (HSI) for material recognition, some methods have been proposed to classify agricultural materials [13,14] and different types of human objects [15,16]. Despite its radiometric accuracy, the resulting hyperspectral images present a high geometric deformation, which makes its combination with other data sources difficult. This problem was approached by Jurado et al. [17] and proposed a method for geometric alignment between hyperspectral images and high-resolution RGB imagery.

Nowadays, a multidisciplinary research field is focused on hyperspectral data capture for developing applications related to computer graphics, remote sensing and computer vision. In addition, there are other applications of hyperspectral imagery that require data at a microscopic level. Gao and Smith [18] summarized the principles ruling the acquisition of hyperspectral imaging, especially at a microscopic scale in biomedical applications. Pu et al. [19] reviewed hyperspectral imaging techniques for food quality and safety detection.

3D reconstruction of real-world scenarios. The modeling of real-world scenarios has enjoyed great interest by both the industry and academia to improve the visualization and interpretation of surveyed areas and phenomena. Over the past years, the production of high-resolution cameras and recent advances in LiDAR technology allow us to get a detailed 3D model of real-world scenarios. Likewise, the use of UAV platforms makes easier the capture of target entities from multiple viewpoints. Accordingly, a wide variety of natural and urban scenarios can be efficiently reconstructed for multiple purposes, from forest monitoring [3,20,21] to urban planning [22–25]. In particular, a wide variety of sensors has burst onto the market for capturing the three-dimensional features of natural and urban environments. Some examples of these acquisition technologies and methods are Radio Detection and Ranging (RaDAR) [26], Light Detection and Ranging (LiDAR) [27] and Structure-from-Motion (SfM) [28]. The geometry can be extracted following image-based approaches and either terrestrial or aerial LiDAR scans. Regardless of the method employed, a georeferenced point cloud is obtained including additional information such as the RGB color corresponding to the area represented by each of these points.

In general, 3D point clouds are commonly used to represent complex surfaces of the real world. In contrast to triangle meshes, they enable a simpler, denser and more close-to-reality representation [29]. The improvement in the acquisition and processing techniques used in photogrammetry [30] along with the low costs of acquiring detailed and reliable 3D information, transformed photogrammetric-related approaches into a proper alternative to specific sensors such as LiDAR and laser scanners [31]. Thus, as long as the implemented configurations ensure the acquisition of a set of overlapping images, high-density point clouds and 3D meshes can be obtained, opening up a set of possibilities using aerial or ground imagery.

Hyperspectral data fusion. Recent work is presented for the integration of 3D models and multi-source data. The image-based fusion of spectral and geometric features of natural and artificial objects is, in fact, the problem of mapping pixels from these images to 3D points with (x, y, z) coordinates. The aim is to generate a 3D model, which can be automatically enriched with multi-source measured data. This model can be fed with information from different sensors. For instance, Zia et al. [32] developed a method that applies SfM to images from different wavelengths and a 3D registration method to combine band-level models into a single 3D model. Kim et al. [33] described a prototype system composed of a laser sensor and a hyperspectral imager to characterize solid objects. Jurado et al. [34] proposed an out-of-core method that enables the on-site processing and visualization of captured data by using collaborative 3D virtual environments. Parallel and remote computing are addressed for multispectral image mapping and occlusion on huge 3D models. Other works were capable of merging hyperspectral data in 3D models generated from standard RGB cameras or laser scanning, which enabled the 3D geological modeling [35] or 3D mapping of underwater environments [36]. Liu et al. [37] provided an in-depth review of HSI and 3D technologies for plant phenotyping by analyzing the literature from close-range applications to remote sensing.

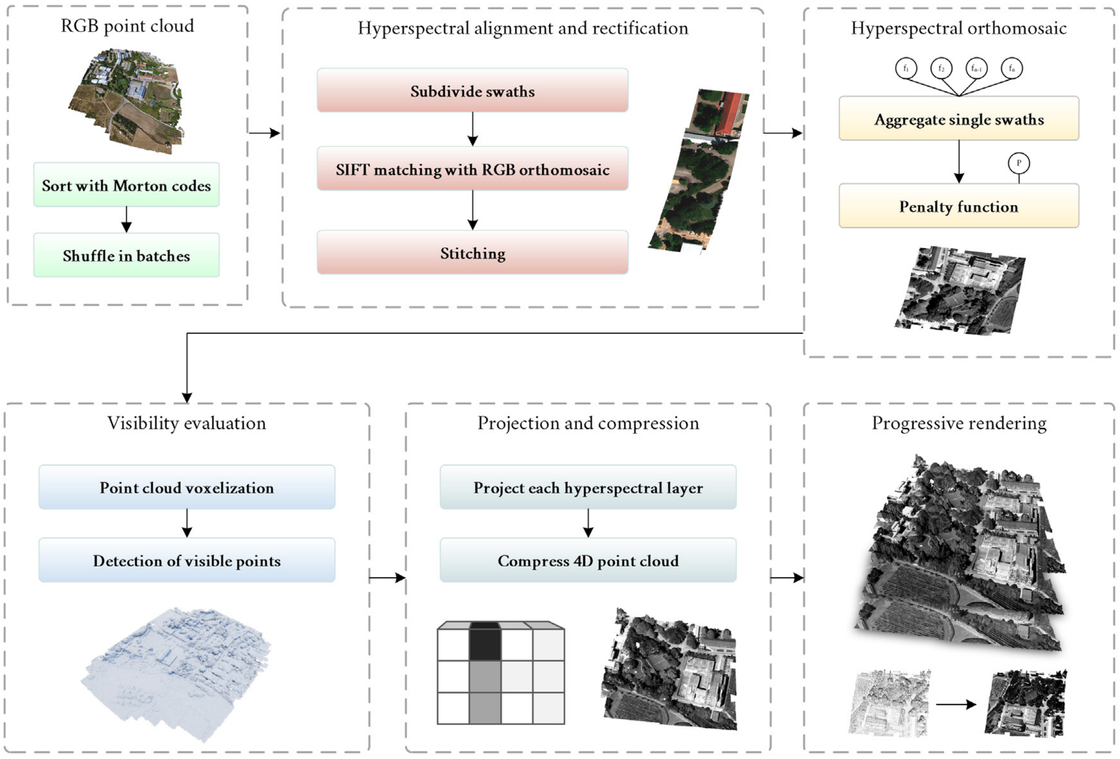


Fig. 1. The overall approach proposed for the generation of 3D hyperspectral point clouds. The starting point cloud is firstly sorted and shuffled to enhance its rendering. Hyperspectral swaths are then corrected and fused to build an orthomosaic. This result is projected into the RGB point cloud, previously voxelized to obtain the visible points. The hypercube is compressed according to a stack-based representation.

As stated above, the processes to deal with such amounts of 3D data and hyperspectral images are computationally demanding. One way of reducing this computational cost is GPU-based parallelization [38,39]. Hyperspectral image processing can take advantage of the computing capabilities of cloud computing or current graphics cards to manage amounts of images [40], perform image analysis by quaternion moments [41] or 3D point clouds. Massive data processing is a trend topic line that arouses the interest of multidisciplinary research. However, there are still many limitations that should be addressed. In this study, our effort is twofold: we aim to accelerate the whole pipeline using compute shaders for hyperspectral image mapping on the point cloud whereas reducing the memory footprint through a spatial data structure aimed at compressing the hypercube.

3. Our method

The proposed approach is based on the efficient generation of hyperspectral point clouds with a low-memory footprint. To this end, the complete procedure is developed on the GPU using OpenGL's compute shaders, both for building and rendering the 3D hyperspectral point cloud. Firstly, the rectification and alignment of hyperspectral images with RGB images are briefly explained according to previous work. Then, hyperspectral images are projected into an 3D RGB point cloud. Throughout the projection procedure, several optimization methods are proposed to address both compression and color aggregation challenges. Once the point cloud is built, its rendering is also approached using GPU-based optimizations concerning visualization and frame rate for large point clouds. Fig. 1 shows the main steps of the proposed pipeline. The 3D reconstruction of the high-resolution point cloud, as well as the acquisition procedure of hyperspectral data is out of the scope of this study. Our aim is focused on data processing and compression for the generation and rendering of dense hyperspectral point clouds.

3.1. Hyperspectral alignment and rectification

The main challenge of post-processing hyperspectral imagery is their high geometrical distortion. Push-broom hyperspectral sensors provide high spectral resolution data, but their scanning acquisition architecture imposes more challenges to creating geometrically accurate mosaics from multiple hyperspectral swaths. This is the first step before data analysis and this task remains a time-consuming and complex processing workflow. In order to provide an efficient solution to correct the image deformation, Jurado et al. [17] proposed a fully automatic method based on an iterative approach to align hyperspectral swaths with an orthorectified RGB mosaic. This method is replicated in our work for the correction of image distortion. In order to make this study self-contained, a brief description is presented as follows.

The main steps for the generation of hyperspectral orthomosaics from the alignment of individual and preprocessed hyperspectral swaths are: (1) image subdivision, (2) feature detection, (3) matching and homography calculation, (4) image transformation, and (5) validation. Firstly, each hyperspectral swath is subdivided into multiple fragments since the geometric deformation varies along the image, being higher on borders. Second, feature detection is developed in order to collect a set of keypoints for each hyperspectral image fragment and the RGB orthophoto through Oriented FAST and Rotated BRIEF (ORB) algorithm [42]. In this phase, we aim to search the best matches in both images based on the calculation of the Hamming distance [43]. Once the matches are obtained, the next step consists of the calculation of the homography. This transformation is estimated in order to fit the position of every pixel of the hyperspectral fragment allowing to achieve a better alignment with the RGB orthophoto used as reference. To consider a valid homography, the resulting error in the alignment must be lower than five pixels. The accuracy of the alignment between the RGB orthophoto mosaic and hyperspectral

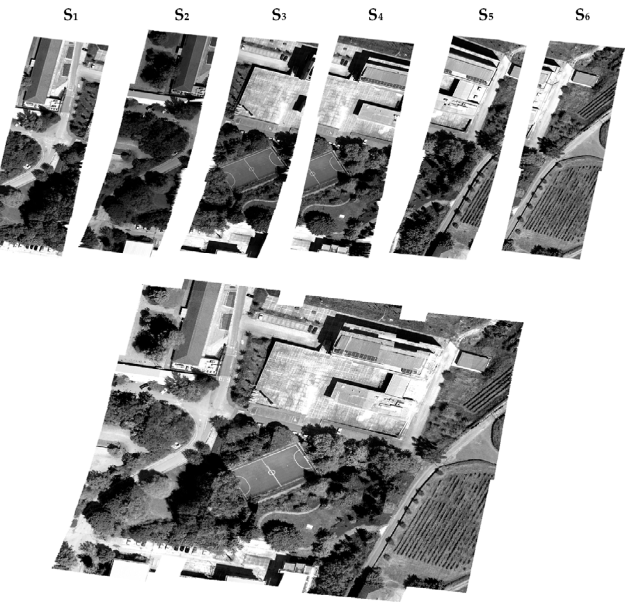


Fig. 2. Hyperspectral swaths for green wavelength and reconstructed orthomosaic using penalty based functions.

swaths is assessed using a validation mask for each image, where control points are represented as unique color combinations. Each control point is manually located in meaningful places such as corners, and key-objects that can easily be recognized in both hyperspectral and RGB imagery. Then, the method is capable of registering the visible markers for each fragment. Thus, the resulting image transformation from the estimated homography is applied and the Euclidean distance is calculated from the position of the projected marker to its corresponding pixel in the RGB orthophoto mosaic. If the error is above the tolerated threshold, the process is iterated by varying the length of the target fragment.

In this study, this approach is followed in order to correct the distortion of UAV hyperspectral swaths. The next step is to generate the hyperspectral orthomosaic and image mapping on the high-resolution point cloud.

3.2. Generation of the hyperspectral orthomosaic

Once hyperspectral swaths are corrected and merged with RGB imagery, both datasets are registered in the Universal Transverse Mercator (UTM) system. Instead of using individual swaths, these can be merged into a single hyperspectral orthomosaic that can be efficiently sampled from the point cloud. Firstly, the offset and size of each swath is known in the UTM system. Therefore, the resulting orthomosaic is initialized combining the 2D axis-aligned bounding box (AABB) of every swath. Accordingly, each hyperspectral pixel can be mapped to a pixel of the final orthomosaic.

Reflectance may vary among hyperspectral swaths, and consequently, values colliding on a pixel can differ from one another. These radiometric differences can be explained by acquisition conditions or errors induced by sensors (e.g., the striping phenomenon) [44], among others. However, we do not intend in this work to solve systematic errors which can be partially removed as a post-processing task. Instead, penalty functions are used as described by Paternain et al. [45] to obtain the most accurate reflectance from multiple swaths.

A local penalty function $P : \mathbb{R}^2 \longrightarrow \mathbb{R}$ computes the penalty of samples x_i with respect to a given aggregator function y as indicated in Eq. (1):

$$P(x_i, y) = (x_i - y)^2 \tag{1}$$

These penalty functions should satisfy the following conditions:

$$\begin{aligned} &P(x_i, y) = 0 && \forall x_i = y \\ &P(x_i, y) > 0 && \forall x_i \neq y \\ &P(x_i, y) \geq P(x_j, y) && |x_i - y| > |x_j - y| \end{aligned} \tag{2}$$

A discriminator function measures the error from each aggregation result to the starting hyperspectral swaths. Then, a dissimilarity function D accumulates the penalties of each set (X) of overlapping samples in a given pixel with respect to a given aggregation function y, as indicated in Eq. (2).

$$D(X, y) = \sum_{x_i \in X} P(x_i, y) \tag{3}$$

The objective of this approach is to select the aggregation operator y that minimizes the expression in Eq. (3), $\min_y D(X, y)$. On the other hand, the aggregation functions are expressed as n-ary functions, $Y : \mathbb{R}^n \longrightarrow \mathbb{R}$, with the arithmetic mean being the most straightforward aggregation operator. Other functions which have been used in this work are the maximum and minimum operators, as well as the geometric and harmonic mean, formally defined as shown in Eqs. (4) to (8).

$$m(X) = \min x_i \tag{4}$$

$$M(X) = \max x_i \tag{5}$$

$$A(X) = \frac{\sum_{i=1}^{n} x_i}{n} \tag{6}$$

$$G(X) = \sqrt[n]{\prod_{i=1}^{n} x_i} \tag{7}$$

$$H(X) = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}} \tag{8}$$

The fusion of hyperspectral swaths is performed on the GPU using OpenGL's compute shaders (Fig. 3). Therefore, the building of the orthomosaic is rapidly solved, obtaining the result depicted in Fig. 2. Note that 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, aggregations are computed per hyperspectral layer and stored in a Shader Storage Buffer Object (SSBO). Some of these aggregations are solved per stage, e.g., *max* and *min* functions, while mean operators are only partially solved as they require a final division. Nevertheless, this calculation can be performed in the following distance-measuring stages. Therefore, images are iterated again to compute local penalty functions (P), whose error is accumulated per aggregation operator, once again, in an SSBO. The last stage is aimed at selecting the operator obtaining the lowest dissimilarity (Fig. 4) to build the orthomosaic with the corresponding aggregation result.

For optimization purposes, hyperspectral images are represented as buffers composed of `uint8_t` values as a rendering-based discretization of the starting reflectance interval. On the other hand, aggregation buffers are scaled to `uint`, as some operators require the sum and product of reflectance samples from different swaths. Also, this procedure is performed on every hyperspectral layer since the selected aggregations may vary among them. Finally, penalty functions are implemented as subroutines, and therefore they can be easily exchanged.

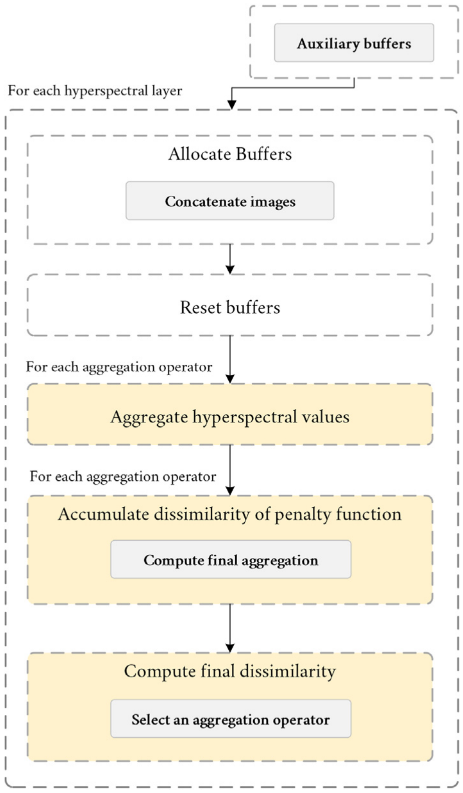

Auxiliary buffers
For each hyperspectral layer
Allocate Buffers
Concatenate images
Reset buffers
For each aggregation operator
Aggregate hyperspectral values
For each aggregation operator
Accumulate dissimilarity of penalty function
Compute final aggregation
Compute final dissimilarity
Select an aggregation operator

Fig. 3. Overview of the aggregation procedure in the GPU.

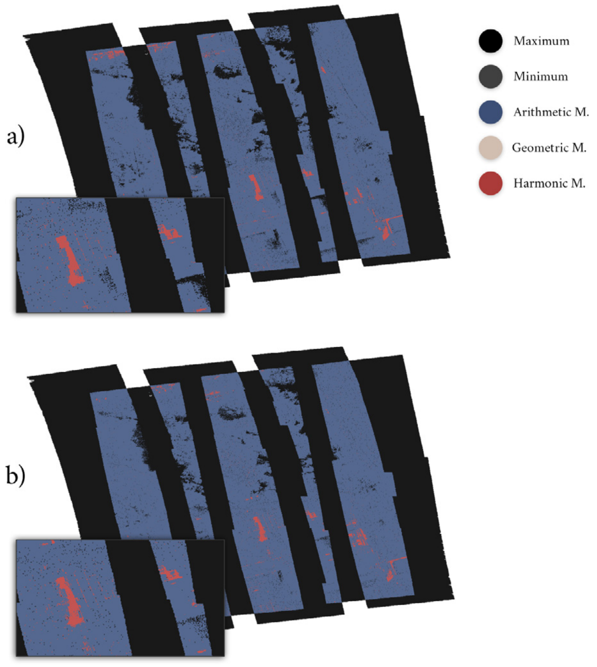


Fig. 4. Point cloud rendered according to the selected aggregation operator for red (a) and green (b) wavelengths. Non-overlapped areas use the first operator (maximum).

3.3. Mapping of hyperspectral orthomosaics

Once hyperspectral orthomosaics are built, they can be projected into a point cloud where both structures are positioned in a common coordinate system, e.g., UTM. In this work, the projection is achieved using an RGB point cloud generated through SfM. Due to the limited resolution of hyperspectral imagery, dense RGB point clouds may obtain interpolated, though inaccurate, hyperspectral samples. Therefore, the point cloud voxelization is approached in 2.5D by generating a heightfield whose dimensionality is defined according to the hyperspectral resolution. The flight was planned to acquire *nadir* images and accordingly, occlusion is intrinsically taken into account with this approach, rather than using oblique directions. Otherwise, voxelization ought to be performed in 3D. Fig. 6 depicts the heightfield of an RGB point cloud rendered with uniform color.

Points are processed in the GPU using both their height and indices within a point cloud batch. The selection of the highest point is implemented as an atomic block using `uint64_t` values. The first 32 bits represent their height (h), whereas the least significant bits store their 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 point while it also carries the index.

The result of this stage is a buffer of 3D points visible from hyperspectral swaths. Therefore, large RGB point clouds are downsampled depending on the hyperspectral resolution. With a resolution of 0.067 m, a point cloud of 330M points was reduced to 17,5M points. From this stage, a hyperspectral point cloud can be straightforwardly computed once hyperspectral layers are transferred as textures that can be sampled in the GPU. The UTM coordinates of each 3D point, originally positioned in a local coordinate system, are calculated according to the point cloud's UTM offset. Hence, the size of the RGB point cloud is significantly reduced as some points may fall outside the hyperspectral acquisition area. This scenario can be either identified by texture coordinates out of range ($u, v < 0$ or $u, v > 1$) or invalid colors sampled from the hyperspectral orthomosaic. Also, the computed orthomosaic is stored as a rectangular texture, and therefore, the background can be marked using the alpha channel.

The main drawback of this approach is the large size of hyperspectral point clouds in environments with low memory capacity, such as the GPU hardware. For the case study of an RGB point cloud of 330M points, the hyperspectral mapping can be represented using 270 layers of 17,5M voxels, i.e., visible points, stored as `uint8_t` (grayscale color). Even with a reduced memory footprint for colors, this approach represents 4.4 GB. Therefore, the following subsection is aimed at handling this drawback.

3.4. Hyperspectral data compression

The compression and reduction of hyperspectral datasets have been widely studied, either for storage optimization [46,47] or enhancing Machine Learning pipelines by reducing the number of features (wavelengths) [48]. The hypercube is characterized by a discrete sampling of a wide wavelength range, either with a constant or varying sampling frequency. Although it shows significant changes throughout the complete spectral range, layers can be significantly compressed when reflectance values are discretized in $[0, 2^8[$, as occurs for rendering applications.

In this work, the hyperspectral point cloud is compressed using a stack-based representation, as proposed by Graciano et al. [49], thus exploiting the similarity of surrounding layers in the Y-axis, i.e., reflectance (Fig. 5). There exist other lossless data compression algorithms for hypercubes, though their rendering and data recovery are not as straightforward as for a stack-based representation. Accordingly, the CSDS-123 [46,47] is a standard algorithm to reduce the size of multispectral and hyperspectral data used in onboard satellites and military drones. Also, it has been previously accelerated in the GPU [50], although is not aimed at providing a readable structure for rendering purposes.

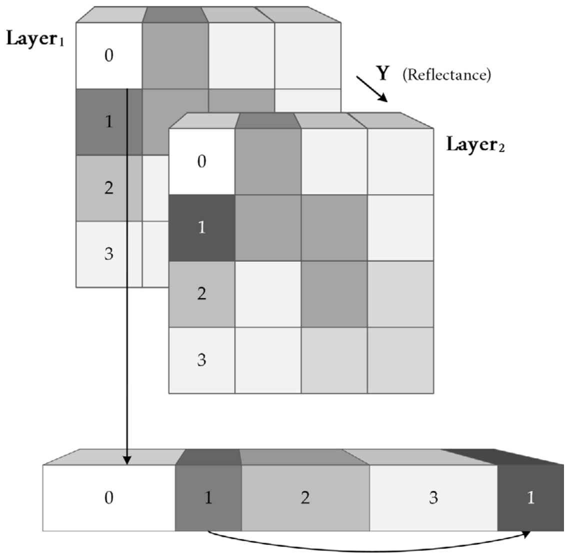


Fig. 5. Stack-based representation for a hypercube of depth 2. For the compact buffer, only the first column is considered.

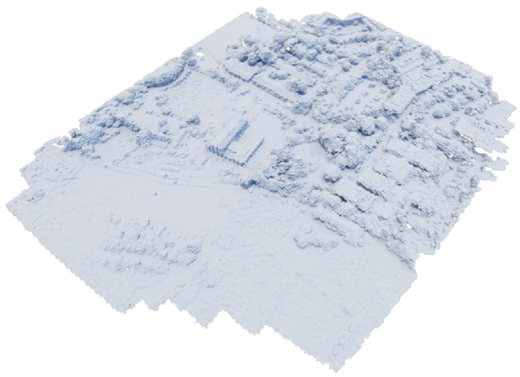


Fig. 6. 2.5D voxelization of an RGB point cloud.

With the stack-based approach, the heightfield is first composed of n voxels, determined by the number of visible points from *nadir* acquisition points. Then, the hypercube is compacted by linking layers storing both the spectral response (1 byte), the number of consecutive layers with such hyperspectral value (2 bytes, as the number of layers may exceed 2^8) and the index of the following layer representation (*uint*). Therefore, the compact layer buffer is approached using uint64_t values split into repeats (16 bits), following index (32 bits) and color (rest of bits, 16 bits).

Despite this data structure allowing the compression of the hypercube, its value encoding presents a larger memory allocation than a voxel grid (8 bits, in comparison with layers of 64 bits). Therefore, a significant compression must be achieved to tackle a higher element-wise memory allocation. It achieves high compression rates both on the studied dataset and publicly available hyperspectral images [51], such as the Pavia Centre (92.13%), Pavia University (92.19%), Kennedy Space Center (95.43%), Salinas Valley (96.39%) or Cuprite (96.41%) datasets. However, large point clouds cannot be stored in a single SSBO. Therefore, they must be split into several batches, whose size depends on the data structure representation, the maximum number of layers and the allocatable memory. Accordingly, current and previous processes must be performed per batch. Although the cited structure achieves a significant compression, the batch dimension is calculated prior to compression and therefore is defined according to a hypercube whose contiguous layers do not overlap their values.

3.5. Point cloud rendering

The visualization of hyperspectral point clouds is relevant for analyzing further details on the surveyed scenarios. However, point clouds may not be appropriate for these purposes due to their gaps when the point density is not high enough. Furthermore, large point clouds harden their traversal and visualization due to low frame rates. Regarding OpenGL rendering, the standard method is to use GL_POINTS primitive. However, it has been shown that compute shaders are capable of significantly enhancing the frame rate by avoiding a fixed rendering pipeline. Instead, the points are projected into an image buffer and rapidly discarded when they lie outside the viewport. Additionally, the ordering of the point buffer also has a considerable impact on the frame rate due to load balancing. Therefore, the widespread sorting based on Morton codes [52] can also be applied to this problem, though it must be shuffled in small batches to ensure that the load is uniformly distributed among the GPU threads. Otherwise, writing operations may be unbalanced on a small set of threads if the buffer is completely ordered, as the outcome most likely depend on spatial ordering.

Based on the work of Schütz et al. [53], the following pipeline is implemented for rendering both RGB and hyperspectral point clouds:

1. A depth buffer with the viewport dimensions is computed using the current point cloud subdivision. Similar to previous processes, the point depth is encoded along with the point index (64 bits). Therefore, the atomic minimum selects the minimum depth. However, this approach leads to a noisy-like rendering for sparse point clouds (Fig. 7). The depth of a pixel is modified according to its surrounding pixels. The minimum depth within an area can be computed using modern OpenGL extensions. Data is exchanged within thread groups whose points fall in the same pixel using shuffleNV and ballotThreadNV operators. Then, shuffleXorNV is used to select the minimum value for each bit, thus computing the minimum depth within a pixel neighborhood.
2. Colors are iteratively accumulated. Note that multiple point cloud subdivisions may aggregate data. Therefore, accumulations are here implemented as atomic additions, whereas (red, green) and (blue, alpha (α)) channels are encoded as 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. The same rendering pipeline is applied to both RGB and hyperspectral point clouds, though the second one solely accumulates data in (blue, alpha) buffer.
3. Accumulated colors are finally transformed by averaging the mean value for each channel. Pixels with $\alpha \leftarrow 0$ obtain the background color.

As a result of the described compression data structure, colors are retrieved by traversing stacks, considering the target layer and the number of repeats per stack layer. Hence, the main concern is the frame rate for the last stacks. Despite the data structure achieving a significant compression, the iterative traversal represents a time-consuming task in OpenGL's shaders. To tackle this problem, images are iteratively constructed in a few frames (see Fig. 8), thus reducing the load of single frames, whereas the user barely notices the iterative process. To this end, a noise buffer ($N \in \mathbf{N}^k, N_i \in [0, n_{points}[$) is built once to determine which

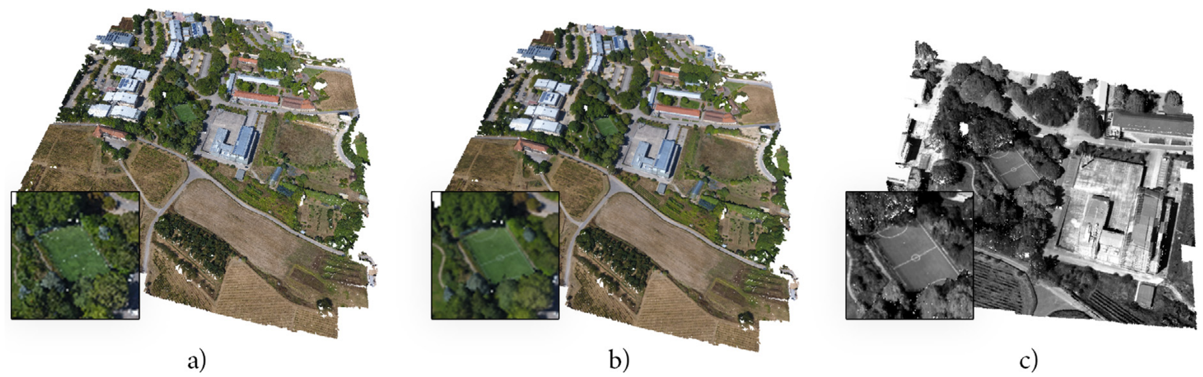


Fig. 7. Rendering of RGB (a, b) and green wavelength (c) point clouds with visualization optimizations (b, c) and naive compute shader approach (a).

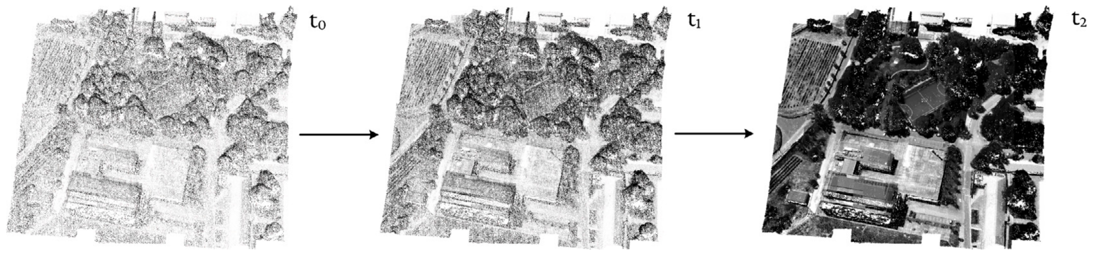


Fig. 8. Iterative rendering of a hyperspectral point cloud during several frames, using 10k points per frame.

points are rendered in every frame. The buffer can be shifted in subsequent frames, allowing to render every visible point and converging on the result of the normal pipeline. The number of frames to achieve such a convergence depends on buffer length, i.e., k.

4. Results

We have evaluated the proposed method through its performance in the GPU. To this end, point clouds and images of different dimensions are used throughout this section. Initially, hyperspectral imagery has dimensions of 640 × 2000, whereas each pixel contains 270 bands. However, the correction process outputs hyperspectral images resized to place swaths within the RGB orthomosaic. The baseline point cloud is generated with two different densities: 150M and 330M, though other subsampled point clouds are generated for the evaluation. Due to the lack of previous work regarding the efficient generation of 3D hyperspectral point clouds, we focus this section 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, GTX 1070 GPU with 8 GB VRAM (Pascal architecture), Windows 10 OS. The proposed methodology is implemented in C++ 17 along with OpenGL (Open Graphics Library). Massively parallel processes are developed in GLSL (OpenGL Shading Language) using general-purpose compute shaders, whereas CPU-based methods are accelerated using the OpenMP (Open Multi-Processing) library for those stages that can be parallelized.

4.1. Data acquisition

This work has been evaluated over imagery obtained from a UAV hexacopter (DJI Matrice 600 Pro (M600)) carrying a Headwall's Nano-Hyperspec sensor. The stability of such a sensor is ensured by a Ronin-MX gimbal, in order to reduce distortions during the push-broom data acquisition process. The used lens presents a focal length of 12 mm, whereas its horizontal field of view is 21.1°. The acquired lines of pixels have a resolution of 640 × 2000 pixels with 270 spectral bands, ranging from 400 to 1000 nm. The sampling interval is 2.2 nm, though it increases to 6 nm at half-maximum. The UAV is also equipped with five global positioning antennas, where only two, mounted in the arms, are aimed at positioning the hyperspectral data. An Inertial Measurement Unit (IMU) is also used to account for yaw, roll and pitch angles. On the other hand, RGB imagery was captured using a DJI Phantom 4 quadcopter and a CMOS camera with a 2.8 mm optical lens and 12.4 MP.

Regarding flight planning, the Universal Ground Control Station (SPH Engineering, Riga, Latvia) was used for the M600 UAV. Hyperspectral swaths were collected with 40% side overlap at an altitude of 100 m, thus acquiring 6 different hyperspectral strips. The second UAV was managed using The DroneDeploy (DroneDeploy, San Francisco, CA, USA), acquiring 324 RGB images at 3.4 cm spatial resolution, with 90% track overlap and 75% side overlap, speed of 6 m/s and altitude of 80 m from take-off position.

The study area is located at the University of Trás-os-Montes e Alto Douro campus, covering 4 hectares. The altitude is 500 m, though it varies up to 30 m in the acquired area. Ground Control Points (GCPs) were uniformly distributed in this area to provide high positional accuracy in contrast to the UAV's GNSS receiver. Therefore, imagery is geolocated using five GCPs consisting of circular targets with a diameter of 0.5 m. Besides GCPs, control points (CP) are 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.

RGB imagery is processed using Pix4DMapper Pro software (Pix4D, Lausanne, Switzerland) to generate high-resolution point clouds through SfM. Firstly, a bundle adjustment is performed based on the images' geographic coordinates acquired by the UAV GNSS receiver and those matching points identified among several images. Therefore, internal and external camera parameters are also estimated in this stage. The process is enhanced by

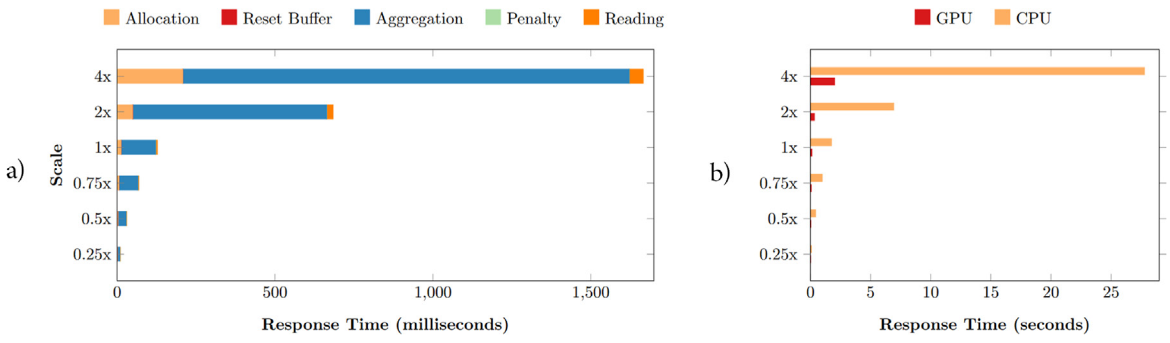


Fig. 9. Response time in milliseconds for the GPU-based approach, stacked per aggregation stage, whereas (b) compares the performance of both CPU and GPU-based methods.

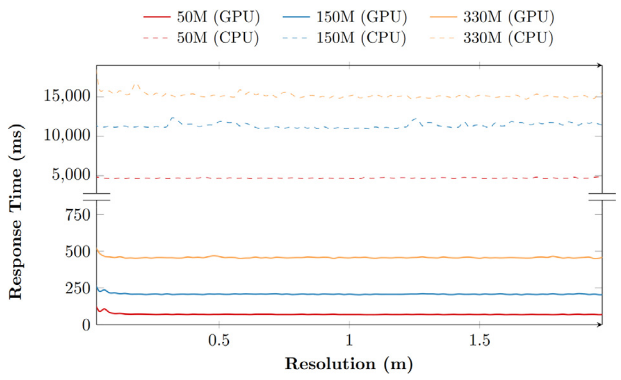


Fig. 10. Performance of 2.5D voxelization for three different point clouds. Dotted lines show the results of the CPU-based approach, while solid lines represent the GPU-accelerated one.

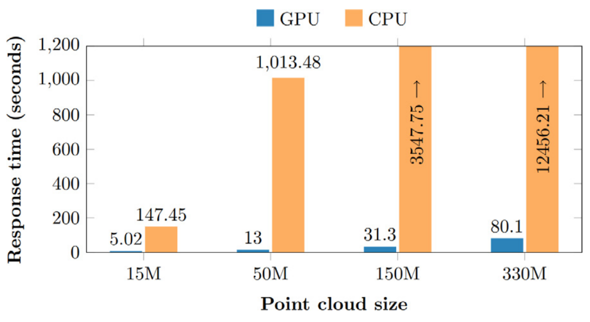


Fig. 11. Response time in seconds to build a hyperspectral point cloud with 270 layers.

marking the cited GCPs. Then, a dense point cloud is generated using the highest density configuration.

4.2. Performance analysis

Individual stages of the proposed methodology are evaluated in this section to show the performance of the overall procedure. Firstly, hyperspectral images are aggregated to compute a hyperspectral orthomosaic. Despite their low resolution, the matching procedure outputs a larger image to fit the RGB orthomosaic. Hence, we have assessed the response time for different image resolutions by scaling it with factors below and above one. As shown in Fig. 9, the response time is low even for upscaled imagery (2x, 4x), though this process must be repeated once per hyperspectral layer. Thus, the aggregation is solved in 130 ms per layer using the starting resolution, whereas the half-size configuration achieves a significant speedup (74.69%). The aggregation stage is the main bottleneck, as it implies multiple operators to be calculated for each pixel. Accordingly, allocation and reading phases also increment their response time due to the buffer length. Regarding the CPU versus GPU comparison, the use of massively parallel algorithms is justified yet for low-dimensionality imagery. For the starting resolution, the speedup of GPU version is 92.54%.

Once the hyperspectral orthomosaic is generated, the point cloud is voxelized in 2.5D to retrieve visible points according to their height within a voxel. Hence, Fig. 10 shows the response time for building heightfields of point clouds with different dimensionality. Also, the resolution varies from 0.02 m to 2 m, including the default hyperspectral resolution (0.067 m). Measurements correspond both to the allocation stage and the procedure itself. Despite the large size of the input point clouds, the voxelization is efficiently solved in the GPU using a single atomic operation for points that lie in a voxel. Thus, the voxelization is solved in 500 ms for the whole interval since the processing task is performed for each point regardless of the resolution. In contrast to the GPU approach, CPU version increases the response time by up to 96.66% (330M).

Projection and compression are then performed considering the visible points and the previously stitched orthomosaic (Fig. 11). Therefore, the response time for building a hyperspectral point cloud while compressing layers is here reported. To this end, we launched this procedure using four point clouds of increasing size, from 15M to 330M. The heightfield is built according to the GSD resolution, whereas the compression rate varies from 70% (15M) to 90% (330M). Note that larger point clouds increase the compression rate as they enhance the overall density and increase the number of points, from where a significant amount presents no variation throughout the hypercube (e.g., shadowed areas). The response time ranges from 5 s for the first point cloud (15M points) to 80 s for 330M points. Hence, the first point cloud is 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 during this procedure, as a result of multiple iterative phases. Accordingly, the speedup ranges from 96.59% (15M) to 99.35% (330M).

Finally, the frame rate for rendering the hyperspectral point cloud is shown in Fig. 12. Two point clouds of dimensionality 150M and 330M are here used, whereas the target layer is iteratively increased for each new frame. Furthermore, two

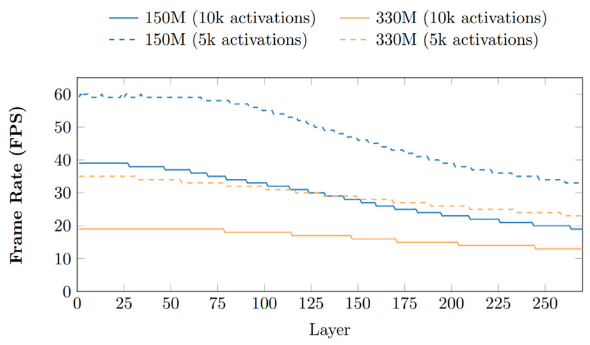


Fig. 12. Frame rate for rendering hyperspectral point clouds with increasing target layer, where activations refer to the number of different points rendered per frame.

configurations are proposed concerning the number of rendered points per frame. As depicted, the main drawback of data compression is the traversal of the data structure to restore the original data, since higher target layers also imply higher delay while traversing the stacks. However, reducing the number of rendered points per frame alleviates the frame load, while the complete scenario is generated in a reduced time. On the other hand, the number of point cloud subdivisions also worsens the performance since it increases the number of compute shader executions. Consequently, point clouds of higher dimensionality present worse performance even for the first layers.

5. Conclusions

In this work, we have proposed a method for the efficient generation of hyperspectral point clouds. To this end, multiple hyperspectral swaths acquired by aerial surveys were firstly fused using penalty and aggregation functions to provide accurate spectral signatures. Also, the hyperspectral point cloud was built from a dense RGB point cloud, considering the hyperspectral resolution. Throughout this process, the point cloud was compressed according to a stack-based structure that takes advantage of the high repeatability of reflectance represented with 8 bits. Finally, the visualization of point clouds was enhanced using modern OpenGL extensions to avoid gaps in sparse point clouds. The rendering pipeline was fused with the compression data structure for the visualization of individual hyperspectral layers. Besides the generation of hyperspectral point clouds, the complete pipeline was implemented in the GPU, thus significantly reducing its response time. Accordingly, the speedup of every stage was shown to be over 90%, even for point clouds with lower size (e.g., 15M points).

In future work, we would like to enhance the rendering pipeline using the compression data structure to reduce the delay derived from traversing the stack in compute shaders. Furthermore, the hypercube compression can be enhanced by aggregating horizontal and vertical features, whereas GPU buffers can be further compacted to reduce the number of point cloud subdivisions and memory allocation of the data structure.

CRediT authorship contribution statement

Alfonso López: Conception and design of study, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Juan M. Jurado:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **J. Roberto Jiménez-Pérez:** Conception and design of study, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Francisco R. Feito:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing – review & editing.

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 result has been partially funded through the research project 1381202-GEU, PYC20-RE-005-UJA, IEG-2021, which are co-financed with the Junta de Andalucía, Spain, Instituto de Estudios Giennenses and the European Union FEDER funds, as well as by the Spanish Ministry of Science, Innovation and Universities via a doctoral grant to the first author (FPU19/00100). All authors approved the version of the manuscript to be published.

References

[1] Dupuy J, Jakob W. An adaptive parameterization for efficient material acquisition and rendering. ACM Trans Graph 2018;37(6):274:1–274:14. http://dx.doi.org/10.1145/3272127.3275059.

[2] Jurado JM, Jiménez-Pérez JR, Pádua L, Feito FR, Sousa JJ. An efficient method for acquisition of spectral BRDFs in real-world scenarios. Comput Graph 2022;102:154–63. http://dx.doi.org/10.1016/j.cag.2021.08.021.

[3] Jurado JM, Ortega L, Cubillas JJ, Feito FR. Multispectral mapping on 3D models and multi-temporal monitoring for individual characterization of olive trees. Remote Sens 2020;12(7):1106. http://dx.doi.org/10.3390/rs12071106.

[4] López A, Jurado JM, Ogayar CJ, Feito FR. An optimized approach for generating dense thermal point clouds from UAV-imagery. ISPRS J Photogramm Remote Sens 2021;182:78–95. http://dx.doi.org/10.1016/j.isprsjprs.2021.09.022.

[5] Adão T, Hruška J, Pádua L, Bessa J, Peres E, Morais R, Sousa JJa. Hyperspectral imaging: A review on UAV-based sensors, data processing and applications for agriculture and forestry. Remote Sens 2017;9(11):1110. http://dx.doi.org/10.3390/rs9111110, Number: 11 Publisher: Multidisciplinary Digital Publishing Institute.

[6] Foo S. A gonioreflectometer for measuring the bidirectional reflectance of material for use in illumination computation. (Ph.D. thesis), Cornell University Graduate School; 2001.

[7] Riviere N, Ceolato R, Hespel L. Multispectral polarized BRDF: Design of a highly resolved reflectometer and development of a data inversion method. Optica Appl 2012;42. http://dx.doi.org/10.5277/oa120101.

[8] Tunwattanapong B, Fyffe G, Graham P, Busch J, Yu X, Ghosh A, Debevec P. Acquiring reflectance and shape from continuous spherical harmonic illumination. ACM Trans Graph 2013;32(4):109:1–109:12. http://dx.doi.org/10.1145/2461912.2461944.

[9] Chen G, Dong Y, Peers P, Zhang J, Tong X. Reflectance scanning: estimating shading frame and BRDF with generalized linear light sources. ACM Trans Graph 2014;33(4):1–11. http://dx.doi.org/10.1145/2601097.2601180.

[10] Ghosh A, Chen T, Peers P, Wilson CA, Debevec P. Estimating specular roughness and anisotropy from second order spherical gradient illumination. Comput Graph Forum 2009;28(4):1161–70. http://dx.doi.org/10.1111/j.1467-8659.2009.01493.x.

[11] Guarnera D, Guarnera G, Ghosh A, Denk C, Glencross M. BRDF representation and acquisition. Comput Graph Forum 2016;35(2):625–50. http://dx.doi.org/10.1111/cgf.12867.

[12] Marschner SR, Westin SH, Lafortune EPF, Torrance KE, Greenberg DP. Image-based BRDF measurement including human skin. In: Lischinski D, Larson GW, editors. Rendering techniques' 99. Vienna: Springer; 1999, p. 131–44. http://dx.doi.org/10.1007/978-3-7091-6809-7_13.

[13] Guarnera GC, Bianco S, Schettini R. Turning a digital camera into an absolute 2D tele-colorimeter. Comput Graph Forum 2019;38(1):73–86. http://dx.doi.org/10.1111/cgf.13393.

[14] Mahesh S, Jayas DS, Paliwal J, White NDG. Hyperspectral imaging to classify and monitor quality of agricultural materials. J Stored Prod Res 2015;61:17–26. http://dx.doi.org/10.1016/j.jspr.2015.01.006.

[15] Martin JA, Gross KC. Enhanced material identification using polarimetric hyperspectral imaging. In: 2014 IEEE applied imagery pattern recognition workshop (AIPR). 2014, p. 1–6. http://dx.doi.org/10.1109/AIPR.2014.7041920.

[16] Chen B, Shi S, Shi S, Sun J, Gong W, Gong W, Yang J, Du L, Guo K, Wang B, Chen B. Hyperspectral lidar point cloud segmentation based on geometric and spectral information. Opt Express 2019;27(17):24043–59. http://dx.doi.org/10.1364/OE.27.024043, Publisher: Optica Publishing Group.

[17] Jurado JM, Pádua L, Hruška J, Feito FR, Sousa JJ. An efficient method for generating UAV-based hyperspectral mosaics using push-broom sensors. IEEE J Sel Top Appl Earth Obs Remote Sens 2021;14:6515–31. http://dx.doi.org/10.1109/JSTARS.2021.3088945, Conference Name: IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing.

[18] Gao L, Smith RT. Optical hyperspectral imaging in microscopy and spectroscopy – a review of data acquisition. J Biophotonics 2015;8(6):441–56. http://dx.doi.org/10.1002/jbio.201400051.

[19] Pu H, Lin L, Sun D-W. Principles of hyperspectral microscope imaging techniques and their applications in food quality and safety detection: A review. Compr Rev Food Sci Food Saf 2019;18(4):853–66. http://dx.doi.org/10.1111/1541-4337.12432.

[20] Nevalainen O, Honkavaara E, Tuominen S, Viljanen N, Hakala T, Yu X, Hyyppä J, Saari H, Pölönen I, Imai NN, Tommaselli AMG. Individual tree detection and classification with UAV-based photogrammetric point clouds and hyperspectral imaging. Remote Sens 2017;9(3):185. http://dx.doi.org/10.3390/rs9030185, Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

[21] Pádua L, Vanko J, Hruška J, Adão T, Sousa JJ, Peres E, Morais R. UAS, sensors, and data processing in agroforestry: a review towards practical applications. Int J Remote Sens 2017;38(8–10):2349–91. http://dx.doi.org/10.1080/01431161.2017.1297548, Publisher: Taylor & Francis.

[22] Can G, Mantegazza D, Abbate G, Chappuis S, Giusti A. Semantic segmentation on Swiss3DCities: A benchmark study on aerial photogrammetric 3D pointcloud dataset. Pattern Recognit Lett 2021;150:108–14. http://dx.doi.org/10.1016/j.patrec.2021.06.004.

[23] Hu Q, Yang B, Khalid S, Xiao W, Trigoni N, Markham A. Towards semantic segmentation of urban-scale 3D point clouds: A dataset, benchmarks and challenges. In: 2021 IEEE/CVF conference on computer vision and pattern recognition (CVPR). 2021, p. 4975–85. http://dx.doi.org/10.1109/CVPR46437.2021.00494.

[24] Wang R, Peethambaran J, Chen D. Lidar point clouds to 3-D urban models: a review. IEEE J Sel Top Appl Earth Obs Remote Sens 2018;11(2):606–27. http://dx.doi.org/10.1109/JSTARS.2017.2781132.

[25] Gobeawan L, Wise DJ, Wong ST, Yee ATK, Lim CW, Su Y. Tree species modelling for digital twin cities. In: Gavrilova ML, Tan CK, editors. transactions on computational science XXXVIII. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer; 2021, p. 17–35. http://dx.doi.org/10.1007/978-3-662-63170-6_2.

[26] Feng Z, Chen Y, Hakala T, Hyyppä J. Range calibration of airborne profiling radar used in forest inventory. In: 2016 IEEE international geoscience and remote sensing symposium (IGARSS). 2016, p. 6672–5. http://dx.doi.org/10.1109/IGARSS.2016.7730742.

[27] Su Y, Guo Q, Xue B, Hu T, Alvarez O, Tao S, Fang J. Spatial distribution of forest aboveground biomass in China: Estimation through combination of spaceborne lidar, optical imagery, and forest inventory data. Remote Sens Environ 2016;173:187–99. http://dx.doi.org/10.1016/j.rse.2015.12.002.

[28] Rahlf J, Breidenbach J, Solberg S, Nsset E, Astrup R. Digital aerial photogrammetry can efficiently support large-area forest inventories in Norway. Int. J. For. Res. 2017;90(5):710–8. http://dx.doi.org/10.1093/forestry/cpx027.

[29] Cao C, Preda M, Zaharia T. 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; 2019, p. 1–9. http://dx.doi.org/10.1145/3329714.3338130.

[30] James MR, Chandler JH, Eltner A, Fraser C, Miller PE, Mills JP, Noble T, Robson S, Lane SN. Guidelines on the use of structure-from-motion photogrammetry in geomorphic research. Earth Surf Process Landf 2019;44(10):2081–4. http://dx.doi.org/10.1002/esp.4637.

[31] Guimarães N, Pádua L, Marques P, Silva N, Peres E, Sousa JJ. Forestry remote sensing from unmanned aerial vehicles: A review focusing on the data, processing and potentialities. Remote Sens 2020;12(6):1046. http://dx.doi.org/10.3390/rs12061046, Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

[32] Zia A, Liang J, Zhou J, Gao Y. 3D reconstruction from hyperspectral images. In: 2015 IEEE winter conference on applications of computer vision. 2015, p. 318–25. http://dx.doi.org/10.1109/WACV.2015.49.

[33] Kim MH, Harvey TA, Kittle DS, Rushmeier H, Dorsey J, Prum RO, Brady DJ. 3D imaging spectroscopy for measuring hyperspectral patterns on solid objects. ACM Trans Graph 2012;31(4). http://dx.doi.org/10.1145/2185520.2185534.

[34] Jurado JM, Padrón EJ, Jiménez JR, Ortega L. An out-of-core method for GPU image mapping on large 3D scenarios of the real world. Future Gener Comput Syst 2022. http://dx.doi.org/10.1016/j.future.2022.03.022.

[35] Nieto JI, Monteiro ST, Viejo D. 3D geological modelling using laser and hyperspectral data. In: 2010 IEEE international geoscience and remote sensing symposium. 2010, p. 4568–71. http://dx.doi.org/10.1109/IGARSS.2010.5651553.

[36] Ferrera M, Arnaubec A, Istenic K, Gracias N, Bajjouk T. Hyperspectral 3D mapping of underwater environments. 2021, arXiv:2110.06571 [cs].

[37] Liu H, Bruning B, Garnett T, Berger B. Hyperspectral imaging and 3D technologies for plant phenotyping: From satellite to close-range sensing. Comput Electron Agric 2020;175:105621. http://dx.doi.org/10.1016/j.compag.2020.105621.

[38] Li QS, Wong FKK, Fung T. Assessing the utility of UAV-borne hyperspectral image and photogrammetry derived 3D data for wetland species distribution quick mapping. In: The international archives of photogrammetry, remote sensing and spatial information sciences, vol. 42. 2017, p. 209, Publisher: Copernicus GmbH.

[39] Zhao B, Liu M, Wu J, Liu X, Liu M, Wu L. Parallel computing for obtaining regional scale rice growth conditions based on WOFOST and satellite images. IEEE Access 2020;8:223675–85. http://dx.doi.org/10.1109/ACCESS.2020.3043003, Conference Name: IEEE Access.

[40] Casella A, De Falco I, Della Cioppa A, Scafuri U, Tarantino E. Exploiting multi-core and GPU hardware to speed up the registration of range images by means of differential evolution. J Parallel Distrib Comput 2019;133:307–18. http://dx.doi.org/10.1016/j.jpdc.2018.07.002.

[41] Salah A, Li K, Hosny KM, Darwish MM, Tian Q. Accelerated CPU–GPUs implementations for quaternion polar harmonic transform of color images. Future Gener Comput Syst 2020;107:368–82. http://dx.doi.org/10.1016/j.future.2020.01.051.

[42] Rublee E, Rabaud V, Konolige K, Bradski G. ORB: An efficient alternative to SIFT or SURF. In: 2011 International conference on computer vision. 2011, p. 2564–71. http://dx.doi.org/10.1109/ICCV.2011.6126544.

[43] Norouzi M, Fleet DJ, Salakhutdinov RR. Hamming distance metric learning. In: Advances in neural information processing systems, vol. 25. Curran Associates, Inc; 2012, p. 1–9.

[44] Pu R. Hyperspectral remote sensing: fundamentals and practices. Boca Raton: CRC Press; 2017, http://dx.doi.org/10.1201/9781315120607.

[45] Paternain D, Fernandez J, Bustince H, Mesiar R, Beliakov G. Construction of image reduction operators using averaging aggregation functions. In: Theme: Aggregation operators, Fuzzy Sets and Systems In: Theme: Aggregation operators, 2015;261:87–111.http://dx.doi.org/10.1016/j.fss.2014.03.008,

[46] Barrios Y, Sánchez AJ, Santos L, Sarmiento R. SHyLoC 2.0: A versatile hardware solution for on-board data and hyperspectral image compression on future space missions. IEEE Access 2020;8:54269–87. http://dx.doi.org/10.1109/ACCESS.2020.2980767, Conference Name: IEEE Access.

[47] Barrios Y, Guerra R, López S, Sarmiento R. Performance assessment of the CCSDS-123 standard for panchromatic video compression on space missions. IEEE Geosci Remote Sens Lett 2022;19:1–5. http://dx.doi.org/10.1109/LGRS.2021.3099032, Conference Name: IEEE Geoscience and Remote Sensing Letters.

[48] Xuan G, Li Q, Shao Y, Shi Y. Early diagnosis and pathogenesis monitoring of wheat powdery mildew caused by blumeria graminis using hyperspectral imaging. Comput Electron Agric 2022;197:106921. http://dx.doi.org/10.1016/j.compag.2022.106921.

[49] Graciano A, Rueda AJ, Pospíšil A, Bittner J, Benes B. QuadStack: An efficient representation and direct rendering of layered datasets. IEEE Trans Vis Comput Graphics 2021;27(9):3733–44. http://dx.doi.org/10.1109/TVCG.2020.2981565, Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[50] Ferraz O, Silva V, Falcao G. Hyperspectral parallel image compression on edge GPUs. Remote Sens 2021;13(6):1077. http://dx.doi.org/10.3390/rs13061077, Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

[51] Hyperspectral remote sensing scenes - grupo de inteligencia computacional (GIC). 2021, URL http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes#Pavia_Centre_and_University.

[52] Lauterbach C, Garland M, Sengupta S, Luebke D, Manocha D. Fast BVH construction on GPUs. Comput Graph Forum 2009;28(2):375–84. http://dx.doi.org/10.1111/j.1467-8659.2009.01377.x.

[53] Schüutz M, Kerbl B, Wimmer M. Rendering point clouds with compute shaders and vertex order optimization. 2021, arXiv:2104.07526 [cs].