remote sensing

MDPI

Article

Real-Time Hyperspectral Data Transmission for UAV-Based Acquisition Platforms

José M. Melián *, Adán Jiménez, María Díaz, Alejandro Morales, Pablo Horstrand, Raúl Guerra, Sebastián López and José F. López

Institute for Applied Microelectronics (IUMA), University of Las Palmas de Gran Canaria (ULPGC), 35001 Las Palmas, Spain; ajimenez@iuma.ulpgc.es (A.J.); mdmartin@iuma.ulpgc.es (M.D.); amorales@iuma.ulpgc.es (A.M.); phorstrand@iuma.ulpgc.es (P.H.); rguerra@iuma.ulpgc.es (R.G.); seblopez@iuma.ulpgc.es (S.L.); lopez@iuma.ulpgc.es (J.F.L.)

* Correspondence: jmelian@iuma.ulpgc.es

Abstract: Hyperspectral sensors that are mounted in unmanned aerial vehicles (UAVs) offer many benefits for different remote sensing applications by combining the capacity of acquiring a high amount of information that allows for distinguishing or identifying different materials, and the flexibility of the UAVs for planning different kind of flying missions. However, further developments are still needed to take advantage of the combination of these technologies for applications that require a supervised or semi-supervised process, such as defense, surveillance, or search and rescue missions. The main reason is that, in these scenarios, the acquired data typically need to be rapidly transferred to a ground station where it can be processed and/or visualized in real-time by an operator for taking decisions on the fly. This is a very challenging task due to the high acquisition data rate of the hyperspectral sensors and the limited transmission bandwidth.

This research focuses on providing a working solution to the described problem by rapidly compressing the acquired hyperspectral data prior to its transmission to the ground station. It has been tested using two different NVIDIA boards as on-board computers, the Jetson Xavier NX and the Jetson Nano. The *Lossy Compression Algorithm for Hyperspectral Image Systems* (HyperLCA) has been used for compressing the acquired data. The entire process, including the data compression and transmission, has been optimized and parallelized at different levels, while also using the Low Power Graphics Processing Units (LPGPUs) embedded in the Jetson boards. Finally, several tests have been carried out to evaluate the overall performance of the proposed design. The obtained results demonstrate the achievement of real-time performance when using the Jetson Xavier NX for all the configurations that could potentially be used during a real mission. However, when using the Jetson Nano, real-time performance has only been achieved when using the less restrictive configurations, which leaves room for further improvements and optimizations in order to reduce the computational burden of the overall design and increase its efficiency.

Keywords: real-time compression; on-board compression; real-time transmission; hyperspectral images; UAVs

Citation: Melián, J.M.; Jiménez, A.; Díaz, M.; Morales, A.; Horstrand, P.; Guerra, R.; López, S.; López, J.F. Real-Time Hyperspectral Data Transmission for UAV-Based Acquisition Platforms. *Remote Sens.* **2021**, *13*, 850. https://doi.org/10.3390/rs13050850

Academic Editor: Vladimir Lukin, Benoit Vozel, Joan Serra-Sagristà

Received: 22 January 2021
Accepted: 22 February 2021
Published: 25 February 2021

1. Introduction

During the last years, there has been a great interest in two very different technologies that have demonstrated to be complementary, unmanned aerial vehicles (UAVs) and hyperspectral imaging. In this way, the combination of both allows to extend the range of applications in which they have been independently used up to now. In one hand, UAVs have been employed in a great variety of uses due to the advantages that these vehicles have over traditional platforms, such as satellites or manned aircrafts. Among these advantages are the high spatial and spectral resolutions that can be reached in images obtained from UAVs, the revisit time and a relative low economical cost. The monitoring of road-traffic [1–4], searching and rescuing operations [5–7], or security and

surveillance [8,9] are some examples of civil applications that have adopted UAVs-based technologies. On the other hand, hyperspectral imaging technology has also gained an important popularity increment due to the high amount of information that this kind of images is able to produce, which is specially valuable for applications that need to detect or distinguish different kind of materials that may look like very similar in a traditional digital image. Accordingly, it can be inferred that the combination of these two technologies by mounting hyperspectral cameras onto drones would bring important advances in fields like real-time target detection and tracking [10].

Despite the benefits that can be obtained combining UAVs and hyperspectral technology, it is necessary to take into account that both also have important limitations. Referring to UAVs, they have a flight autonomy that is limited by the capacity of their batteries, a restriction that gets worse as the weight carried by the drone increases with additional elements, such as cameras, gimbal, on-board PC, or sensors. In reference to the hyperspectral technology, the analysis of the large quantity of data provided by this kind of cameras to extract conclusions for the targeted application implies a high computational cost and makes it difficult to achieve real-time performance. So far, two main approaches have been followed in the state-of-the-art of hyperspectral UAVs-based applications:

1. **Storing the acquired data into a non-volatile memory and off-board processing it once back in the laboratory.** The main advantage of this approach is its simplicity and the possibility of processing the data without any limitation in terms of computational resources. However, it is not viable for applications that require a real-time response, and it does not allow the user to visualize the acquired data during the mission. For instance, following this strategy, the drone could fly over a region, scan it with the hyperspectral camera, and store all of the acquired data into a solid state drive (SSD), so, once the drone lands, it is powered off and the disk is removed for extracting the data and processing it [11–13]. However, if, by any reason, the acquisition parameters were not correctly configured or the data was not captured as expected, this will not be detected until finishing the mission and extracting the data for its analysis.
2. **On-board processing the acquired hyperspectral data.** This is a more challenging approach due to the high data acquisition rate of the hyperspectral sensors as well as the limited number of computational resources available on-board. Its main advantage is that the acquired data could be potentially analysed in real-time or near real-time and, so, decisions could be taken on the fly according to the obtained results. However, the data analysis methods that can be on-board executed are very limited [10,14,15]. Additionally, since, while following this approach, the data are fully processed on-board and never transmitted to the ground station during the mission, it cannot be used for any supervised or semi-supervised application in which an operator is required for visualizing the acquired data or its results and taking decisions according to it. Some examples of this kind of applications are surveillance and search and rescue missions.

This work focuses on providing an alternative solution for a UAV-based hyperspectral acquisition platform to achieve greater flexibility and adaptability to a larger number of applications. Concretely, the aforementioned platform consists of a visible and near-infrared (VNIR) camera, the Specim FX10 pushbroom scanner, [16] mounted on a DJI Matrice 600 [17], which also carries a NVIDIA Jetson board as on-board computer. A detailed description of this platform can be found in [10]. Two different Jetson boards have been used in the experiments and it can be set in the platform as on-board computer, the Jetson Nano [18] and the Jetson Xavier NX [19]. They are both pretty similar in shape and weight, although the Jetson Xavier NX has a higher computational capability as well as a higher price.

The main goal of this work is to provide a working solution that allows us to rapidly download the acquired hyperspectral data to the ground station. Once the data are in the ground station, they could be processed in a low latency manner and both the hyperspectral data or the results obtained after the analysis could be visualized by an operator in real-time.

This is advantageous for many applications that involve a supervised or semi-supervised procedure, such as defense, surveillance, or search and rescue missions. Currently, these kind of applications are mostly carried out by using RGB first person view cameras [1,2], but they do not take advantage of the possibilities that are offerred by the extra information provided by the hyperspectral sensors. Figure 1 graphically describes the desired targeted solution. To achieve this goal, it is assumed that a Wireless Local Area Network (WLAN) based on the 802.11 standard, which was established by the Institute of Electrical and Electronics Engineers (IEEE), will be available during the flight, and that both the on-board computer and the ground station will be connected to it to use it as data downlink. It will be considered to be real-time transmissions those situations in which the acquisition frame rate and the transmission frame rate reach the same value, although there may be some latency between the capturing process and the transmission one.

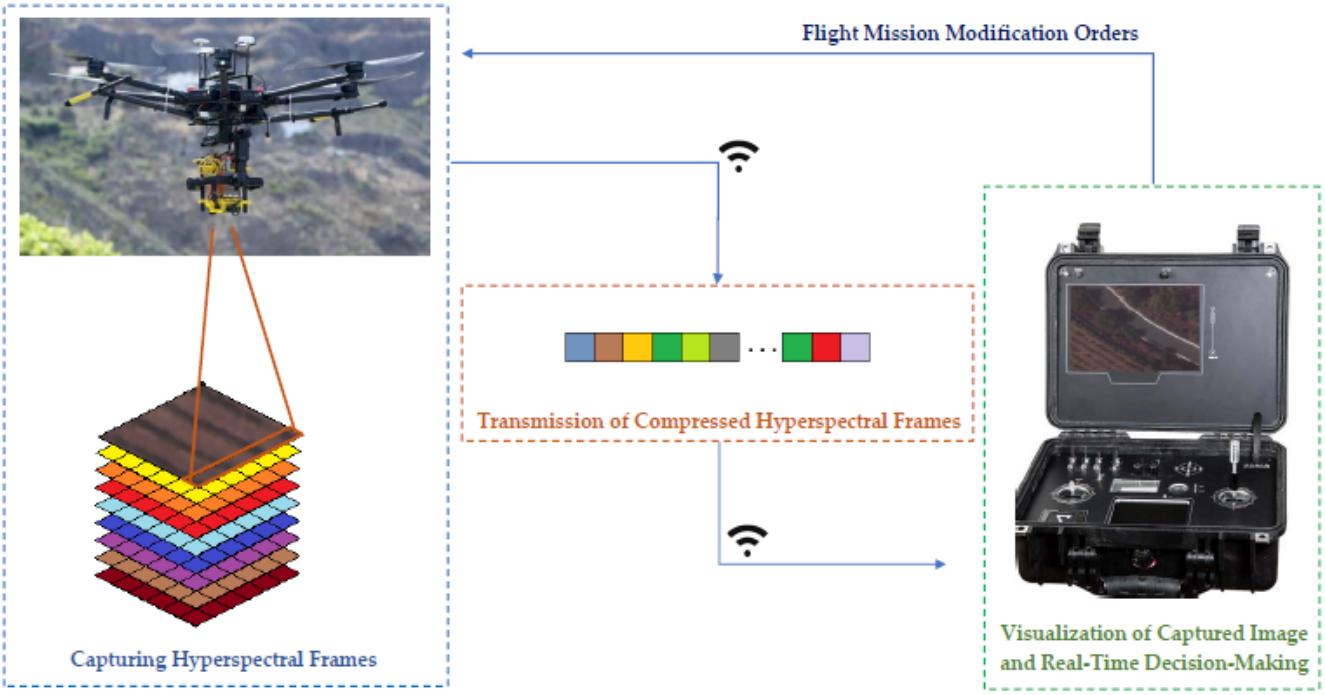

Figure 1. Overall description of the targeted design purpose.

According to our experience in the flight missions carried up to now, the frame rate of the Specim FX10 camera mounted in our flying platform is usually set to a value between 100 and 200 frames per second (FPS) and, so, these are the targeted scenarios to be tested in this work. The Specim FX10 camera produces hyperspectral frames of 1024 pixels with up to 224 spectral bands each, storing them using 12 or 16 bits per pixel per band (bpppb). This means that the acquisition data rate goes from 32.8 MB/s (100 FPS and 12 bpppb) to 87.5 MB/s (200 FPS and 16 bpppb). These values suggest the necessity of carrying out the hyperspectral data compression before its transmission to the ground station in order to achieve real-time performance. For that purpose, the *Lossy Compression Algorithm for Hyperspectral Image Systems* (HyperLCA) has been chosen. This specific compressor has been selected due to the fact that it offers several advantages with respect to other state-of-the art solutions. One one side, it presents a low computational complexity and high level of parallelism in comparison with other transform-based compression approaches. On the other side, the HyperLCA compressor is able to achieve high compression ratios while preserving the relevant information for the ulterior hyperspectral applications. Additionally, the HyperLCA compressor independently compresses blocks of hyperspectral pixels without any spatial alignment required, which makes it especially advantageous for compressing hyperspectral data that were captured by pushbroom or whiskbroom sensors

and providing a natural error-resilience behaviour. All of these advantages were deeply analyzed in [20].

The data compression process reduces the amount of data to be downloaded to the ground station, which makes the achievement of a real-time data transmission viable. However, it is also necessary to achieve a real-time performance in the compression process. For doing so, the HyperLCA compressor has been parallelized taking advantage of the available LPGPU included in the Jetson boards as it was done in [21]. However, multiple additional optimizations have been developed in this work with respect to the one presented in [21] to make the entire system work as expected, encompassing the capturing process, the on-board data storing, the hyperspectral data compression and the compressed data transmission.

The document is structured, as follows. Section 2 contains the description of the algorithm used in this work for the compression of the hyperspectral data. Section 3 displays the information of the on-board computers that have been tested in this work. The proposed design for carrying out the real-time compression and transmission of the acquired hyperspectral data is fully described in Section 4.1. Section 4.2 contains the data and resources that are used for the experiments and Section 5 shows the description of the experiments and obtained results. Finally, the obtained conclusions are summarized in Section 7.

2. Description of the HyperLCA Algorithm

The HyperLCA compressor is used in this work to on-board compress the hyperspectral data, reducing its volume before being transmitted to the ground station, as it was previously specified. This compressor is a lossy transform-based algorithm that presents several advantages that satisfy the necessities of this particular application very well. The most relevant are [20]:

- It is able to achieve high compression ratios with reasonable good rate-distortion ratios and keeping the relevant information for the ulterior hyperspectral images analysis. This allows considerably reducing the data volume to be transmitted without decreasing the quality of the results obtained by analyzing the decompressed data in the ground station.
- It has a reasonable low computational cost and it is highly parallelizable. This allows for taking advantage of the LPGPU available on-board to speed up the compression process for achieving a real-time compression.
- It permits independently compressing each hyperspectral frame as it is captured. This makes possible to compress and transmit the acquired frames in a pipelined way, enabling the entire system to continuously operate in an streaming fashion, with just some latency between the acquired frame and the frame received in the ground station.
- It permits setting the minimum desired compression ratio in advance, making it possible to fix the maximum data rate to be transmitted prior to the mission, thus ensuring that it will not saturate the donwlink.

The compression process within the HyperLCA algorithm consists of three main compression stages, which are a spectral transform, a preprocessing stage, and the entropy coding stage. In this work, each compression stage independently processes one single hyperspectral frame at a time. The spectral transform sequentially selects the most different pixels of the hyperspectral frame using orthogonal projection techniques. The set of selected pixels is then used for projecting this frame, obtaining a spectral decorrelated and compressed version of the data. The preprocessing stage is executed after the spectral transform for adapting the output data for being entropy coded in a more efficient way. Finally, the entropy coding stage manages the codification of the extracted vectors using a Golomb–Rice coding strategy. In addition to these three compression stages, there is one extra initialization stage, which carries out the operations that are used for initializing the compression process according to the introduced parameters. Figure 2 graphically

shows these compression stages, as well as the data that the HyperLCA compressor share between them.

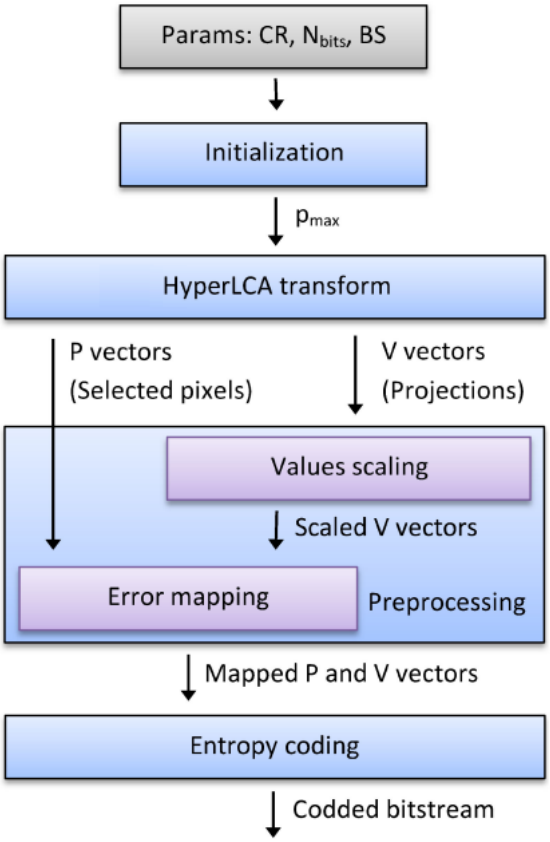

Figure 2. Flowchart of the HyperLCA compressor.

2.1. HyperLCA Input Parameters

The HyperLCA algorithm needs three input parameters in order to be configured:

1. **Minimum desired compression ratio (CR)**, defined as the relation between the number of bits in the real image and the number of bits of the compressed data.
2. **Block size (BS)**, which indicates the number of hyperspectral pixels in a single hyperspectral frame.
3. **Number of bits used for scaling the projection vectors (N_{bits})**. This value determines the precision and dynamic range to be used for representing the values of the V vectors.

2.2. HyperLCA Initialization

Once the compressor has been correctly configured, its initialization stage is carried out. This initialization stage consists on determining the number of pixel vectors and projection vectors ($p_{\max}$) to be extracted for each hyperspectral frame, as shown in Equation (1), where DR refers to the number of bits per pixel per band of the hyperspectral image to be compressed and N_{bands} refers to the number of spectral bands. The extracted pixel vectors are referred as P and the projection vectors are referred as V in the rest of this manuscript.

$$p_{\max} \leq \frac{DR \cdot N_{bands} \cdot BS}{CR \cdot (DR \cdot N_{bands} + N_{\text{bits}} \cdot BS)} \quad (1)$$

The $p_{\max}$ value is used as an input of the HyperLCA transform, which is the most relevant part of the HyperLCA compression process. The compression that is achieved within this process directly depends on the number of selected pixels, $p_{\max}$. Selecting more pixels provides better decompressed images, but lower compression ratios.

2.3. HyperLCA Transform

For each hyperspectral frame, the HyperLCA Transform calculates the average pixel, also called centroid. Subsequently, the centroid pixel is used by the HyperLCA Transform to select the first pixel as the most different from the average. This is an unmixing like strategy used in the HyperLCA compressor for selecting the most different pixels of the data set to perfectly preserve them through the compression–decompression process. This results in being very useful for many remote sensing applications, like anomaly detection, spectral unmixing, or even hyperspectral and multispectral image fusion, as demonstrated in [22–24].

The HyperLCA Transform provides most of the compression ratio that was obtained by the HyperLCA compressor and also most of its flexibility and advantages. Additionally, it is the only lossy part of the HyperLCA compression process. Algorithm 1 describes, in detail, the process followed by the HyperLCA Transform for a single hyperspectral frame. The pseudocode assumes that the hyperspectral frame is stored as a matrix, M, with the hyperspectral pixels being placed in columns. The first step of the HyperLCA Transform consists on calculating the centroid pixel, c, and subtracting it to each pixel of M, obtaining the centralized frame, M_c. The transform modifies the M_c values in each iteration.

Figure 3 graphically describes the overall process that is followed by the HyperLCA Transform for compressing a single hyperspectral frame. As shown in this figure, the process that is followed by the HyperLCA Transform mainly consists of three steps that are sequentially repeated. First, the brightest pixel in M_c, which is the pixel with more remaining information, p_i, is selected (lines 2 to 7 of Algorithm 1). After doing so, the vector v_i is calculated as the projection of the centralized frame, M_c, in the direction spanned by p_i (lines 8 to 10 of Algorithm 1). Finally, the information of the frame that can be represented with the extracted p_i and v_i vectors is subtracted from M_c, as shown in line 11 of Algorithm 1.

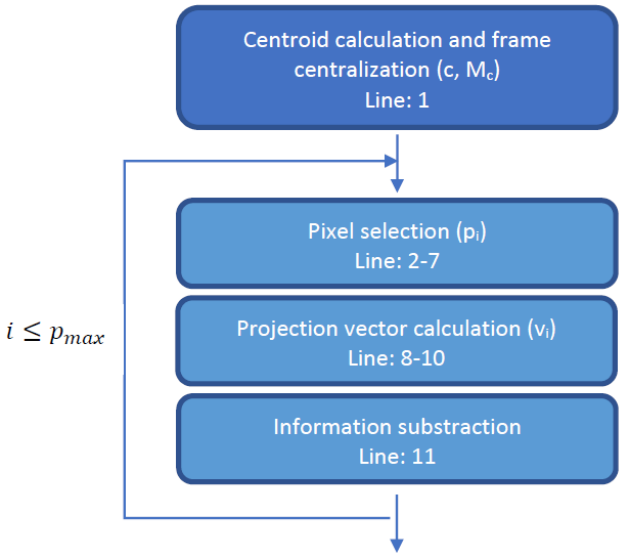

Figure 3. Flowchart of the HyperLCA Transform

Accordingly, M_c contains the information that is not representable with the already selected pixels, P, and V vectors. Hence, the values of M_c in a particular iteration, i, would be the information lost in the compression–decompression process if no more pixels p_i and v_i vectors are extracted.

Algorithm 1 HyperLCA transform.

Inputs:

$M = [r_1, ..., r_{N_p}]$, $p_{\max}$

Outputs:

c, $P = [p_1, ..., p_{p_{\max}}]$, $V = [v_1, ..., v_{p_{\max}}]$

Declarations:

c; {Centroid pixel.}

$P = [p_1, ..., p_{p_{\max}}]$; {Extracted pixels.}

$V = [v_1, ..., v_{p_{\max}}]$; {Projected image vectors.}

$M_c = [x_1, ..., x_{N_p}]$; {Centralized version of M.}

Algorithm:

{Centroid calculation and frame centralization (c, M_c)}
for $i = 1$ **to** $p_{\max}$ **do**
    for $j = 1$ **to** N_p **do**
        $b_j = x_j^t \cdot x_j$
    end for
    $j_{\max} = \arg\max(b_j)$
    $p_i = r_{j_{\max}}$
    $q = x_{j_{\max}}$
    $u = x_{j_{\max}} / ((x_{j_{\max}})^t \cdot x_{j_{\max}})$
    $v_i = u^t \cdot M_c$
    $M_c = M_c - v_i \cdot q$
end for

2.4. HyperLCA Preprocessing

This stage is crucial in the HyperLCA compression algorithm to adapt the HyperLCA Transform output data for being entropy coded in a more efficient way. This compression stage encompasses two different parts.

2.4.1. Scaling V Vectors

After executing the HyperLCA Transform, the resulting V vectors contain the projection of the frame pixels into the space spanned by the different orthogonal projection vector u_i in each iteration. This results in floating point values of V vector elements between -1 and 1. These values need to be represented using integer data type for their codification. Hence, vectors V can be easily scaled in order to fully exploit the dynamic range available according to the N_{bits} used for representing these vectors and avoid loosing too much precision in the conversion, as shown Equation (2). After doing so, the scaled V vectors are rounded to the closest integer values.

$$v_{j_{\text{scaled}}} = (v_j + 1) \cdot (2^{N_{\text{bits}}-1} - 1) \quad (2)$$

2.4.2. Error Mapping

The entropy coding stage takes advantage of the redundancies within the data to assign the shortest word length to the most common values in order to achieve higher compression ratios. In order to facilitate the effectiveness of this stage, the output vectors of the HyperLCA Transform, after the preprocessing of V vectors, are independently lossless processed to represent their values using only positive integer values closer to zero than the original ones, using the same dynamic range. To do this, the HyperLCA algorithm makes use of the prediction error mapper described in the Consultative Committee for Space Data Systems (CCSDS) that are recommended standard for lossless multispectral and hyperspectral image compression [25].

2.5. HyperLCA Entropy Coding and Bitstream Generation

The last stage of the HyperLCA compressor corresponds to a lossless entropy coding strategy. The HyperLCA algorithm makes use of the Golomb–Rice algorithm [26] where each single output vector is independently coded.

Finally, the outputs of the aforementioned compression stages are packaged in the order that they are produced, generating the compressed bitstream. The first part of the bitstream consists of a header that includes all of the necessary information to correctly decompress the data.

The detailed description of the coding methodology followed by the HyperLCA compressor as well as the exact structure of the compressed bitstream can be found in [20].

3. On-Board Computers

The on-board computer is one of the key elements of the entire acquisition platform, as described in [10]. It is in charge of managing the overall mission, controlling the UAV actions and flying parameters (direction, speed, altitude, etc.), as well as the data acquisition controlling all of the sensors available on-board, including the Specim FX10 hyperspectral camera. For doing so, two different Nvidia Jetson boards have been separately tested in this work, the Jetson Nano [18] and the Jetson Xavier NX [19]. These two boards present very good computational capabilities as well as many connection interfaces that facilitate the integration of all the necessary devices into the system, while, at the same time, are characterized by a high computational capability in relation to their reduced size, weight and power consumption. Additionally, their physical characteristics are generally better than the ones that are presented by the Jetson TK1, the board originally used in the first version of our UAV capturing platform, described in [10]. This is a remarkable point when taking into account the restrictions in terms of available space and load weight in the drone. Table 1 summarizes the main characteristics of the Jetson TK1, whereas, in Tables 2 and 3, the main characteristics of the two Jetson boards tested in this work are summarized.

In addition to the management of the entire mission and acquisition processes, the on-board computer is also used in this work for carrying out the necessary operations to compress and transmit in real-time the acquired hyperspectral data to the ground station. For doing so, the compression process, which is the most computational demanding one, has been accelerated while taking advantage of the LPGPUs integrated into these Jetson boards.

Table 1. Technical specifications of the NVIDIA Jetson TK1 development board used in the original flight platform.

Board Model	Jetson TK1
LPGPU	GPU NVIDIA Kepler with 192 CUDA cores (uptto 326 GFLOPS)(Model GK20)
CPU	NVIDIA 2.43 GH < ARM quad-core CPU with Cortex A15 battery saving shadow core
Memory	16GB fast eMMC 4.51 (routed to SDMMC4)
Weight	500 g
Dimensions	127 × 127 × 25.4 mm
Power consumption	1W minimum

By default, both the Jetson Nano and Jetson Xavier NX are prepared for running the operating system in a Secure Digital card (SD). However, this presents limitations in terms of memory speed as well as in the maximum amount of data that can be stored. To overcome this issue, an external SSD that is connected through an USB3.0 port has been used in the Jetson Nano. The Jetson Xavier NX is already prepared to integrate a Non-Volatile Memory Express (NVMe) 2.0 SSD keeping the system more compact and efficient. However, this memory was not yet available during this work and, hence, all of the experiments executed in the Jetson Xavier NX were carried out using just the SD card memory.

Additionally, a Wifi antenna is required. to be able to connect to the WLAN to transmit the compressed hyperspectral data to the ground station. While the Jetson Xavier NX already integrates one, the Jetson Nano does not. Hence, an external USB2.0 TP-Link TL-WN722N Wifi antenna [27] has been included in the experiments that were carried out using the Jetson Nano.

Table 2. Technical specifications of the NVIDIA Jetson Nano and NVIDIA Jetson Xavier NX.

Board Model	Jetson Nano
LPGPU	GPU NVIDIA Maxwell with 128 CUDA cores (Model GM200)
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Weight	140g
Dimensions	99.06x78.74x27.94 mm
Power consumption	5W minimum
Board model	Jetson Xavier NX
LPGPU	GPU NVIDIA Volta with 384 CUDA cores and 48 Tensor Cores (Model GV100)
CPU	6-core NVIDIA Carmel ARM v8.2 64-bit CPU + 6MB L2 + 4MB L3
Memory	8 GB 128-bit LPDDR4x 51.2 GB/s
Weight	180g
Dimensions	99.06x78.74x27.94 mm
Power consumption	10W minimum

Table 3. Technical specifications of the LPGPU in NVIDIA Jetson Nano and NVIDIA Jetson Xavier NX.

Technical Specifications	GM200	GV100
LPGPU	Jetson Nano	Jetson Xavier NX
Number of CUDA cores	640	384
Number of Tensor Cores	-	48
Number of Streaming multiprocessors (SM)	5	6
Compute Capability	5.2	7.0
Warp Size	32	32
Maximum blocks per SM	32	32
Maximum warps per SM	64	64
Maximum threads per SM	2048	2048
32 bit registers per SM	64 KB	64 KB
Maximum shared memory per SM	96 KB	96 KB

4. Materials and Methods

4.1. Proposed Methodology

As already mentioned, the goal of this work is to provide a working solution that allows for rapidly downloading the acquired hyperspectral data to the ground station, where it could be visualized in real-time. It is assumed that a WLAN will be available during the flight, and that both the on-board computer and the ground station will be connected to it to use it as data downlink. Additionally, it is necessary to independently compress each acquired hyperspectral frame in real-time, so that it can be rapidly transferred to the ground station without saturating the donwlink. A previous work was already published, where the viability of carrying out the real-time compression of the hyperspectral data using the HyperLCA algorithm and some NVIDIA Jetson boards was tested [21]. In that work, it was assumed that the hyperspectral data that were collected by the Specim FX10 camera would be directly placed in Random-access memory (RAM), and that the compression process would directly read from it. Figure 4 graphically shows the workflow that was proposed in that work.

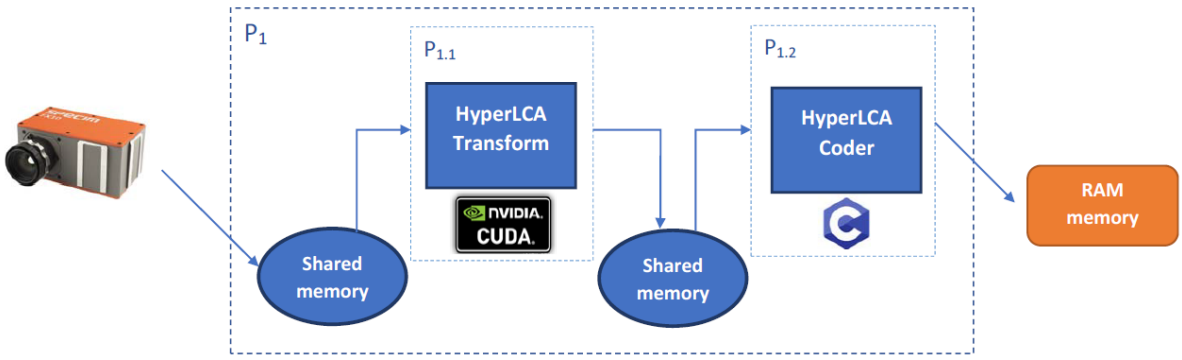

Figure 4. Previous work design for the real-time compression of the hyperspectral data.

Ring buffers are used to prevent the saturation of the RAM memory of the board. These ring buffers allow for a maximum number of frames to be stored at the same time in the memory. Once that the last position of the ring buffer is written, it returns to the first one, overwriting the information that is present in this position. This methodology is able to achieve real-time performance and very high compression speeds, as demonstrated in [21]. For doing so, the HyperLCA Transform, which is the most computational demanding part of the HyperLCA compressor, has been implemented into the LPGPU that is available in

the Jetson board using a set of self developed kernels that were programmed using CUDA. However, this methodology presents some weaknesses that need to be overcome for the purpose of this work:

- **Information lost if any part of the process delays**. If anything affects the performance of the compression or transmission process during the mission, so that one of them is delayed, part of the hyperspectral data will be lost. This is due to the fact that the data are never written to a non-volatile memory on-board, and that the ring buffers will be overwritten after a while.
- **Original uncompressed data are not preserved.** The captured hyperspectral frames are just stored in the corresponding ring buffer in RAM memory, from where it is read by the compression process. However, they are never stored in a non-volatile memory and, hence, just the compressed-decompressed data will be available for the analysis. Because the HyperLCA compressor is a lossy transform-based one, the original data cannot be recovered.
- **Restricted input data format.** The overall compression process that is presented in [21] was developed assuming that the captured hyperspectral frames would be placed in the RAM memory in Band Interleaved by Pixel (bip) format using 16 bits unsigned integer values and coded in little-endian. However, many of the hyperspectral pushbroom cameras, such as the Specim FX10, work in Band Interleaved by Line (bil) format and use different precision and endianness.
- **Real-time data transmission not tested.** While, in [21], it was proved that it was possible to achieve real-time compression performance using the HyperLCA compressor in the Jetson boards, the transmission to the ground station was not tested.

A new design is proposed in this work to overcome these weaknesses and achieve the desired performance. Figure 5 shows a graphic description of this design.

The main changes with respect to the design previously proposed in [21] are:

- **Acquired frames and compressed frames are stored in non-volatile memory**. Each acquired hyperspectral frame is independently stored in a single file in the non-volatile memory of the on-board computer. Both the compression and transmission processes read from these files. By doing so, it is guaranteed that all of the frames will be compressed and transmitted to the ground station, even if a delay occurs in any process. Additionally, the original uncompressed hyperspectral data can be extracted from the on-board computer once that the mission finishes and the UAV lands.
- **Flexible input data format.** The compression process has been adapted to be able to process data in different input formats and data types, namely (bip and bil, little-endian, and big-endian), as well as being able to adapt to 12 and 16 bits resolutions. This makes the proposed design more flexible and adaptable to different hyperspectral cameras.
- **Real-time data transmission tested.** The transmission of the compressed hyperspectral data from the on-board computer to the ground station has been tested in this work, thus verifying the possibility of achieving real-time performance.

Each of these changes is explained in detail in the subsequent sections.

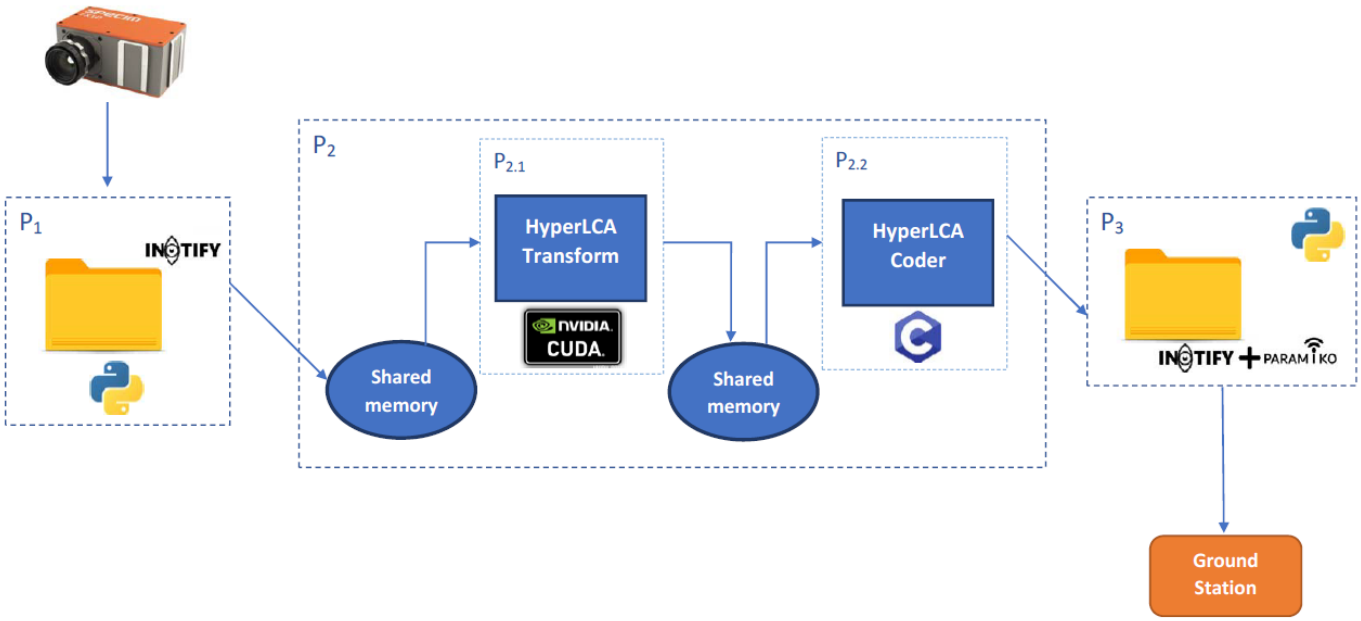

Figure 5. Proposed design for the real-time compression and transmission of the hyperspectral data

4.1.1. Use of Non Volatile Memory

As previously mentioned, in the design that is described in [21], both the captured frames and the compressed frames are stored in ring buffers in RAM memory. This may result in frames being lost if any of the compression or transmission processes delay during mission. In order to avoid this potential failure, the design proposed in this work stores each frame captured by the Specim FX10 hyperspectral camera as a single file in non-volatile memory. These files are read to feed the compression process, and the compressed bitstreams for each frame are also stored as files to non-volatile memory. By doing so, even if the compression or transmission process delays for a while, all of the frames will be compressed and transmitted to the ground station with a certain latency. Additionally, the original frames keep being stored in the on-board computer and they can be extracted from it once the mission finishes and the UAV lands.

Despite the clear advantages of this new methodology, it also presents some limitations. First of all, the writing and reading speed in the non-volatile memory is usually lower than the corresponding speed of the RAM memory, thus making it more challenging to achieve real-time performance. Additionally, it is necessary to add new steps to the overall process to manage the writing and reading stages of the produced files and the synchronization between the data capturing, compression, and transmission processes.

Different levels of parallelism have been employed in order to carry out this methodology. A graphic representation of the overall process is displayed in Figure 5. First of all, a Python process (P1 in Figure 5) is executed using the Python module, named iNotify, which permits event-driven-programing. This process reacts every time that a file is created in the target directory by the capturing process, reading the stored hyperspectral frame and then loading it into the RAM memory for the compression process (P2 in Figure 5).

Secondly, another process (referred to as P2 in Figure 5) reads the hyperspectral frames to be compressed from the RAM memory, compresses them, and stores the resulting bitstream in a new file. This process involves the execution of the HyperLCA Transform, implemented in the LPGPU included in the Jetson board, and the HyperLCA entropy coder, executed in the CPU. For synchronizing the processes included inside P2, a set of ring buffers and memory pointers is used. Concretely, two shared memory ring buffers are used for transferring the data between the different subprocesses:

- **Input Frame Ring Buffer**. Part of the shared memory where the hyperspectral data to be compressed are placed.
- **Transform Output Data**. Part of the shared memory where the results of the HyperLCA Transform are stored for its codification.

Additionally, four single shared memory pointers are used to synchronize the execution of the different parallel subprocesses:

- **Last Captured Frame Index**. Index indicating the last frame captured by the hyperspectral camera.
- **Last Transformed Frame Index**. Index specifying the last transformed frame.
- **Last Coded Frame Index**. Index indicating the last coded frame.
- **Process Finished**. Boolean value indicating if the overall process should stop.

Finally, another Python process (P3 in Figure 5) is executed using the Python module named iNotify, together with the Python module, named Paramiko. This process reacts every time that a compressed bitstream is written to a file by process P2 in the target directory and transmits it to the ground station via SSH connection.

Altogether, these processes (P1 to P3) result in the next synchronized behavior, which keeps allowing both the real-time compression and the real-time transmission processes while at the same time is able to recover itself from losses of capabilities and permits to analyze the original captured data after the drone lands and the data are extracted from the disk.

1. The Specim FX10 VNIR sensor captures a new frame and stores it in a specific folder placed into a non-volatile memory. In this moment, the Python module named iNotify, which permits event-driven-programing, detects the creation of a new hyperspectral file that is loaded into the *Input Frame Ring Buffer* and the *Last Captured Frame Index* is increased. If the HyperLCA Transform is delayed, it is detected by this process by checking the *Last Tranformed Frame Index*, preventing overwriting the data in the *Input Frame Ring Buffer* until the HyperLCA Transform moves to the next frame.
2. The HyperLCA Transform is continuously checking the *Last Captured Frame Index* to know when a new frame is loaded in the *Input Frame Ring Buffer*. In that moment, the frame is copied from the shared memory to the device memory to be compressed by the HyperLCA Transform. Once this process finishes, the corresponding bitstream is copied to the *Transform Output Data* and the *Last Transformed Frame Index* is increased. If the codification process delays, it is detected by this process by checking the *Last Coded Frame Index*, preventing overwriting the data in the *Transform Output Data Ring Buffer* until the HyperLCA Coder moves to the next frame.
3. The entropy coder is continuously checking the *Last Transformed Frame Index* to know when a new frame has been compressed by the HyperLCA Transform and stored in the *Transform Output Data*. After that, the frame is coded by the entropy coder and the resultant compressed frame is written into a specific folder within a non-volatile memory.
4. Once this file is written in the specific folder, the iNotify Python module detects this new element and triggers the transmission process of the compressed frame to the ground station via Secure Shell Protocol (SSH) while using the Python module, named Paramiko.

4.1.2. Flexible Input Data Format

As already described, the HyperLCA Transform stage, which was implemented into the LPGPU available in the Jetson board in [21], expects the input data to be formatted as 16 bits unsigned integers, in little-endian and BIP order. In order to make it more flexible and adaptable to any input data format, the first kernel of the HyperLCA Transform has been replaced by a set of six new kernels that support the conversion from different input data formats to the one that is required by the next kernel of the HyperLCA Transform. Each of these kernels first converts the data to 16 bits unsigned integer, in little-endian and

BIP order, as expected by the previous version, and then casts these values to floating point arithmetic, as it is required by the subsequent HyperLCA Transform operations. The kernel to be executed in each case is selected from the CPU while taking the input format of the hyperspectral frames to be compressed into account. The six new proposed kernels are:

- **uint16_le_bip_to_float**. This kernel is used when the input image is formatted as unsigned 16 bits integers, in bip format and little-endian.
- **uint16_be_bip_to_float**. This kernel is used when the input image is formatted as unsigned 16 bits integers, in bip format and big-endian.
- **uint16_le_bil_to_float**. This kernel is used when the input image is formatted as unsigned 16 bits integers, in bil format and little-endian.
- **uint16_be_bil_to_float**. This kernel is used when the input image is formatted as unsigned 16 bits integers, in bil format and big-endian.
- **uint12_bip_to_float**. This kernel is used when the input image is formatted as unsigned 12 bits integers, in bip format.
- **uint12_bil_to_float**. This kernel is used when the input image is formatted as unsigned 12 bits integers, in bil format.

4.1.3. Packing Coded Frames

As it has been stated before in Section 4.1.1, the hyperspectral data are transmitted from the UAV to the ground station once the compressed frames are stored in non-volatile-memory. In one hand, this is possible due to the use of the SSH protocol that establishes a secured connection between the on-board computer that was installed on the drone and the ground station. On the other hand, it is the Python module, named Paramiko, which implements that SSH connection generating a client-server (drone-ground station) functionality for the transmission of the hyperspectral information.

As already described, the compression process is individually applied to each frame and, so, each single frame could be written to a single file. Because the transmission process is triggered every time that a file is written in the specified location using the Python iNotify module, the transmission process would be triggered once per frame and a SSH transmission would be executed for each of them. However, this may lead to a situation in which the overhead produced in the SSH connection for initiating the transmission of each data file is closed to the time that is needed to transmit the data. Thereby, in order to overcome this issue, the compression process may accumulate a certain amount of compressed frames to write them all together in a single file, thus reducing the communication overhead at the cost of slightly increasing the transmission latency. Figure 6 illustrates this strategy, where each block corresponds to a compressed hyperspectral frame.

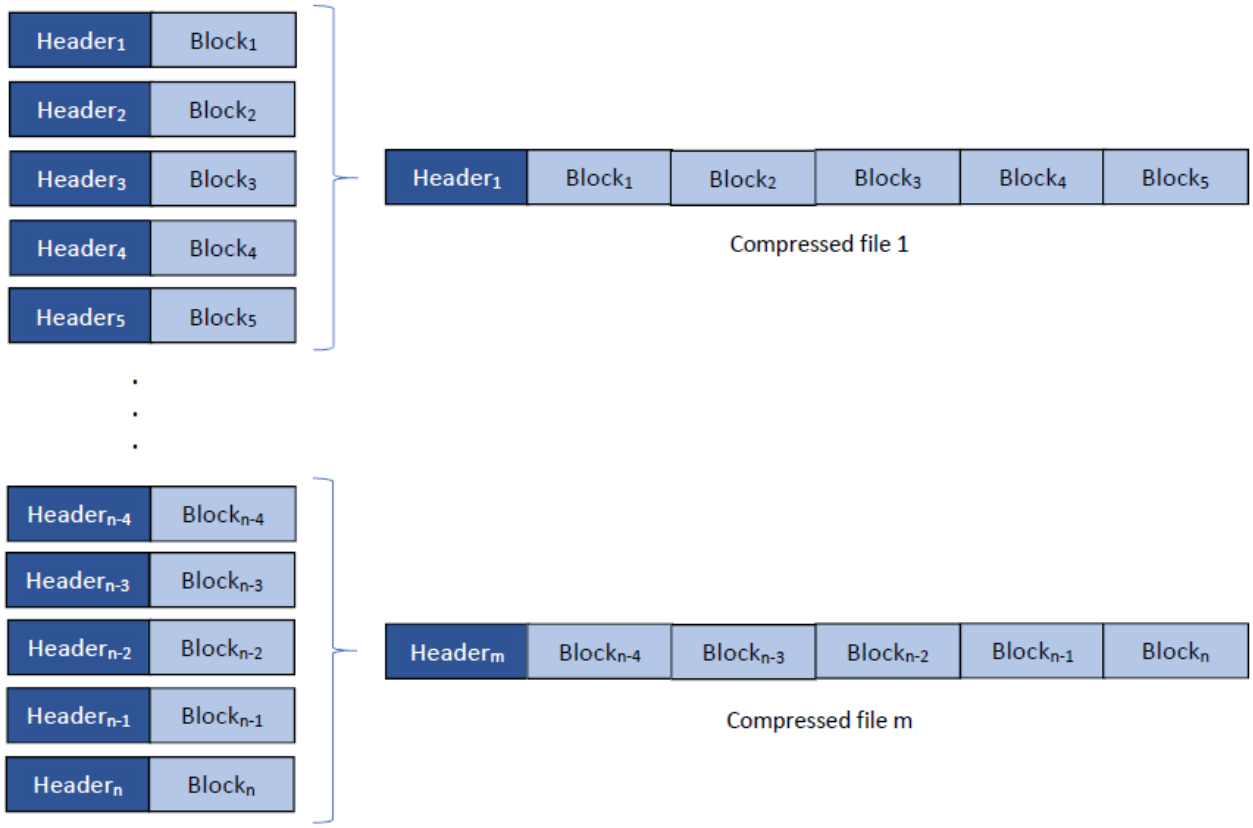

Figure 6. Strategy of packing various compressed frames into a single file

4.2. Data Set Used in the Experiments

The proposed design for the real-time hyperspectral data compression and transmission has been tested in this work carrying out multiple experiments and using real hyperspectral data that are captured by the aforementioned UAV-based acquisition platform [10]. This design includes a compression stage using the HyperLCA algorithm, whose performance, in terms of compression quality, has been evaluated in detail in previous works [20,21,28]. Accordingly, this work exclusively focuses in the development and validation of a design that can achieve a real-time hyperspectral data compression and transmission in a real scenario.

An available hyperspectral dataset that was obtained from one of the performed missions has been used to simplify the experiments execution and the evaluation process. This image is a subset of the hyperspectral data collected during a flight campaign over a vineyard area in a village called Tejeda located in the center of the island of Gran Canaria. The exact coordinates of the scanned terrain are 27°59'35.6" N 15°36'25.6" W (graphically indicated in Figure 7a. The flight was performed at a height of 45 m over the ground and at a speed of 4.5 m/s with the hyperspectral camera capturing frames at 150 FPS. This resulted in a ground sampling distance in line and across line of approximately 3 cm. This flight mission consisted of 12 waypoints that provided six swathes, but just one of them was used in the experiments that were carried out in this work. The ground area covered by this swath is highlighted in the Google Map picture displayed in Figure 7b. In particular, a smaller portion of 825 subsequent hyperspectral frames of 1024 pixels and 160 spectral bands each was selected for the experiments (as highlighted in Figure 7c. Despite the Specim FX10 camera used for acquiring this data can collect up to 224 bands, the first 20 bands and the last 44 ones were discarded during the capturing process due to a low *Signal-to-Noise-Ratio* (SNR), as described in [29].

Using the described hyperspectral image, an acquisition simulation process has been developed, which independently stores each of the 825 hyperspectral frames as a single

file in a target directory and at user defined speed. This process has been used in the experiments to emulate the camera capturing process with the possibility of controlling the simulated capturing speed, while, at the same time, allowing for the execution of all the experiments with the same data and in similar conditions.

The Specim FX10 camera used for collecting the aforementioned images measures the incoming radiation using a resolution of 12 *bpppb*. However, it can be configured in two different modes, 12-*Packed* and 12-*Unpacked* to store each value using 12, or 16 *bpppb*, respectively. In the 12-*Unpacked* mode, four zeros are padded at the beginning of each sample. With independence of the mode used, the hyperspectral frames are always stored in *bil* format. However, in order to test the proposed design with more details, the acquired image has been off-board formatted to store it using different configurations. On one side, it has been stored using the 12-*Packed*, *bil* format, which is the most efficient format that is produced by the sensor. This has been labeled as *uint12-bil* in the results. On the other side, it has been stored as 12-*Unpacked*, *bip* format, which is not a format that can be directly produced by the sensor and requires reordering the data. Additionally, this format requires more memory, which makes it less efficient. However, this format is the one for which the reference HyperLCA compressor and its parallel implementation proposed in [21] were originally designed, and allows for using it without including any of the additional data transformation described in Section 4.1.2. This has been labeled as *uint16-bip* in the results. The results that could be obtained with any other combination of data precision and samples order would be in the range of the results that were obtained for these two combinations.

5. Results

Several experiments have been carried out in this work to evaluate the performance of the proposed designed for real-time hyperspectral data compression and transmission. First, the achievable transmission speed that could be obtained without compressing the acquired hyperspectral frames is analyzed in Section 5.1. The obtained results verify that the data compression is necessary for achieving a real-time transmission. Secondly, the maximum compression speed that can be obtained without parallelizing the compression process using the available GPU has been tested in Section 5.2. The obtained results demonstrate the necessity of parallelizing the process using the available GPU to achieve real-time performance. Finally, the results that can be obtained with the proposed design taking advantage of the different levels of parallelism and the available GPU are tested in Section 5.3, demonstrating its suitability for the real-time hyperspectral data compression and transmission.

In the following Tables 4–7, the simulated capturing speed and the maximum compressing and transmission speeds are referred to as Capt, Comp, and Trans, respectively. In addition, the WiFi signal strength is named Signal in the aforementioned tables of results, a value that is directly obtained from the WLAN interface during the transmission process.

5.1. Maximum Transmission Speed without Compression

First of all, the maximum transmission speed that could be achieved if the acquired frames were not compressed has been tested. For doing so, the targeted capturing speed has been set to 100 FPS, since it is the lowest frame rate typically used in our flight missions. Table 4 displays the obtained results.

Table 4. Transmission speed without compressing the frames.

Input Parameters			Jetson Xavier NX Speed (FPS)			Jetson Nano Speed (FPS)		
Format	FPS	Packing	Capt	Trans	Signal	Capt	Trans	Signal
uint12 - bil	100	1	99	26	42/100	98	13	100/100
uint12 - bil	100	5	103	36	45/100	101	18	100/100
uint16 - bip	100	1	101	21	47/100	93	11	100/100
uint16 - bip	100	5	100	30	38/100	105	15	100/100

As it can be observed, when independently transmitting each frame (Packing = 1), the transmission speed is too low in relation to the capturing speed, even capturing at 100 FPS. When transmitting five frames at a time, the transmission speed increases, but it is still very low. It can also be observed that, when using the *uint12-bil* format, the transmission speed is approximately 25 percent faster than when using the *uint16-bip* one. This makes sense, since the amount of data per file when using the *uint12-bil* format is 25 percent lower than the *uint16-bip* one. These results verify the necessity of compressing the acquired hyperspectral data before transmitting it to the ground station.

Additionally, it can also be observed in the results that the Jetson Xavier NX is able to achieve a higher transmission rate than the Jetson Nano, even if the Jetson Nano has a higher WiFi signal strength. This could be due to the fact that the Jetson Nano is using an external TP-Link TL-WN722N WiFi antenna that is connected through USB2.0 interface, while the WiFi antenna used by the Jetson Xavier NX is integrated in the board.

5.2. Maximum Compression Speed without Gpu Implementation

Once the necessity of carrying out the compression of the acquired hyperspectral data has been demonstrated, the necessity of speeding up this process by carrying out a parallel implementation is to be tested. For doing so, the compression process within the HyperLCA algorithm has been serially executed in the CPU integrated in both boards for evaluating the compression speed. The capturing speed has been set to the minimum value used in our missions (100 FPS), as done in Section 5.1. The compression parameters have been set to the less restrictive values typically used for the compression within the HyperLCA algorithm (CR = 20 and N_{bits} = 12). From the possible combinations of input parameters for the HyperLCA compressor, these values should lead to the fastest compression results according to the analysis done in [21], where the HyperLCA compressor was implemented onto a Jetson TK1 and Jetson TX2 developing boards. Additionally, the data format used for this experiment is *uint16-bip*, since this is the data format for which the reference version of the HyperLCA compressor is prepared. By using this data format, the execution of extra data format transformations that would lead to slower results is prevented. Table 5 displays the obtained results.

Table 5. Compression speed without using parallelism.

Input Parameters					Jetson Xavier NX Speed (FPS)		Jetson Nano Speed (FPS)	
Format	CR	N_{bits}	FPS	Packing	Capt	Comp	Capt	Comp
uint16 - bip	20	12	100	1	103	23	96	9

The compression speed achieved without speeding up the process is too low and far from a real-time performance, as it can be observed in Table 5. This demonstrates the necessity of a parallel implementation that takes advantage of the available LPGPUs integrated in the Jetson boards for increasing the compression speed. It can also be observed that the Jetson Xavier NX offers a better performance that the Jetson Nano.

5.3. Proposed Design Evaluation

In this last experiment, the performance of the proposed design has been evaluated for both the Jetson Xavier NX and the Jetson Nano boards. The capturing speed has been set to the minimum and maximum values that are typically used in our flying missions, which are 100 FPS and 200 FPS, respectively. Two different combinations of compression parameters have been tested. The first one, CR = 20 and N_{bits} = 12, corresponds to the less restrictive scenario and should lead to the fastest compression results according to [21]. Similarly, the second one, CR = 12 and N_{bits} = 8, corresponds to the most restrictive case and it should lead to the slowest compression results, as demonstrated in [21]. Furthermore, both the *uint16-bip* and *uint12-bil* data formats have been tested. Finally, the compressed frames have been packed in two different ways, individually and in groups of 5. All of the obtained results are displayed in Table 6.

The Jetson Xavier NX is always capable of compressing more than 100 frames per second, regardless of the configuration used, as observed in Table 6. However, it is only capable of achieving 200 FPS in the compression in the less restrictive scenario, which is *uint12-bil*, CR=20, N_{bits}=12 and Packing=5. Additionally, when capturing at 200 FPS, there are other scenarios in which real-time compression is almost achieved, producing compression speeds up to 190 FPS or 188 FPS. On the other hand, the Jetson Nano only presents a poorer compression performance, achieving real-time compression speed in the less restrictive scenario.

Regarding the transmission speed, it can be observed that, when packing the frames in groups of five, the transmission speed is always the same as the compression speed, but in the two fastest compression scenarios in which the compression speed reaches 200 and 188 FPS, the transmission speed stays at 185 and 162 FPS, respectively. This may be to the fact that the WiFi signal strength is not high enough (36/100 and 39/100) in these two tests. Nevertheless, two additional tests have been carried out for these particular situations with the Jetson Xavier NX, whose results are displayed in Table 7. A real-time transmission is achieved when increasing the packing size to 10 frames, as shown in this table. It can be also observed that in these two tests real-time compression has also been achieved. This may be due to the fact that a lower number of files are being written and read by the operating system. This suggests that a faster performance could be obtained in the Jetson Xavier NX using a solid stage disk (SSD) instead of the SD card that has been used so far in these experiments.

Table 6. Speed results for the proposed design.

Input Parameters					**Jetson Xavier NX Speed (FPS)**				**Jetson Nano Speed (FPS)**			
Format	CR	N_{bits}	FPS	Packing	Capt	Comp	Trans	Signal	Capt	Comp	Trans	Signal
uint12 - bil	20	12	100	1	104	104	75	48/100	93	93	56	100/100
				5	106	106	106	37/100	101	101	101	100/100
			200	1	197	190	77	47/100	185	105	59	100/100
				5	202	200	185	36/100	208	110	110	100/100
	12	8	100	1	104	104	69	48/100	91	44	44	100/100
				5	104	104	104	37/100	94	44	44	100/100
			200	1	205	140	73	47/100	205	44	44	100/100
				5	187	138	138	37/100	187	45	45	100/100
uint16 - bip	20	12	100	1	102	102	75	47/100	85	82	39	100/100
				5	103	103	103	38/100	98	83	83	100/100
			200	1	189	166	65	46/100	194	82	52	100/100
				5	193	188	162	39/100	186	84	84	100/100
	12	8	100	1	100	99	68	47/100	99	36	36	100/100
				5	101	101	101	38/100	99	36	36	97/100
			200	1	196	119	49	38/100	188	35	35	100/100
				5	171	110	110	37/100	208	36	36	100/100

Table 7. Speed results for the proposed design when increasing the packing size to 10 frames.

Input Parameters					Jetson Xavier NX Speed (FPS)			
Format	CR	N_{bits}	FPS	Packing	Capt	Comp	Trans	Signal
uint12 - bil	20	12	200	10	206	206	206	37/100
uint16 - bip	20	12	200	10	203	203	203	48/100

Additionally, the time at which each hyperspectral frame has been captured, compressed, and transmitted in these two last experiments is graphically displayed in Figure 8. Figure 8a shows the results that were obtained for the *uint12-bil* data format, while Figure 8b shows the values that were obtained for the *uint16-bip* one. In these two graphics, it can be observed how, on average, the slope of the line representing the captured frames (in blue color), the line representing the compressed frames (in orange), and the line representing the transmitted frames (in green color) is the same, indicating a real-time performance. It can also be observed the effect of packing the frames in groups of 10 in the transmission line. Finally, Figure 8a also shows the effect of a short reduction in the transmission speed due to a lower WiFi signal quality. As it can be observed in this figure, the transmission process is temporarily delayed and so is the compression one to prevent writing new data to the shared memory before processing the existing one. After a while, both the connection and the overall process are fully recovered, demonstrating the benefits of the proposed methodology previously exposed in Section 4.1.

Finally, all of the obtained results demonstrate that packing the compressed frames reduces the communication overhead and accelerates the transmission process. It can also be observed that using the data in *uint12-bil* format accelerates the compression process, since less data are been transferred through the shared memory, demonstrating the benefits of the developed kernels for allowing the compression process to be adapted to any input data format.

In general, the results that were obtained for the Jetson Xavier NX demonstrate that the proposed design can be used for real-time compressing and transmitting hyperspectral images at most of the acquisition frame rates typically used in our flying missions and using different configuration parameters for the compression. Additionally, it could be potentially faster if a solid stage disk (SSD) memory were used instead of the SD card that was used in the experiments. Regarding the Jetson Nano, a real-time performance was hardly achieved in just the less restrictive scenarios.

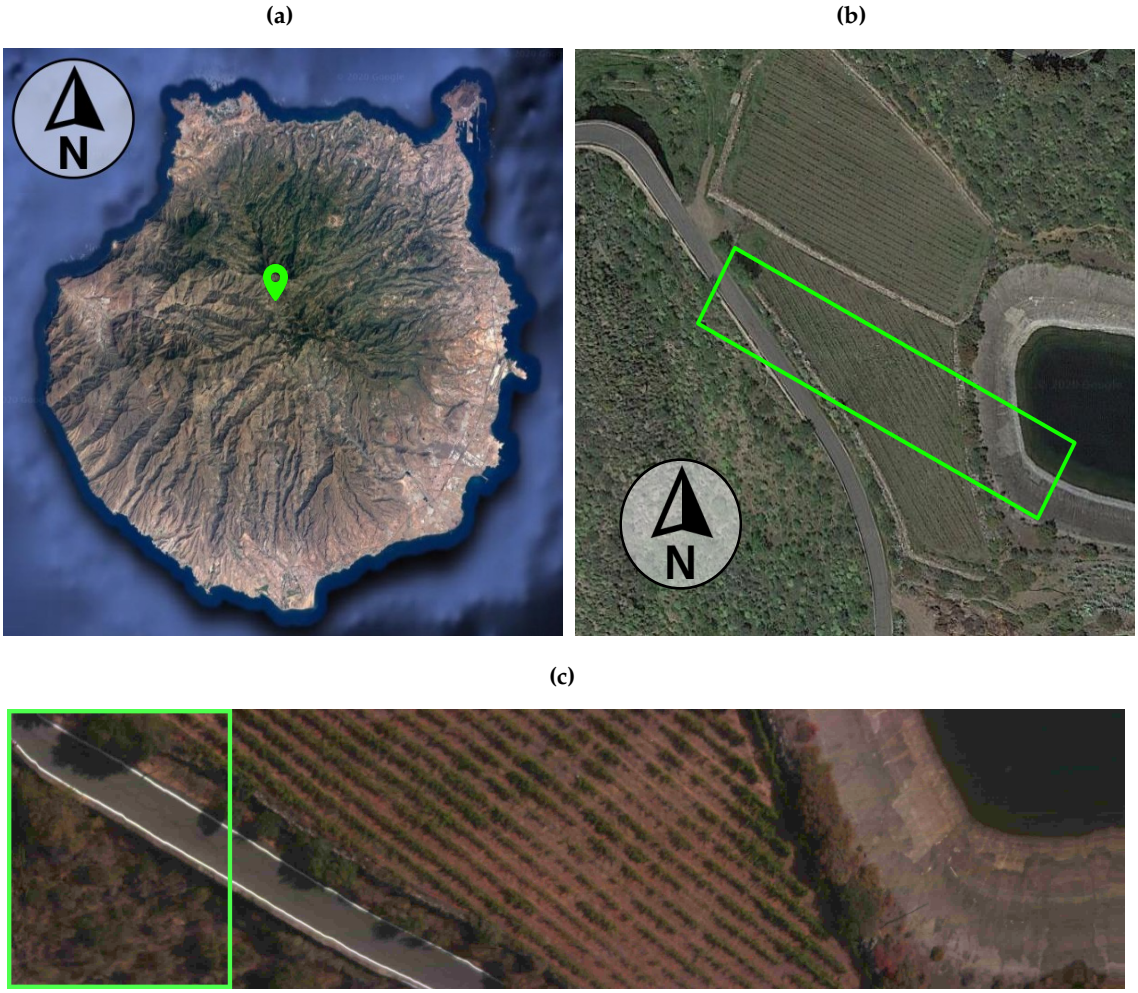

Figure 7. Graphical description of the appearance and location of the terrain corresponding to the image used in this work for the experiments. (**a**) Google Maps pictures indicating the terrain location on the island of Gran Canaria. (**b**) Google Maps pictures indicating the area covered during the selected swath of the flight campaign over the vineyard. (**c**) RGB respresentation of the real hyperspectral data, highlighting the portion of it that was used for the experiments.

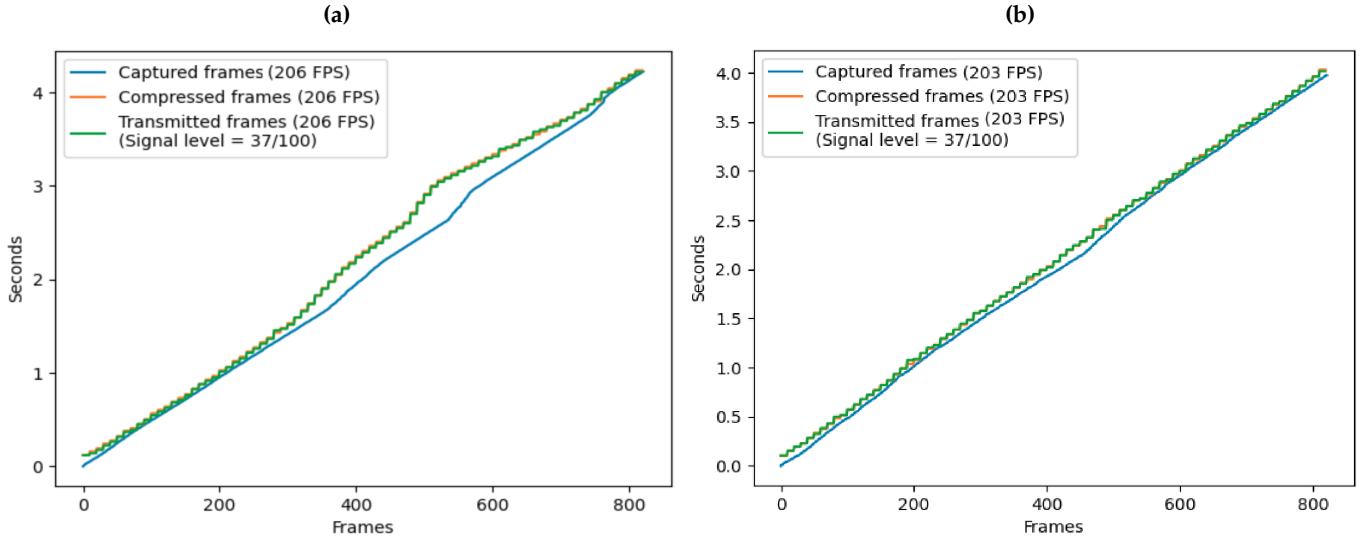

Figure 8. Graphical representation of the time in which each hyperspectral frame is captured, compressed and transmitted for the experiments displayed in Table 7. (**a**) *uint12-bil* version. (**b**) *uint16-bip* version.

6. Discussion

The main goal of this work is to be able to transmit the hyperspectral data captured by an UAV-based acquisition platform to a ground station in such a way that it can be analyzed and/or visualized in real-time to take decisions on the flight. The experiments conducted in Section 5.1 show the necessity of compressing the acquired hyperspectral data in order to achieve a real-time transmission. Nevertheless, the achievement of a real-time compression performance on-board this kind of platforms is not an easy task when considering the high data rate that is produced by the hyperspectral sensors and the limited computational resources available on-board. This fact has been experimentally tested in Section 5.2. While considering this, the compression algorithm to be used must meet several requirements, including a low computational cost, a high level of parallelism that allows for taking advantage of the LPGPUs available on-board to speed up the compression process, being able to guarantee high compression ratios, and integrate an error resilience nature to ensure that the compressed data can be delivered as expected. These requirements are very similar to those found in the space environment, where the acquired hyperspectral data must be rapidly compressed on-board the satellite to save transmission bandwidth and storage space, using limited computational resources and ensuring an error resilience behaviour.

The HyperLCA compressor has been selected in this work for carrying out the compression of the acquired hyperspectral data, since this compressor was originally developed for the space environment and satisfies all the mentioned requirements [20] [28]. This compressor has been tested in previous works against those that were proposed by the *Consultative Committee for Space Data Systems* (CCSDS) in their *Recomended Standards (Blue Books)* [25]. Concretely, in [20], it was compared with the Karhunen–Loeve transform-based approach described in the *CCSDS 122.1-B-1 standard* (Spectral Preprocessing Transform for Multispectral and Hyperspectral Image Compression) [30] as the best spectral transform for decorrelating hyperspectral data in terms of accuracy. The results shown in [20] indicate that the HyperLCA transform is able to achieve a similar decorrelation performance, but at a much lower computational cost and introducing extra advantages, such as higher levels of parallelism and an error resilience behavior. In [20], the HyperLCA compressor was also tested against the lossless prediction-based approach that was proposed in the *CCSDS 123.0-B-1 standard* (Lossless Multispectral and Hyperspectral Image Compression) [31]. As expected, the compression ratios that can be achieved by a lossless approach are very far from those that are required by the application targeted in this work. The CCSDS has recently published a new version of this prediction-based algorithm, making it able to behave not only as a lossless solution, but also as a near-lossless one to achieve higher compression ratios. This new solution has been published under the *CCSDS 123.0-B-2 standard* (Low-Complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression) [32] and it represents a very interesting alternative to be considered in future works.

On the other hand, although presenting a new methodology to transmit hyperspectral information from a UAV to a ground station in real-time is the main goal of this research work, the ulterior hyperspectral imaging applications have been always taken into account during the development process. This means that, while it is necessary to carry out lossy-compression to meet the compression ratios that are imposed by the acquisition data rates and transmission bandwidth, the quality of the hyperspectral data received in the ground station has to be good enough to preserve the desired performance in the targeted applications. As previously described in this work, the HyperLCA compressor was developed following an unmixing-like strategy in which the most different pixels present in each image block are perfectly preserved through the compression-decompression process. This is traduced in the fact that most of the information that is lost in the compression process corresponds to the image noise, while the relevant information is preserved, as demonstrated in [20,29]. In [20], the impact of the compression-decompression process within the HyperLCA algorithm was tested when using the decompressed images for hyperspectral linear unmixing, classification, and anomaly detection, demonstrating that

the use of this compressor does not negatively affect the obtained results. This specific study was carried out while using well known hyperspectral datasets and algorithms, such as the *Pavia University* data set coupled with the *Support Vector Machine* (SVM) classifier, or the *Rochester Institute of Technology* (RIT) and the *World Trade Center* (WTC) images coupled with the *Orthogonal Subspace Projection Reed-Xiaoli* (OSPRX) anomaly detector. The work presented in [29] carries out a similar study, just for anomaly detection, but using the hyperspectral data that were collected by the acquisition platform used in this work and with the exact same configurations, both in the acquisition stage and compression stage. Concretely, the data used in this work, as described in Section 4.2, are a reduced subset of the hyperspectral data used in [29].

Finally, all of this work has been developed while assuming that a Wireless Local Area Network (WLAN), based on the 802.11 standard, will be available during the flight, and that both the on-board computer and ground station will be connected to it to use it as data downlink. Further research works are needed to increase the availability and range of this kind of networks or to be able to integrate the proposed solution with another wireless transmission technologies to make it available to a wider range of remote sensing applications.

7. Conclusions

In this paper, a design for the compression and transmission of hyperspectral data acquired from an UAV to a ground station has been proposed so that it can be analysed and/or visualized by an operator in real-time. This opens the possibility of taking advantage of the spectral information collected by the hyperspectral sensors for supervised or semi-supervised applications, such as defense, survilliance, or search and rescue missions.

The proposed design assumes that a WLAN will be available during the flight, and that both the on-board computer and the ground station will be connected to it to use it as data downlink. This design can work with different input data formats, which allows for using it with most of the hyperspectral sensors present in the market. Additionally, it preserves the original hyperspectral data in a non-volatile memory, producing two additional advantages. On one side, if the connection is lost for a while during the mission, the information is not lost, and the process will go on once the connection is recovered, guaranteeing that all of the acquired data are transmitted to the ground station. On the other side, once the mission finishes and the drone lands, the real hyperspectral data can be extracted from the non-volatile memory without compression if a more detailed analysis is required.

The entire design has been tested using two different boards from NVIDIA that integrate LPGPUs, the Jetson Xavier NX, and the Jetson Nano. The LPGPU has been used for accelerating the compression process, which is required for decreasing the reduction of the data volume for its transmission. The compression has been carried out using the HyperLCA algorithm, which permits achieving high compression ratios with a relatively high rate-distortion relation and at a reduced computational cost.

Multiple experiments have been executed to test the performance of all the stages that build up the proposed design for both the Jetson Xavier NX and Jetson Nano boards. The results obtained for the Jetson Xavier NX demonstrate that the proposed design can be used for real-time compressing and transmitting hyperspectral images at most of the acquisition frame rates typically used in our flying missions and using different configuration parameters for the compression. Additionally, it could be potentially faster if a solid stage disk (SSD) memory was used instead of the SD card that was used in these experiments. On the other hand, when using the Jetson Nano, a real-time performance was achieved in just the less restrictive scenarios.

Future research lines may include the optimization of the proposed design for reducing its computational burden, so that it can achieve a more efficient performance, especially when using boards with more limitations in terms of computational resources as an on-board computer.

Author Contributions: R.G. proposed the design for the real-time compression and transmission of the hyperspectral data. J.M.M. programmed the necessary GPU code, aided by M.D. who created the simulation platform. J.F.L. acquired the hyperspectral image from an UAV; S.L. conceived and designed the simulations; A.M. and A.J. performed the simulations; P.H. supervised the technical work and paper reviews. All authors contributed to the interpretation of the results and the writing of the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been funded by the Ministry of Economy and Competitiveness (MINECO) of the Spanish Government (PLATINO projecto, no. TEC2017-86722-C4-1 R), the European Comission and the European Regional Development Fund (FEDER) under project APOGEO (grant number MAC/1.1.b/226) and the Agencia Canaria de Investigación, Innovación y Sociedad de la Información (ACIISI) of the Conserjería de Economía, Industria, Comercio y Conocimiento of the Gobierno de Canarias, jointly with the European Social Fund (FSE) (POC2014-2020, Eje 3 Tema Prioritario 74 (85%)).

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ro, K.; Oh, J.S.; Dong, L. Lessons learned: Application of small uav for urban highway traffic monitoring. In Proceedings of the 45th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, 8 January 2007–11 January 2007; p. 596.
2. Chen, Y.M.; Dong, L.; Oh, J.S. Real-time video relay for uav traffic surveillance systems through available communication networks. In Proceedings of the 2007 IEEE Wireless Communications and Networking Conference, Hong Kong, China, 11–15 March 2007; pp. 2608–2612.
3. Qu, Y.; Jiang, L.; Guo, X. Moving vehicle detection with convolutional networks in UAV videos. In Proceedings of the 2016 2nd International Conference on Control, Automation and Robotics (ICCAR), Hong Kong, China, 28–30 April 2016; pp. 225–229.
4. Zhou, H.; Kong, H.; Wei, L.; Creighton, D.; Nahavandi, S. Efficient road detection and tracking for unmanned aerial vehicle. *IEEE Trans. Intell. Transp. Syst.* **2014**, *16*, 297–309.
5. Silvagni, M.; Tonoli, A.; Zenerino, E.; Chiaberge, M. Multipurpose UAV for search and rescue operations in mountain avalanche events. *Geomat. Nat. Hazards Risk* **2017**, *8*, 18–33.
6. Scherer, J.; Yahyanejad, S.; Hayat, S.; Yanmaz, E.; Andre, T.; Khan, A.; Vukadinovic, V.; Bettstetter, C.; Hellwagner, H.; Rinner, B. An autonomous multi-UAV system for search and rescue. In Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use, Florence, Italy, May 2015; pp. 33–38.
7. Doherty, P.; Rudol, P. A UAV search and rescue scenario with human body detection and geolocalization. In *Australasian Joint Conference on Artificial Intelligence*; Springer, Gold Coast, Australia, 2-6 December 2007; pp. 1–13.
8. Li, Z.; Liu, Y.; Hayward, R.; Zhang, J.; Cai, J. Knowledge-based power line detection for UAV surveillance and inspection systems. In Proceedings of the 2008 23rd International Conference Image and Vision Computing New Zealand, Christchurch, New Zealand, 26–28 November 2008; pp. 1–6.
9. Semsch, E.; Jakob, M.; Pavlicek, D.; Pechoucek, M. Autonomous UAV surveillance in complex urban environments. In Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, Milan, Italy, 15–18 September 2009; Vol. 2, pp. 82–85.
10. Horstrand, P.; Guerra, R.; Rodríguez, A.; Díaz, M.; López, S.; López, J.F. A UAV platform based on a hyperspectral sensor for image capturing and on-board processing. *IEEE Access* **2019**, *7*, 66919–66938.
11. Zarco-Tejada, P.J.; Guillén-Climent, M.L.; Hernández-Clemente, R.; Catalina, A.; González, M.; Martín, P. Estimating leaf carotenoid content in vineyards using high resolution hyperspectral imagery acquired from an unmanned aerial vehicle (UAV). *Agric. For. Meteorol.* **2013**, *171*, 281–294.
12. Vanegas, F.; Bratanov, D.; Powell, K.; Weiss, J.; Gonzalez, F. A novel methodology for improving plant pest surveillance in vineyards and crops using UAV-based hyperspectral and spatial data. *Sensors* **2018**, *18*, 260.
13. Mitchell, J.J.; Glenn, N.F.; Anderson, M.O.; Hruska, R.C.; Halford, A.; Baun, C.; Nydegger, N. Unmanned aerial vehicle (UAV) hyperspectral remote sensing for dryland vegetation monitoring. In Proceedings of the 2012 4th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), Shanghai, China, 4–7 June 2012; pp. 1–10.
14. Valentino, R.; Jung, W.S.; Ko, Y.B. A design and simulation of the opportunistic computation offloading with learning-based prediction for unmanned aerial vehicle (uav) clustering networks. *Sensors* **2018**, *18*, 3751.
15. Freitas, S.; Silva, H.; Almeida, J.; Silva, E. Hyperspectral imaging for real-time unmanned aerial vehicle maritime target detection. *J. Intell. Robot. Syst.* **2018**, *90*, 551–570.
16. Specim, Specim FX Series Hyperspectral Cameras. Available online: http://www.specim.fi/fx/ (accessed on 4 May 2019).
17. DJI, MATRICE 600 PRO. Available online: https://www.dji.com/bg/matrice600 (accessed on 4 May 2019).

18. NVIDIA, Jetson Nano Developer Kit. Available online: https://developer.nvidia.com/embedded/jetson-nano-developer-kit (accessed on 4 May 2019).
19. NVIDIA, Jetson Xavier NX. Available online: https://www.nvidia.com/es-es/autonomous-machines/embedded-systems/jetson-xavier-nx/ (accessed on 4 May 2019).
20. Guerra, R.; Barrios, Y.; Díaz, M.; Santos, L.; López, S.; Sarmiento, R. A new algorithm for the on-board compression of hyperspectral images. *Remote Sens.* **2018**, *10*, 428.
21. Díaz, M.; Guerra, R.; Horstrand, P.; Martel, E.; López, S.; López, J.F.; Sarmiento, R. Real-time hyperspectral image compression onto embedded GPUs. *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.* **2019**, *12*, 2792–2809.
22. Díaz, M.; Guerra, R.; López, S.; Sarmiento, R. An Algorithm for an Accurate Detection of Anomalies in Hyperspectral Images With a Low Computational Complexity. *IEEE Trans. Geosci. Remote. Sens.* **2018**, *56*, 1159–1176.
23. Guerra, R.; Santos, L.; López, S.; Sarmiento, R. A new fast algorithm for linearly unmixing hyperspectral images. *IEEE Trans. Geosci. Remote. Sens.* **2015**, *53*, 6752–6765.
24. Guerra, R.; López, S.; Sarmiento, R. A computationally efficient algorithm for fusing multispectral and hyperspectral images. *IEEE Trans. Geosci. Remote. Sens.* **2016**, *54*, 5712–5728.
25. Consultative Committee for Space Data Systems (CCSDS), Blue Books: Recommended Standards. Available online: https://public.ccsds.org/Publications/BlueBooks.aspx (accessed on 16 January 2021).
26. Howard, P.G.; Vitter, J.S. Fast and efficient lossless image compression. In Proceedings of the Data Compression Conference, 1993. DCC'93, Snowbird, UT, USA, 30 March–2 April 1993; pp. 351–360.
27. TP-LINK, TP-LINK-TL-WN722N. Available online: https://www.tp-link.com/es/home-networking/adapter/tl-wn722n/#overview (accessed on 12 November 2020).
28. Guerra, R.; Barrios, Y.; Díaz, M.; Baez, A.; López, S.; Sarmiento, R. A hardware-friendly hyperspectral lossy compressor for next-generation space-grade field programmable gate arrays. *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.* **2019**, *12*, 4813–4828.
29. Díaz, M.; Guerra, R.; Horstrand, P.; López, S.; López, J.F.; Sarmiento, R. Towards the Concurrent Execution of Multiple Hyperspectral Imaging Applications by Means of Computationally Simple Operations. *Remote Sens.* **2020**, *12*, 1343.
30. Consultative Committee for Space Data Systems (CCSDS), Blue Books: Recommended Standards. Available online. Spectral Preprocessing Transform for Multispectral and Hyperspectral Image Compression. Available online: https://public.ccsds.org/Pubs/122x1b1.pdf (accessed on 17 Frebuary 2021).
31. Consultative Committee for Space Data Systems (CCSDS), Blue Books: Recommended Standards. Available online. Lossless Multispectral and Hyperspectral Image Compression. Available online: https://public.ccsds.org/Pubs/123x0b1ec1s.pdf (accessed on 17 Frebuary 2021).
32. Consultative Committee for Space Data Systems (CCSDS), Blue Books: Recommended Standards. Available online. Low-Complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression. Available online: https://public.ccsds.org/Pubs/123x0b2c2.pdf (accessed on 17 February 2021).