DOI: 10.1016/j.future.2018.07.051

One-pass lossless data hiding and compression of remote sensing data☆

Bruno Carpentieri, Arcangelo Castiglione, Alfredo De Santis, Francesco Palmieri, Raffaele Pizzolante *

Dipartimento di Informatica, Università degli Studi di Salerno, Via Giovanni Paolo II, 132, I-84084, Fisciano (SA), Italy

HIGHLIGHTS

- A one-pass framework for lossless compression and simultaneously reversible data hiding for remote sensing data.
- In a single pass a marked and compressed image is produced by the proposed framework.
- The resulting compressed marked image can be decompressed and restored (the exact input image is obtained) or decompressed only (a marked version of the input image is obtained).
- The marked image can be used for several purposes, in which it is not necessary to extract the original data and an acceptable grade of degradation is tolerated.

Keywords:
Reversible data hiding
Reversible invisible watermark
Hyperspectral data hiding
Hyperspectral images protection
Data integrity
Authentication

ABSTRACT

The information obtained by means of spectral remote sensing (i.e., the hyperspectral images) are involved in several real-life scenarios and applications. Historical research, monitoring of environmental hazards, forensics and counter-terrorism are some examples of contexts in which the hyperspectral data play an important role.

In many contexts, the hyperspectral images could also play sensitive roles (e.g., in military applications, etc.) and are generally exchanged among several entities, in order to carry out different tasks on them. Therefore, it is important to guarantee their protection. A meaningful choice is the protection through *data hiding* techniques.

In fact, by means of reversible data hiding techniques, the imaging data become a sort of *information carrier* and can be used for delivering other important data that can be used, for instance, to check the integrity of the original imaging data.

In this paper, we introduce a one-pass framework that is able to perform the lossless data hiding and the lossless compression of the marked stream, at the same time, by exploiting the capabilities of the predictive paradigm. Substantially, in a single pass, a marked and compressed stego image is obtained, which can be exactly restored by the receiver: by decompressing and reversibly reconstructing the original unaltered image. In addition, our framework also permits to perform only the decompression (without the extraction of the hidden information). In this manner, the resulting stego (marked) hyperspectral image, could be used for several purposes, in which it is not necessary to extract the original data and an acceptable grade of degradation is tolerated. We also implement a *proof-of-concept* of the proposed framework to assess the effectiveness of our contribution. Finally, we report the achieved experimental results, which outperform other similar approaches.

1. Introduction

By means of remote sensing it is possible to acquire information about distant objects, without coming into physical contact with them, by exploiting the fact that an object reflects, absorbs and emits electromagnetic radiation according to its chemical composition [1–3]. Again, by analyzing the energy of the radiation as a function of the wavelength, it is possible to obtain a *spectral signature*. A spectral signature can be considered as a sort of unambiguous fingerprint that can be used to uniquely characterize any given material. For example, different organizations, such as NASA, etc., have cataloged the spectral signatures of various minerals, hence permitting an easier identification of such materials.

☆ This document is a collaborative effort.

* Corresponding author.
E-mail addresses: bcarpentieri@unisa.it (B. Carpentieri), arcastiglione@unisa.it (A. Castiglione), ads@unisa.it (A. De Santis), fpalmieri@unisa.it (F. Palmieri), rpizzolante@unisa.it (R. Pizzolante)

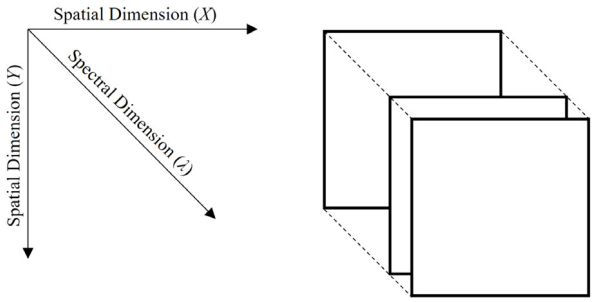


Fig. 1. Example of a hyperspectral datacube.

The information acquired through spectral remote sensing (i.e., the *hyperspectral images*) are used for several purposes and applications, varying from surveillance to historical research, archeology, environmental sciences, monitoring of environmental hazards, assessment of food quality, and several other real-life contexts, such as laboratory research, biomedical and Earth imaging, forensics, counter-terrorism, skin health checking, etc. Notice that such images can be air-borne and space-borne acquired.

In detail, a hyperspectral sensor acquires information by considering the portion of the electromagnetic spectrum that varies from the visible part (400–760 nanometers — nm) to near-infrared (about 2400 nm). More precisely, we can see a hyperspectral image as a *datacube*, since it is composed by a sequence of narrow contiguous *spectral bands*, composing a three-dimensional data, as shown in Fig. 1. In general, each band covers a bandwidth of about 10 nm and a hyperspectral image is composed by a few hundreds of bands [1,4,5].

For example, hyperspectral images produced by the *NASA AVIRIS (Airborne Visible/Infrared Imaging Spectrometer)* sensors are composed by 224 bands, and are acquired by revealing the frequencies of ultraviolet and infrared rays at wavelengths in the range between 400 and 2500 nm [1].

Motivations. Given the different multidisciplinary application contexts of hyperspectral images, in general, it is very likely that such images are exchanged between different entities, to effectively carry out a wide range of tasks, typically performed in cloud-based environments. Again, many of the application contexts of hyperspectral images are security-sensitive, so particular effort should be devoted in their protection. Although cryptography is commonly used as an effective tool for ensuring security and content authentication [6–9], however, the disguised nature of encrypted data may draw the attention of an adversary. In such a scenario, *data hiding* has been found to be an efficient and effective alternative to cryptography for ensuring security, content authentication and copyright protection [10–14]. In addition, by means of data hiding techniques, imaging data become a sort of *information carriers*, which can be used for delivering other important data [15]. An image for carrying data is called a *cover image*, whereas, the image carrying the embedded information is called a *stego image* or *marked image*. Data hiding on images presents two (virtually conflicting) requirements: the *embedding capacity*, i.e., the number of bits embedded into the host image and the *visual quality* of the embedded image. Notice that if during the transmission an image shows signs of hiding effect, an adversary may notice that the digital media carries secret messages. Therefore, the stego image should have imperceptible differences with respect to the cover image [16,17]. The goal of data hiding is to create schemes with high embedding capacity and good stego image quality. In particular, *reversible data hiding (RDH)* is a technique which not only enables the embedding of hidden data into cover images, but also restores original images from the stego images after the extraction of the embedded data [18–28]. However, the drawbacks of RDH with respect to non-reversible techniques are: lower payloads, larger distortion and higher computational cost.

Again, it is important to highlight that images collected through hyperspectral sensors are generally processed by automatic applications [29]. Thus, it is highly suggested to store, maintain and transmit such data into the original form. However, since hyperspectral sensors generate a significant amount of data, it can be necessary to compress such data. Moreover, though lossy compression algorithms achieve a better compression ratio than lossless ones, they introduce distortion and make the data usable only by applications where a certain level of distortion is allowed. Therefore, it is easy to note that in this scenario only the lossless compression of hyperspectral images permits to have a reduction in the amount of data, without any loss of useful information.

Contribution. In this work we propose a novel *one-pass* framework, which by relying on the capabilities of a predictive paradigm, enables at the same time the data hiding and compression operations on hyperspectral images. More precisely, it achieves, in a single pass, a marked and compressed stego image. On the other side, the receiver can extract the hidden information and then decompress and reversibly reconstruct the original unaltered image. Again, our framework also enables the receiver to perform the decompression without extracting the hidden information. The resulting stego (marked) hyperspectral image could be used for several purposes, e.g., in situations where it is not necessary to extract the original data and an acceptable grade of degradation is tolerated.

In the literature, several lossless compression algorithms for hyperspectral images, based on the predictive paradigm, have been proposed (e.g., the ones addressed in [30–32]). On the other hand, despite various watermarking methods for remote sensing data have been introduced in the state of the art (e.g., [33–35]), the field concerning the high capacity lossless data hiding has not been exhaustively explored so far. Indeed, to the best of our knowledge, our proposal represents the first one-pass framework specifically designed for hyperspectral images, which is able to perform the lossless data hiding and the lossless compression of the marked stream, at the same time. In addition, our framework is also suitable for *on-board* implementations (i.e., for implementations on a hardware-constrained sensor), since it is designed to require limited computational capability and memory space. More precisely, the proposed framework relies on a data hiding approach based on the *modification of prediction errors (MPE)* technique [36] and exploits the fact that, since the histograms in the domain of prediction errors are sharply distributed, the embedding capacity is higher than that of traditional *histogram-shifting methods* for the same image quality. The key idea behind our proposal is that we employ three-dimensional predictors, which are ad-hoc designed to exploit, in the best way possible, the redundancies of hyperspectral images, rather than using bi-dimensional predictors, as for example has been done in [37].

In order to assess the effectiveness of the proposed scheme, we carried out extensive performance evaluation experiments, which show a saving in terms of execution time, ranging from 20.08% to 47.73% with respect to more traditional solutions performing data hiding and then compression. Moreover, our framework enables a gain in terms of embedding capacity ranging from 5.48% to 170.8% with respect to state-of-the-art solutions. Again, by varying the size and the type of the embedded data payload, the two images, namely, the cover image and the produced stego image, are extremely similar and hence difficult to distinguish, thus making the detection of the hidden payload by a malicious user extremely difficult. Finally, when compared to the pure compression, i.e., the compression performed on images which do not carry any hidden payload, our proposal does not significantly affect the compression

performance in terms of bits per sample. In particular, the loss in terms of compression performance ranges from 0.47% to 8.88%.

Organization. The paper is structured as follows: in Section 2 we describe concepts and techniques underlying our proposal; in Section 3 we describe the scheme for reversible data hiding in hyperspectral images, by highlighting its main characteristics and advantages with respect to the state of the art; in Section 4 we show and comment results achieved by testing our proposal on a publicly available dataset. Finally, in Section 5 we draw our conclusions and highlight future research directions.

2. Preliminaries

In this section we describe concepts and techniques underlying our proposal. More precisely we first describe the main features of the histogram-shifting technique. Afterwards, we provide an overview of the data hiding based *on Modification of Prediction Errors (MPE)*.

2.1. The histogram-shifting technique

Histogram-shifting is a technique introduced by Ni et al. [38], which embeds in a reversible manner the data into the image, by shifting the histogram bins. More precisely, in such a method, at most one grayscale level in each pixel is changed. In detail, the embedding procedure of the histogram shifting technique works as follows [38]:

1. Obtain the histogram $h(x), x \in [0, 255]$ of the 8-bit cover image.
2. Find the maximum value $h(\alpha)$ and the minimum value $h(\beta)$ of $h(x)$, where $\alpha, \beta \in [0, 255]$. The parameter α is referred to as *peak point*, while β is referred to as *minimum point*. Again, if $h(\beta) = 0$, then β is referred to as *zero point*. Notice that is assumed $\alpha < \beta$.
3. If the minimum value $h(\beta) > 0$, then store in the array L_β all the positions of the pixels having gray level β and set $h(\beta) = 0$.
4. Scan the whole image in a sequential order, i.e., row-by-row and from the top to the bottom. Then, shift the histogram $h(x)$, with $x \in (\alpha, \beta)$, to the right by one unit. In this way, all pixel values satisfying $x \in (\alpha, \beta)$ are added by one.
5. Scan the whole image again and embed the secret bits together with L_β (if any). If the pixel value is α and the bit to be embedded is 1, then the pixel value is set to $\alpha + 1$, whereas, if the bit to be embedded is 0, the pixel value is not modified.

In the histogram-shifting technique, if $h(\beta) > 0$, additional recovery information L_β should embedded into the cover image. Formally, the *embedding capacity (EC)* of this method is given by Eq. (1),

$$EC = h(\alpha) - \left\| L_\beta \right\| \ , \tag{1}$$

where $\left\| L_\beta \right\|$ is the number of bits required to store L_β. To extract the secret data and restore the stego image, the following procedure is used.

1. Scan the stego image in the same order as that used in the embedding phase. If a pixel value $\alpha + 1$ is found, a bit 1 is extracted, whereas, if a pixel value α is found, a bit 0 is extracted.
2. Scan the stego image again. If the pixel value $x \in (\alpha, \beta]$, then the pixel value x is subtracted by 1.
3. If the recovery information L_β is found in the extracted data, set the pixel value to β if the location of this pixel is found in L_β. In this way, the original image can be recovered without any distortion.

2.2. Data hiding based on modification of prediction errors (MPE)

The distortion arising from histogram-shifting technique is mainly based on the number of feature elements which are shifted. Therefore, the number of embeddable elements should be increased, whereas, the number of feature elements that has to be shifted should be minimized.

In general, the performance of the histogram-shifting technique can be improved by generating features which have a sharply distributed histogram. In this way, the number of embeddable elements can be maximized. For this reason, Hong et al. [37] proposed a reversible data hiding technique which is based on *modification of prediction errors (MPE)*. Instead of using feature elements in the spatial domain, Hong et al. use a predictor to create feature elements in the domain of prediction errors. MPE relies on the histogram-shifting technique to embed data in the domain of prediction errors. More precisely, MPE does not need to generate the error histogram, due to some characteristics of the error histogram for most natural images.

One of the most important advantages of the MPE is that the prediction significantly increases the number of embeddable features within the feature set, thus increasing the embedding capacity. Furthermore, the position of the peak point which maximizes the embedding capacity is known, since there is a peaked histogram centered at prediction value equal to zero. Therefore, to search the peak and minimum point, there is no need to scan the image again. Again, no evacuation of histogram bins is needed beforehand. In detail, MPE only modifies fewer prediction errors to embed fewer amounts of data. Finally, the embedding procedure only performs simple operations, e.g., addition and subtraction, and the whole image is scanned just one time during the embedding process.

3. The data hiding and compression framework

In this section we describe the proposed *one-pass* framework for hyperspectral images. We stress that since such images are collected by means of remote sensing, one of the most important features of our framework is that it is suitable for *on-board* implementation, due to its limited computational complexity and memory space required. Indeed, as stated before, our framework allows, at the same time, to hide a data payload and compress the marked stream.

We remark that the data hiding strategy described in Section 3.1 is based on the *modification of prediction errors (MPE)* technique [36]. The main characteristic of the MPE technique is the use of the feature elements in the domain of prediction errors to hide data, instead of using the features in the spatial domain.

The main idea underlying such technique is that since the histograms in the domain of prediction errors are sharply distributed, the embedding capacity is higher than that of traditional *histogram-shifting methods* for the same image quality. In detail, *histogram-shifting methods* shift the histograms of the feature elements to prepare vacant positions for the embedding. The occurrence of the most frequent feature elements, which are called *embeddable elements*, determines the embedding capacity, whereas, the distortion resulting from histogram-shifting embedding, primarily depends on the number of feature elements that are shifted. On the other hand, the MPE technique only modifies less error values, for embedding fewer data bits. Therefore, MPE-based techniques are able to achieve high quality stego images.

In Section 3.2 we present the architecture and the overall logical functioning of the proposed scheme. In particular, we describe how the modeling of the prediction errors can be exploited to perform, at the same time, the data hiding and compression of the marked stream.

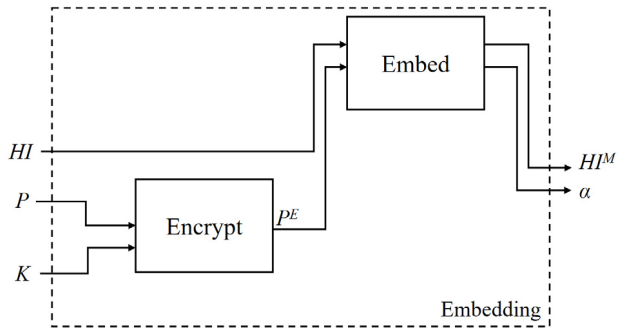


Fig. 2. Block diagram of the embedding process.

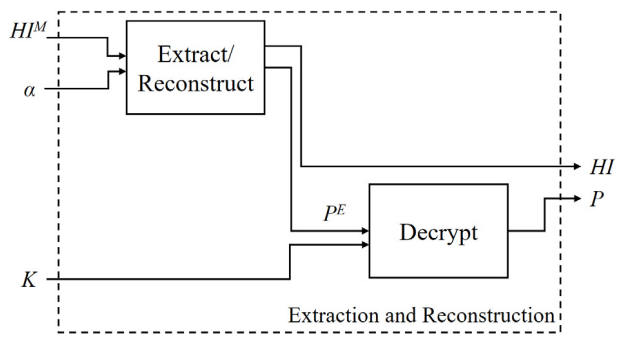


Fig. 3. Block diagram of the extraction and reconstruction processes.

3.1. Lossless data hiding

In Fig. 2, we show the block diagram for the embedding process. In the first stage of this process, the input payload P, i.e., the message to be hidden, is encrypted by using the input key K. Subsequently, the encrypted payload P^E is embedded in the input hyperspectral image HI. The embedding process returns two outputs: HI^M and α, where HI^M is the marked hyperspectral image carrying the hidden information P^E, whereas, α contains the auxiliary information needed for the extraction and reconstruction. We emphasize that α is a set of coordinates and its size is bounded by a certain number of bits. In the following, we will provide more details concerning such aspect.

In Fig. 3 we show the block diagram for the extraction and reconstruction processes. Starting from the input HI^M and the auxiliary information α, the encrypted payload P^E is extracted and the original hyperspectral image HI is reconstructed. Subsequently, P^E is decrypted by using the input key K to obtain the original payload P.

By means of the Algorithm 1 we describe the details of the embedding process. In detail, the first phase of the embedding concerns the initialization of the used data structures. More precisely, the memory space for the output HI^M is allocated, while the first row and the first column of each band of HI are copied into HI^M, since such data are not altered by the embedding process. Afterwards, α is set to be empty. We remark that the set α will be used to contain some auxiliary information. Finally, a global variable i is initialized to 1, to be used as a pointer to the next bit of P^E which will be embedded. Basically, each sample x of HI is processed, by means of the three *for* loops (from line 7 to line 35). From now on, we denote by the term *sample* a pixel of the hyperspectral image. Again, we denote as HI_R and HI_C the number of rows and columns of the hyperspectral image taken as input, respectively. Once processed, the modified sample x^M is stored in HI^M. HI^M represents the marked hyperspectral image and will be returned as the output of the whole embedding process. In particular, the sample x^M is predicted by using a three-dimensional predictor, except for the samples of the first band (when b is equal to 1), which are predicted through a bi-dimensional predictor, since the first band has no previous band as reference. It is important to emphasize that the prediction is performed by considering the neighboring samples (i.e., the *prediction context*) retrieved from HI^M. In this way, also the extraction and reconstruction processes are able to perform the same prediction of the embedding, since such processes use as input the marked hyperspectral image HI^M. Then, the result of the prediction is stored in $\widehat{x}^M$. We remark that the *extractAndReconstruct* procedure (Algorithm 3) outlines the extraction and reconstruction processes. Notice that the *extractAndReconstruct* procedure uses the *reconstructErrorValue* procedure, which is reported in Algorithm 4.

Subsequently, the prediction error e is computed by subtracting the value of $\widehat{x}^M$ from the value of x of HI. The prediction error e is then modified by means of the *calculateNewErrorValue* procedure of Algorithm 2, to embed (if possible) the value of the bit P_i^E, i.e., the ith bit of the encrypted payload P^E. As it can be observed from Algorithm 2, the ith bit of P^E (i.e., P_i^E) can be embedded only when the value of the prediction error e is equal to 0 or is equal to -1. In particular, there are no modification on e (thus, $e_N = e$) when P_i^E is equal to 0, and e is equal to 0 or equal to 1. Otherwise, if P_i^E is equal to 1, e is modified, by increasing its value of 1 (if e is equal to 0) or by decreasing its value of 1 (if e is equal to -1). Again, if it is possible to embed P_i^E, the global variable i is increased and points to the next bit to embed. We remark that the prediction errors histogram will be modified also when P_i^E cannot be embedded, by increasing e of 1 when its value is greater than 0 or by decreasing e of 1 when its value is less than -1. The modified prediction error is then stored in e_N and it is added to $\widehat{x}^M$. The result of this latter sum, denoted as x^M, is then stored in the marked hyperspectral image HI^M.

It is important to point out that the value of the modified prediction e_N could give rise to underflow or overflow issues. For instance, assume the following scenario: each sample of the input hyperspectral image HI (and consequently of the marked hyperspectral image HI^M) can take a value in the range $[HI_{min}, HI_{max}]$, the value of $\widehat{x}^M$ is equal to HI_{min} and the value of e_N is -1. In such a scenario, the value of x^M, obtained by computing $\widehat{x}^M + e_N$, will be $HI_{min} - 1$, which is out of the allowed range, thus leading to an underflow issue. A similar scenario could arise also in the case of an overflow issue, if we consider a scenario where $\widehat{x}^M = HI_{max}$ and e_N is equal to 1. Therefore, it is clear that overflow and underflow issues need to be managed, in order to obtain as output a valid hyperspectral image.

Starting from the consideration that a prediction error e can be modified by adding (or subtracting) 1 to (from) its value (according to Algorithm 2), the samples that do not cause underflow or overflow can assume values in a limited range, i.e., $]HI_{min}, HI_{max}[$. Thus, the samples which have a value equal to HI_{min} or HI_{max} need to be cutted off, since they may cause underflow/overflow issues. In the following, we refer to such samples as *not valid samples*. When dealing with such samples, the algorithm stores the coordinates of each sample in the set α. In addition, assuming that the coordinates relative to each of the above samples are (r, c, b), denoting that the sample is located in the bth band at the rth row and the cth column, all of the other samples of such a band will not carry any further bits of P^E. Indeed, the not valid sample and all the other samples of the bth band of HI_M, will be set to have the same value as the respective ones of HI.

We emphasize that the size of the set α is less or equal than the number of bands in HI, which is referred to as HI_B. Hence, the size of the representation of α is easily estimable and it is less or equal than $HI_B \times M$ bits, where M is the number of bits needed to represent the coordinates of a sample in HI. It is important to

point out that the representation of α has a significant smaller size than the capacity of embedding, therefore, such a representation can be embedded into HI^M, through a low-complexity reversible data hiding approach similar to the one proposed in [39].

Finally, it is important to highlight that the auxiliary information α is essential for the extraction and reconstruction processes. Indeed, such processes do not know whether a sample which stores a value of HI_{min} or HI_{max} is a sample that hides a bit or has a value equal to the respective one in the original image, i.e., a sample which has not been altered by the embedding algorithm, since it may cause an underflow/overflow issue.

3.2. Lossless data embedding and compression

In general, lossless compression schemes for remote sensing images are based on the *Predictive-Coding Model*, so that such schemes can be suitable for on-board implementations, e.g., directly on sensors or instruments, which usually are hardware-constrained. In detail, two main stages characterize a predictive-based compression algorithm for remote sensing images:

1. Context-modeling;
2. Prediction error modeling and coding.

In the context-modeling phase, a prediction model (often denoted as *predictor*) is used. In particular, each sample x is predicted by means of the predictor, which uses the already coded neighbors of the sample itself. The output of the prediction phase is the *predicted sample*, denoted as $\hat{x}$. It is important to emphasize that each sample x is replaced by its prediction error e. A prediction error e is calculated through the difference between the value of x and the value of $\hat{x}$, as shown by Eq. (2).

$$e = x - \hat{x} \,. \tag{2}$$

We remark that the context-modeling phase (or *prediction phase*) is the most important phase, since it exploits the redundancy among the samples. In general, during the prediction error modeling and coding phase, each prediction error is first mapped through an invertible mapping function similar to the one reported by Eq. (3),

$$m(e) = \begin{cases} 2|e| & if\ e \geq 0, \\ 2|e| - 1 & otherwise. \end{cases} \tag{3}$$

It is important to note that the mapping function does not alter the redundancy among the prediction errors. Subsequently, the mapped prediction errors are encoded by means of an *entropy coder* (e.g., *arithmetic coder*, etc.).

In Section 3.1 we described how the prediction is exploited to hide the bits of the payload P into an input hyperspectral image HI. Instead, in the current section, we mainly focus on the description of the proposed *one-pass* framework, by highlighting how it is possible to exploit the capabilities of the prediction to carry out, at the same time, two tasks, namely, the hiding of bits and the compression of the achieved bit stream.

Fig. 4a shows the block diagram highlighting the key aspects related to the *embed* procedure discussed in Section 3.1, whereas, Fig. 4b, shows the block diagram characterizing a generic model of a prediction-based lossless compression algorithm for hyperspectral images. In Fig. 4c, we show a block diagram highlighting the key points related to the proposed framework, whose details are described by the *embedAndCompress* procedure (Algorithm 5). From Fig. 4c, it is easy to note that the proposed one-pass framework is made up by combining data hiding and compression schemes.

More precisely, the *embedAndCompress* procedure allows the hiding of the payload P into the input hyperspectral image HI

Algorithm 1: The pseudo-code of the *embed* procedure.

Input:

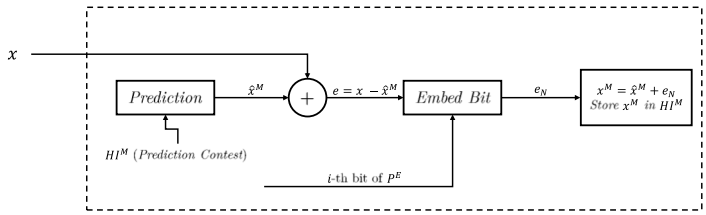
- **HI**: Input Hyperspectral Image
- **P**: Payload to hide in **HI**
- **K**: Key for the encryption of **P**

Output:

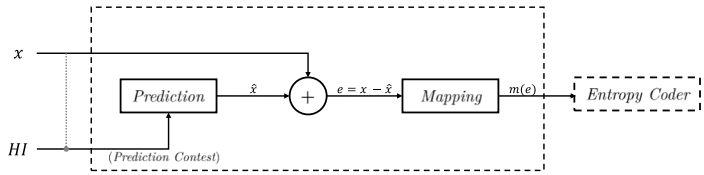
- $\mathbf{HI^M}$: Marked Hyperspectral Image
- α: Auxiliary Information

1: **procedure** EMBED(HI, P, K)
2: *Allocate the necessary memory space for* HI^M;
3: *Copy the first row and the first column of each band from HI to* HI^M;
4: *Encrypt the payload P with the key K and store the result in* P^E;
5: *Initialize the set* α *to be empty*, i.e., $\alpha = \varnothing$;
6: *Initialize the global variable i to* 1, i.e., $i = 1$;
7: **for** $r = 2$ **to** HI_R **do**
8: **for** $c = 2$ **to** HI_C **do**
9: **for** $b = 1$ **to** HI_B **do**
10: Let x be the sample at the (r, c) coordinates of the bth band of HI;
11: Let x^M be the sample at the (r, c) coordinates of the bth band of HI^M;
12: Let β be a set defined as $\beta = \{b_j | \forall (r_j, c_j, b_j) \in \alpha\}$;
13: **if** $b \in \beta$ **then**
14: $x^M = x$;
15: *Store* x^M *in* HI^M *and continue with the next iteration*;
16: **end if**
17: **if** $x == HI_{min}$ **or** $x == HI_{max}$ **then**
18: $x^M = x$;
19: **if** *not all the bits of* P^E *are embedded* **then**
20: $\alpha = \alpha \cup \{(r, c, b)\}$;
21: **end if**
22: *Store* x^M *in* HI^M *and continue with the next iteration*;
23: **end if**
24: $\widehat{x}^M$ = prediction of x^M using the prediction context obtained from HI^M;
25: $e = x - \widehat{x}^M$
26: **if** *not all the bits of* P^E *are embedded* **then**
27: $e_N = calculateNewErrorValue(P_i^E, e)$;
28: **else**
29: $e_N = e$;
30: **end if**
31: $x^M = \widehat{x}^M + e_N$;
32: *Store* x^M *in* HI^M;
33: **end for**
34: **end for**
35: **end for**
36: **return** HI^M and α;
37: **end procedure**

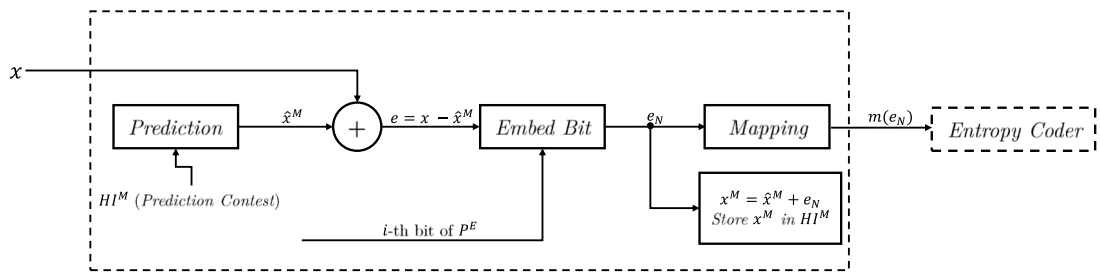
and, at the same time, the compression of HI. Similarly to the *embed* procedure, in the initialization phase (from line 2 to 5) the following data structures are allocated and initialized: HI^M, the set α and the global variable i. In addition, the payload P is encrypted by using the input key K and the result is stored in P^E. Notice that HI^M represents the marked hyperspectral image, which will be maintained in memory and built step-by-step during the process. Moreover, it is important to point out that HI^M will be used in the



(a) Block Diagram (Data Hiding).



(b) Block Diagram (Compression).



(c) Block Diagram (*Proposed*).

Fig. 4. Block Diagrams of the key points of (a) Data Hiding, (b) Compression and (c) Proposed Framework.

prediction phase. In fact, as it can be observed from line 12 of the *embed* procedure, the prediction of $\hat{x}^M$ is performed by using the prediction context obtained from HI^M. We stress that it is essential to perform the prediction from HI^M, since the decompression and extraction algorithm (Algorithm 6) is only able to process the prediction errors of HI^M and it does not have access to the original input hyperspectral image *HI*. Therefore, also the *embedAndCompress* procedure has to work on the marked hyperspectral image HI^M, to perform the correct and coherent extraction of the hidden data, as well as the decompression and the reconstruction.

It is easy to note that the *embedAndCompress* procedure (Algorithm 5) is very similar to the *embed* procedure (Algorithm 1). The main difference is given by the instruction reported in the line 34 of the *embedAndCompress* procedure, where the modified prediction error e_N is mapped through the invertible mapping function reported in Eq. (3). We remark that e_N carries, if possible, the *i*th bit of P^E, denoted as P_i^E. Once mapped, e_N is sent to an entropy coder. The subprocedure responsible to perform the mapping and the sending is the *mapAndSendError*. Instead, in the *embed* procedure, the modified prediction error e_N is only used to compute x^M, which is subsequently stored in the marked hyperspectral image HI^M.

Notice that the two conditional statements (*if*) involved in the managing of the underflow and overflow issues are exactly the same in both the *embed* (lines 13–16 and 17–23) and *embedAndCompress* (lines 18–21 and 22–28) procedures, except for the fact that only in the *embedAndCompress* procedure the prediction error *e* is mapped and sent to an entropy coder (lines 19 and 23, respectively). By using the prediction error *e*, the decompression and extraction procedure (Algorithm 6) is able to reconstruct exactly the value of the pixel *x*, since such a pixel has the same value in both *HI* and HI^M.

Finally, we stress that the management of the first row and the first column of *HI* is slightly different between the *embed* and *embedAndCompress* procedures. In detail, the *if* conditional statement reported from line 14 to 17 of the *embedAndCompress* procedure (Algorithm 5) enables to send the first row and the first column of *HI* to an entropy coder, by means of the *sendPixelValue* procedure, without performing the prediction and mapping. In this manner, the decompression and extraction procedure (Algorithm 6) can properly reconstruct the first row and the first column. Notice that in the *embed* procedure (Algorithm 1), such information has been exactly copied from *HI* to HI^M (line 3 of the *embed* procedure).

3.2.1. *Optimizing the embedding phase*

The *embedAndCompress* procedure maintains in memory the whole marked hyperspectral image HI^M. We remark that HI^M needs to have the same dimensions as *HI*. It is important to note that this aspect could be a drawback for eventual on-board implementations, e.g., due to the constrained hardware capabilities of the sensors. Starting from the consideration that for the prediction of the current sample *x* is only used its prediction context, namely, the neighboring samples in the current band and in the previous bands, it is possible to optimize the used memory space by maintaining, for each band, only such latter samples, which are effectively involved in the prediction phase of *x*.

4. Experimental results

In this section we report the results of the performed testing activity, aimed at assessing the performance and effectiveness of the proposed data hiding and compression scheme. In particular, in this section we first show that such a scheme is able to achieve a saving, in terms of execution time, ranging from 20.08% to 47.73%, with respect to the traditional solutions which first perform data hiding and then compression. We also show that

Algorithm 2: The pseudo-code of the *calculateNewErrorValue* procedure.

Input:

- $\mathbf{P_i^E}$: $i-th$ bit of the encrypted payload P^E
- $\mathbf{e}$: Prediction error

Output:

- $\mathbf{e_N}$: Modified Prediction Error used to compute the marked image

```
procedure CALCULATENEWERRORVALUE(P_i^E, e)
    e_N = e;
    if e == 0 or e == −1 then
        if P_i^E == 1 then
            if e == −1 then
                e_N = e − 1;
            else
                e_N = e + 1;
            end if
        end if
        Increase of 1 the global variable i;
    else
        if e > 0 then
            e_N = e + 1;
        else
            e_N = e − 1;
        end if
    end if
    return e_N;
end procedure
```

through our framework it is possible to achieve a gain in terms of embedding capacity ranging from 5.48% to 170.8%. Furthermore, by employing standard metrics for evaluating image quality, we can observe that by varying the size and type of the embedded data payload, the two images, namely, the cover image and the marked image, are extremely similar and difficult to distinguish, making the detection of the hidden payload extremely difficult. Finally, we show that when compared to the pure compression, i.e., the compression performed on images which do not carry any hidden payload, our proposal does not significantly affect the compression performance in terms of bits per sample. In particular, the loss in terms of compression performance ranges from 0.47% to 8.88%.

4.1. Involved metrics

Potentially, any kind of processing performed on a given image may cause loss of information or quality. Such a loss should be evaluated, in order to assess image quality. Available methods for evaluating image quality can be classified in two categories: *subjective* and *objective*. The former ones rely on human judgment and work without reference to explicit criteria. Conversely, the latter methods rely on comparisons through explicit numerical criteria and can use several references, for example the ground truth, prior knowledge expressed in terms of statistical parameters and tests, etc. [40].

There are mainly two categories of objective quality (or distortion) assessment methods. The first category is characterized by mathematically defined measures, e.g., *mean squared error (MSE)*, *peak signal to noise ratio (PSNR)*, *root mean squared error (RMSE)*, *mean absolute error (MAE)*, *signal-to-noise ratio (SNR)*, etc. The second category of measurement methods is based on *human visual system (HVS)* characteristics, in an attempt to integrate perceptual quality measures [41].

Algorithm 3: The pseudo-code of the *extractAndReconstruct* procedure.

Input:

- $\mathbf{HI^M}$: Input Marked Hyperspectral Image
- $\mathbf{K}$: Key for the decryption of the reconstructed payload $\mathbf{P^R}$
- α: Auxiliary Information

Output:

- $\mathbf{HI^R}$: Reconstructed Hyperspectral Image
- $\mathbf{P^R}$: Reconstructed and decrypted payload

```
procedure EXTRACTANDRECONSTRUCT(HI^M, K, α)
    Allocate the necessary memory space for HI^R;
    Copy the first row and the first column of each band from HI^M
to HI^R;
    Initialize the set α and the set β to be empty, i.e., α = ∅ and
β = ∅;
    Initialize the global variable i to 1, i.e., i = 1;
    for r = 2 to HI_R^M do
        for c = 2 to HI_C^M do
            for b = 1 to HI_B^M do
                Let x^M be the sample at the (r, c) coordinates of
the bth band of HI^M;
                if all the bits of P^E are extracted then
                    x^R = x^M;
                    Store x^R in HI^R and continue with the next itera-
tion;
                end if
                if b ∈ β or (r, b, c) ∈ α then
                    x^R = x^M;
                    if (r, c, b) ∈ α then
                        β = β ∪ {b};
                    end if
                    Store x^R in HI^R and continue with the next itera-
tion;
                end if
                x̂^R = prediction of x^R using the prediction context
obtained from HI^R;
                e = x^M − x̂^R
                e_R = reconstructErrorValue(e, P^E);
                x^R = x̂^R + e_R;
                Store x^R in HI^R;
            end for
        end for
    end for
    Decrypt the extracted payload P^E with the key K and store the
result in P^R;
    return HI^R and P^R;
end procedure
```

It is important to emphasize that mathematically defined measures are easy to calculate and have low computational complexity. Again, such measures are independent from both viewing conditions and individual observers. We remark that if there are different viewing conditions, a method which depends on the viewing condition will generate different measurement results. Furthermore, it is in charge of the user to measure the viewing conditions, as well as to calculate and insert the condition parameters to the measurement systems. Conversely, a method which does not depend on the viewing conditions, provides a single quality value that enables to assess how good the image is [41]. From now on, we denote by $\mathbf{x} = \{x_i \mid i = 1, 2, \ldots, N\}$ and $\mathbf{y} = \{y_i \mid i = 1, 2, \ldots, N\}$ the original and the test image signals, respectively.

Algorithm 4: The pseudo-code of the *reconstructErrorValue* procedure.

Input:

- $\mathbf{e}$: Error Value
- $\mathbf{P^R}$: Encrypted payload that will be extracted

Output:

- $\mathbf{e^R}$: Reconstructed Error Value

```
procedure RECONSTRUCTERRORVALUE(e, P^E)
    if e == 0 then
        P^E = concatBitValue(P^E, 0);
        Increase of 1 the global variable i;
    else
        if e == 1 then
            P^E = concatBitValue(P^E, 1)
            e_R = e − 1;
            Increase of 1 the global variable i;
        else
            if e == −1 then
                P^E = concatBitValue(P^E, 0)
                Increase of 1 the global variable i;
            else
                if e == −2 then
                    P^E = concatBitValue(P^E, 1)
                    e_R = e + 1;
                    Increase of 1 the global variable i;
                else
                    if e > 0 then
                        e_R = e − 1;
                    else
                        e_R = e + 1;
                    end if
                end if
            end if
        end if
    end if
    return e_R;
end procedure
```

4.1.1. Peak Signal-to-Noise Ratio (PSNR)

In order to measure the distortion between the original hyperspectral image and the marked one, i.e., the one which carries hidden data, we consider the *Peak Signal-to-Noise Ratio (PSNR)* metric [40], outlined by means of Eq. (4),

$$PSNR(HI_{(i)}, HI^M_{(i)}) = 10 \log_{10} \left(\frac{(2^{15}-1)^2}{MSE(HI_{(i)}, HI^M_{(i)})} \right) , \tag{4}$$

where $HI_{(i)}$ is the ith band of hyperspectral image HI and $HI^M_{(i)}$ is the ith band of the marked hyperspectral image HI^M, respectively. The MSE is defined by means of Eq. (5),

$$MSE(HI_{(i)}, HI^M_{(i)}) = \frac{1}{HI_R HI_C} \sum_{c=1}^{W} \sum_{r=1}^{H} \left(HI_{(r,b,c)} - HI^M_{(r,b,c)} \right) , \tag{5}$$

where $HI_{(r,c,b)}$ denotes the sample of HI located in the bth band at the rth row and the cth column. Notice that H denotes the number of rows and W denotes the number of columns, respectively.

The PSNR value approaches infinity as the MSE approaches zero; this shows that a higher PSNR value provides a higher image quality. At the other end of the scale, a small value of the PSNR implies high numerical differences between the compared images.

4.1.2. Signal-to-Noise Ratio (SNR)

The Signal-to-Noise Ratio (SNR) [42] is characterized by Eq. (6),

$$SNR = 10 \log_{10} \frac{E_X}{E_N} , \tag{6}$$

where E_X and E_N denote the energy of the original signal and of the reconstruction error, respectively. In detail, we denote by $x_{i,j,k}$ the original image and by $\hat{x}_{i,j,k}$ the reconstructed image, where the indexes i, j, and k characterize the lines, pixels and bands. Then, we have $E_X = \sum_{i,j,k} x^2_{i,j,k}$, and $E_N = \sum_{i,j,k} (x_{i,j,k} - \hat{x}_{i,j,k})^2$. Notice that the energy E_X depends on the mean value of the signal. It is important to emphasize that hyperspectral images generally have a nonzero mean value, since in the transform domain the mean value is often caught by one transform coefficient (or by a small set of coefficients). As reported in [42], the consequence is that it makes sense to consider energy instead of variance, since energy-preserving transforms or approximations thereof are used, which also represent the image mean value.

4.1.3. Structural similarity (SSIM)

The SSIM is a well-established quality metric used to measure the similarity between two images. Such a metric has been proposed by Wang et al. [43] and it is related to the quality perception of the *human visual system (HVS)*. The SSIM enables to monitor the distortion of any image through a combination of three factors: *loss of correlation*, *luminance distortion* and *contrast distortion*. The SSIM is well-suited to measure the fidelity of signals and it exploits the assumption that natural images have highly structured signals with strong neighborhood dependencies. We remark that such dependencies carry useful information about the structures of the objects in the visual scene.

Let $\mathbf{x}$ and $\mathbf{y}$ be two nonnegative image signals. Assuming that one of the signals has perfect quality, the similarity measure is a quantitative measurement of the quality of the second signal. In detail, the task of similarity measurement is carried out by three comparison functions: *luminance*, *contrast* and *structure*. Notice that the three components are relatively independent, e.g., a change in luminance and/or contrast does not affect the structures of images.

The luminance comparison function is defined through Eq. (7),

$$l(\mathbf{x}, \mathbf{y}) = \frac{2\mu_x \mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} , \tag{7}$$

where

$$\mu_x = \frac{1}{N} \sum_{i=1}^{N} x_i$$

and the constant C_1 enables to avoid instability when $\mu_x^2 + \mu_y^2$ is very close to zero. In detail, C_1 is characterized by Eq. (8),

$$C_1 = (K_1 L)^2 , \tag{8}$$

where L is the dynamic range of the pixel values (255 for 8-bit grayscale images), and $K_1 \ll 1$ is a small constant. Similar considerations also apply to contrast comparison and structure comparison described later.

The contrast comparison function is given by Eq. (9),

$$c(\mathbf{x}, \mathbf{y}) = \frac{2\sigma_x \sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} , \tag{9}$$

where

$$\sigma_x = \left(\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu_x)^2 \right)^{\frac{1}{2}}$$

Algorithm 5: The pseudo-code of the *embedAndCompress* procedure.

Input:

- **HI**: Input Hyperspectral Image
- **P**: Payload to hide in **HI**
- **K**: Key for the encryption of **P**

Output:

- α: Auxiliary Information

```
procedure EMBEDANDCOMPRESS(HI, P, K)
    Allocate the necessary memory space for HI^M;
    Encrypt the payload P with the key K and store the result in P^E;
    Initialize the set α to be empty, i.e., α = ∅;
    Initialize the global variable i to 1, i.e., i = 1;
    for r = 1 to HI_R do
        for c = 1 to HI_C do
            for b = 1 to HI_B do
                Let x be the sample at the (r, c) coordinates of the bth band of HI;
                Let x^M be the sample at the (r, c) coordinates of the bth band of HI^M;
                Let β be a set defined as β = {b_j | ∀(r_j, c_j, b_j) ∈ α};
                x̂^M = prediction of x^M using the prediction context obtained from HI^M;
                e = x − x̂^M
                if r == 1 or c == 1 then
                    sendPixelValue(x);
                    Store x in HI^M and continue with the next iteration;
                end if
                if b ∈ β then
                    mapAndSendError(e);
                    Store x in HI^M and continue with the next iteration;
                end if
                if x == HI_min or x == HI_max then
                    mapAndSendError(e);
                    if not all the bits of P^E are embedded then
                        α = α ∪ {(r, c, b)};
                    end if
                    Store x in HI^M and continue with the next iteration;
                end if
                if not all the bits of P^E are embedded then
                    e_N = calculateNewErrorValue(P_i^E, e);
                else
                    e_N = e;
                end if
                mapAndSendError(e_N);
                x^M = x̂^M + e_N;
                Store x^M in HI^M;
            end for
        end for
    end for
    return α;
end procedure
```

and $C_2 = (K_2L)^2$, $K_2 \ll 1$. An important feature of this function is that with the same amount of contrast change $\triangle\sigma = \sigma_y - \sigma_x$, this measure is less sensitive to the case of high base contrast σ_x than low base contrast. This is consistent with the contrast-masking feature of the HVS.

Algorithm 6: The pseudo-code of the *decompressExtractAndReconstruct* procedure.

Input:

- **K**: Key for the decryption of the reconstructed payload $\mathbf{P^R}$
- α: Auxiliary Information

Output:

- $\mathbf{HI^R}$: Reconstructed and Decompressed Hyperspectral Image
- $\mathbf{P^R}$: Reconstructed and decrypted payload

```
procedure DECOMPRESSEXTRACTANDRECONSTRUCT(K, α)
    Allocate the necessary memory space for HI^R and for HI^M;
    Initialize the set α and the set β to be empty, i.e., α = ∅ and β = ∅;
    Initialize the global variable i to 1, i.e., i = 1;
    for r = 1 to HI_R^M do
        for c = 1 to HI_C^M do
            for b = 1 to HI_B^M do
                if r == 1 or c == 1 then
                    x = receivePixelValue();
                    Store x in HI^R;
                    Store x in HI^M;
                    Continue with the next iteration;
                end if
                x̂^M = prediction of x^M using the prediction context obtained from HI^M;
                e_M = receiveAndUnmapError();
                if all the bits of P^E are extracted or b ∈ β or (r, c, b) ∈ α then
                    x = x̂^M + e_M;
                    Store x in HI^R;
                    Store x in HI^M;
                    if (r, c, b) ∈ α then
                        β = β ∪ {b};
                    end if
                    Continue with the next iteration;
                end if
                e_R = reconstructErrorValue(e_M, P^E);
                x^M = x̂^M + e_M;
                x^R = x̂^M + e_R;
                Store x^M in HI^M;
                Store x^R in HI^R;
            end for
        end for
    end for
    Decrypt the extracted payload P^E with the key K and store the result in P^R;
    return HI^R and P^R;
end procedure
```

Structure comparison is performed after luminance subtraction and variance normalization. More precisely, we associate the two unit vectors $(\mathbf{x} - \mu_x)/\sigma_x$ and $(\mathbf{y} - \mu_y)/\sigma_y$, each lying in the hyperplane defined by (3), to the structure of the two images. The correlation (inner product) between such two vectors is a simple and effective measure to quantify the structural similarity. Notice that the correlation between $(\mathbf{x} - \mu_x)/\sigma_x$ and $(\mathbf{y} - \mu_y)/\sigma_y$ is equivalent to the correlation coefficient between $\mathbf{x}$ and $\mathbf{y}$. The structure comparison function is given by Eq. (10),

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \ . \tag{10}$$

Notice that as in the luminance and contrast measures, a small constant has been introduced in both denominator and numerator. In the discrete form, σ_{xy} can be estimated as

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y) \ . \tag{11}$$

Finally, the three comparison functions represented by Eqs. (7), (9) and (10) are combined to create the SSIM index between signals x and y

$$SSIM(\mathbf{x}, \mathbf{y}) = [l(\mathbf{x}, \mathbf{y})]^{\alpha} \cdot [c(\mathbf{x}, \mathbf{y})]^{\beta} \cdot [s(\mathbf{x}, \mathbf{y})]^{\gamma} , \tag{12}$$

where $\alpha > 0$, $\beta > 0$ and $\gamma > 0$ are parameters used to adjust the relative importance of the three components. In order to simplify the expression, we set $\alpha = \beta = \gamma = 1$ and $C_3 = C_2/2$ in this paper. This results in a specific form of the SSIM index

$$SSIM(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \ . \tag{13}$$

4.1.4. Quality index (Q-Index)

It is a mathematically defined universal image quality index [41]. The Q-Index is referred to be "universal", since the quality measurement approach does not depend on the image under evaluation, the viewing conditions or the individual observers. Such index is applicable to several image processing applications and provides meaningful comparison among different types of image distortions.

The Q-Index is defined by means of Eq. (14)

$$Q = \frac{4\sigma_{xy}\bar{x}\bar{y}}{(\sigma_x^2 + \sigma_y^2)[(\bar{x})^2 + (\bar{y})^2]} \ , \tag{14}$$

where

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \ ,$$

$$\sigma_x^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2, \qquad \sigma_y^2 = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \bar{y})^2 \ ,$$

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y}) \ .$$

The value of Q lies within the range $[-1, 1]$. The best value for Q is 1 and it is achieved if and only if $y_i = x_i$ for all $i = 1, 2, \ldots, N$, hence if the compared images are exactly the same. This quality index characterizes any form of distortion as a combination of three different factors, namely, *loss of correlation*, *luminance distortion* and *contrast distortion*. Therefore, the definition of Q-Index can be rewritten as a product of three components,

$$Q = \frac{\sigma_{xy}}{\sigma_x\sigma_y} \cdot \frac{2\bar{x}\bar{y}}{(\bar{x})^2 + (\bar{y})^2} \cdot \frac{2\sigma_x\sigma_y}{\sigma_x^2 + \sigma_y^2} \ , \tag{15}$$

where the first component is the correlation coefficient between $\mathbf{x}$ and $\mathbf{y}$, which measures the degree of linear correlation between $\mathbf{x}$ and $\mathbf{y}$, and its dynamic range is $[-1, 1]$. The best value is 1 and it is obtained when $y_i = ax_i + b$ for all $i = 1, 2, \ldots, N$, where a and b are constants and $\alpha > 0$. The second component, with a value in the range $[0, 1]$, measures how close the mean luminance is between $\mathbf{x}$ and $\mathbf{y}$. It equals 1 if and only if $\bar{x} = \bar{y}$. σ_x and σ_y can be viewed as an estimate of the contrast of $\mathbf{x}$ and $\mathbf{y}$, so the third component measures how similar the contrasts of the images are. Its range of values is also $[0, 1]$, where the best value is 1 and it is achieved if and only if $\sigma_x = \sigma_y$.

Table 1
Dataset description.

Hyperspectral images	Columns	Lines	Bands
Moffett field	753	1924	224
Cuprite	754	2776	224
Lunar lake	781	6955	224
Low altitude	614	1087	224

4.2. Testing environment and testbed

We designed and tested a working prototype of our proposal, in order to assess its effectiveness. The testing activity has been performed on four hyperspectral images belonging to the *Airborne Visible/Infrared Imaging Spectrometer (AVIRIS)* public dataset, provided by the *Jet Propulsion Laboratory (JPL)* of the *NASA*. In Table 1 we report the dimensions of each tested image, outlined in rows from the second to the fifth. In detail, the second, the third and the fourth column denote the number of columns, rows and bands of each image, respectively.

It is important to point out that each sample is stored as a 16-bits signed integer. Thus, a sample can assume a value in the range $[-2^{15}, 2^{15} - 1]$. For such a reason, to avoid underflow and overflow issues, we set $HI_{min} = -2^{15}$ and $HI_{max} = 2^{15} - 1$, respectively.

We use the following data payloads:

- **Data Payload 1** (D_1)
  - *Description*: Composed of all bit values equal to 1;
  - *Size*: Covers the whole *Embedding Capacity* (EC);
- **Data Payload 2** (D_2)
  - *Description*: Pseudo-random generated;
  - *Size*: Covers the whole *Embedding Capacity* (EC);
- **Data Payload 3** (D_3)
  - *Description*: Pseudo-random generated;
  - *Size*: 6000000 bits (732 KB);

Notice that in all the performed experiments, we do not encrypt the data payloads and we use an arithmetic coder as entropy coder. Again, we consider three predictors: the *Median Edge Detector (MED)* predictor, the *Linear Predictor* (LP) [44] and the *3D-MultiBand Linear Predictor* (3D-MBLP) [4]. As done in [44], in all the experiments we process the first 8 bands of each image by using the MED predictor. We remark that the 3D-MBLP is a configurable predictor, in which it is possible to set two parameters, denoted as N and B, where N denotes the number of samples for the current band and for each of the B previous bands involved in the prediction. In detail, in all the experiments, we use the following four configurations:

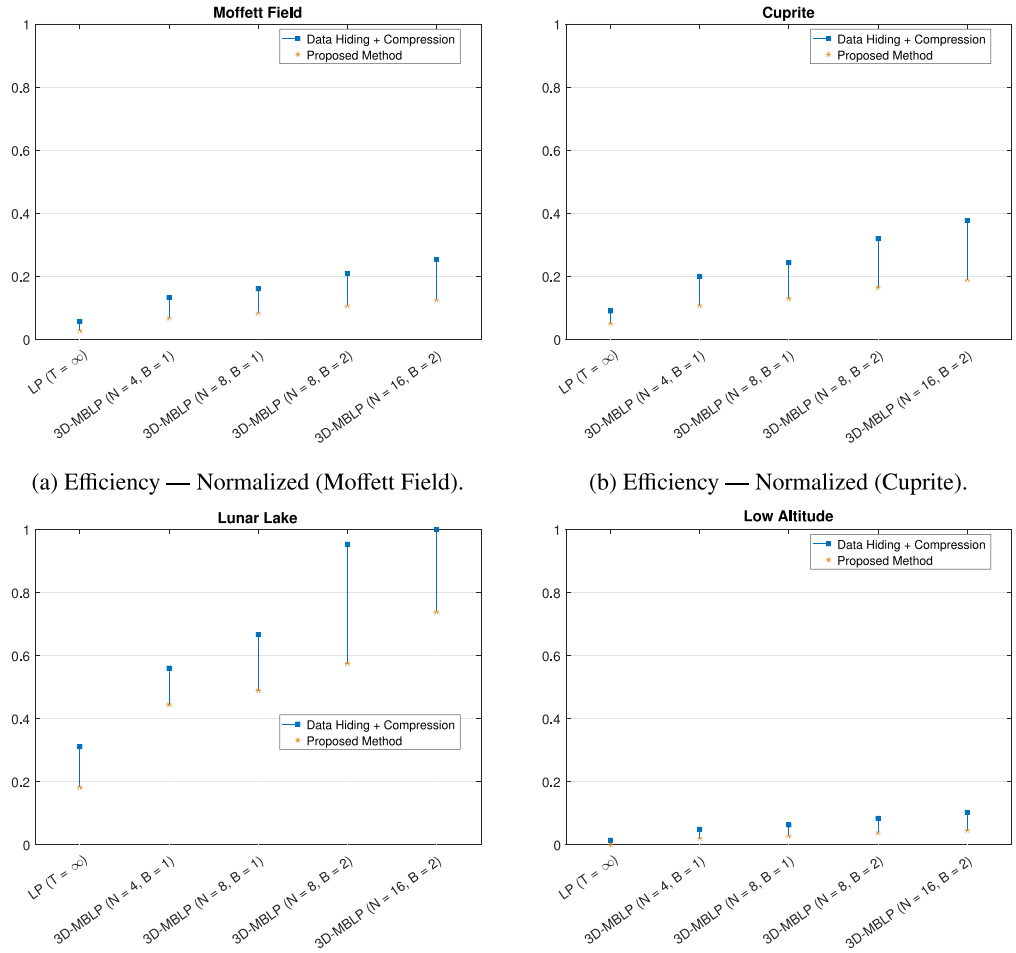
- $N = 4, B = 1$;
- $N = 8, B = 1$;
- $N = 8, B = 2$;
- $N = 16, B = 2$.

In order characterize the computational costs related to the aforementioned predictors, in Table 2 we report an estimation of the number of floating-point operations needed to predict a sample of a hyperspectral image.

4.2.1. Efficiency and compression performance

With the aim of evaluating our proposal in terms of time efficiency, we have monitored the execution time of the following two scenarios:

- **Scenario 1** — *Execution of Data Hiding + Compression*
  - Execution of a proof of concept implementing the *embed* procedure reported in Algorithm 1;



(a) Efficiency — Normalized (Moffett Field).

(b) Efficiency — Normalized (Cuprite).

(c) Efficiency — Normalized (Lunar Lake).

(d) Efficiency — Normalized (Low Altitude).

Fig. 5. Efficiency — Normalized [0, 1] (Graphs).

- Execution of a proof of concept implementing a predictive-based lossless compression procedure (from now on, referred to as the *compression* algorithm), similar to the one shown in Fig. 4b;

- **Scenario 2** — *Execution of Our Proposal*
  - Execution of a proof of concept implementing the *embedAndCompress* procedure reported in Algorithm 5.

More precisely, we evaluate the execution time of the *embedAndCompress* procedure (Scenario 2), as well as the time required to complete the execution of both the *embed* procedure and the *compression* algorithm (Scenario 1). In the Scenario 1, the hyperspectral image *HI* is processed by the *embed* procedure and the output of such a procedure, denoted as HI^M, became the input of the *compression* algorithm. It is important to point out that the results (images) produced by Scenario 1 and Scenario 2 are equivalent. We remark that for the monitoring of the execution time, we have not taken into account the time needed for the access to the physical support (i.e., reading/writing operations), as well as the execution time of the arithmetic coder used for the *embedAndCompress* procedure and the lossless compression algorithm.

Table 3 and Fig. 5 summarize the normalized execution times (in the range [0, 1]), on all the tested images (columns from the second to the fifth), concerning the Scenario 1 (*Data Hiding + Compression*) and the Scenario 2 (*Proposed*). Recall that we considered the LP and the 3D-MBLP predictors (rows from the second to the sixth). We also report the percentage change achieved by comparing the execution time of the Scenario 1 with respect to the one of the Scenario 2. We monitored the executions by using the same parameters for both the scenarios and we used the data payload D_2 for the embedding.

Table 2
# of floating-point operations per sample.

	# of f.p. operations p.s.
MED Predictor (*baseline*)	∼2
LP ($T = \infty$)	∼6
3D-MBLP ($N = 4, B = 1$)	∼5
3D-MBLP ($N = 8, B = 1$)	∼9
3D-MBLP ($N = 8, B = 2$)	∼35
3D-MBLP ($N = 16, B = 2$)	∼67

4.2.2. Capacity

In Table 4 we report an estimation of the *Embedding Capacity* (EC) achieved by our proposal, by considering the LP and the 3D-MBLP predictors (rows from the second to the seventh), for each evaluated hyperspectral image (columns from the second to the fifth). More precisely, we have obtained the estimation of the EC by simulating the embedding of a data payload composed of all the bits equal to 1. The second row of Table 4 reports the EC estimated by using the bi-dimensional MED predictor. The estimated EC, achieved through the MED predictor, is also used as baseline and it is compared with respect to the ECs obtained by using the other

Table 3
Execution times — Normalized in the range [0, 1].

	Moffett field	*Cuprite*	*Lunar lake*	*Low altitude*
LP ($T = \infty$)				
Data Hiding + Compression	0.0566	0.0897	0.3110	0.0141
Proposed	0.0260	0.0487	0.1809	0.0000
Percent change	−39.40%	−37.07%	−39.18%	−40.28%
3D-MBLP ($N = 4, B = 1$)				
Data Hiding + Compression	0.1320	0.1997	0.5600	0.0488
Proposed	0.0660	0.1057	0.4434	0.0186
Percent change	−43.12%	−42.61%	−20.08%	−43.23%
3D-MBLP ($N = 8, B = 1$)				
Data Hiding + Compression	0.1602	0.2431	0.6661	0.0625
Proposed	0.0812	0.1282	0.4883	0.0258
Percent change	−43.62%	−43.52%	−25.87%	−43.92%
3D-MBLP ($N = 8, B = 2$)				
Data Hiding + Compression	0.2089	0.3190	0.9532	0.0836
Proposed	0.1041	0.1630	0.5735	0.0362
Percent change	−45.62%	−45.89%	−38.98%	−45.37%
3D-MBLP ($N = 16, B = 2$)				
Data Hiding + Compression	0.2535	0.3765	1.0000	0.1025
Proposed	0.1225	0.1870	0.7374	0.0439
Percent change	−47.73%	−47.68%	−25.72%	−47.43%

Table 4
Embedding capacity (in bits).

	Moffett field	*Cuprite*	*Lunar lake*	*Low altitude*
MED predictor (*baseline*)	40943761	65597022	169888412	6448934
LP ($T = \infty$)	43708046	64614107	179203290	12173558
Percentage change	*+6.75%*	*−1.5%*	*+5.48%*	*+88.77%*
3D-MBLP ($N = 4, B = 1$)	57574404	78080592	221298199	14886599
Percentage change	*+40.62%*	*+19.03%*	*+30.26%*	*+130.84%*
3D-MBLP ($N = 8, B = 1$)	61009543	82695892	234315111	15666995
Percentage Change	*+49.01%*	*+26.07%*	*+37.92%*	*+142.94%*
3D-MBLP ($N = 8, B = 2$)	71022431	97211303	273799861	16544623
Percentage Change	*+73.46%*	*+48.19%*	*+61.16%*	*+156.55%*
3D-MBLP ($N = 16, B = 2$)	74412161	100957897	286086788	17463812
Percentage Change	*+81.74%*	*+53.91%*	*+68.4%*	*+170.8%*

Table 5
Embedding rate (average).

	Moffett field	*Cuprite*	*Lunar lake*	*Low altitude*
MED Predictor (*baseline*)	0.1262	0.1399	0.1396	0.0431
LP ($T = \infty$)	0.1347	0.1378	0.1473	0.0814
3D-MBLP ($N = 4, B = 1$)	0.1774	0.1665	0.1819	0.0996
3D-MBLP ($N = 8, B = 1$)	0.1880	0.1764	0.1926	0.1048
3D-MBLP ($N = 8, B = 2$)	0.2189	0.2073	0.2250	0.1107
3D-MBLP ($N = 16, B = 2$)	0.2293	0.2153	0.2351	0.1168

two predictors: LP (reported in the third row) and 3D-MBLP (rows from the fourth to the seventh). Each row from the third to the seventh reports the percentage change, that is, the percentage increase or decrease with respect to the used baseline (i.e., the MED predictor).

In Fig. 6 we show the trend of the EC values (y-axis) achieved for each band (x-axis) of the *Moffett Field*, *Lunar Lake*, *Cuprite* and *Low Altitude* images, respectively.

Furthermore, in Table 5 we report the *Embedding Rate* (ER) of a given hyperspectral image (columns from the second to the fifth), obtained according to Eq. (16)

$$ER = \frac{1}{HI_B} \cdot \sum_{i=1}^{HI_B} \frac{EC_i}{HI_R \cdot HI_C} \ , \tag{16}$$

where EC_i is the EC of the ith band.

4.2.3. Imperceptibility performances

In this section we evaluate the imperceptibility of the proposed scheme. In detail, we invoke the *embedAndCompress* procedure on the hyperspectral image HI and then decompress the output of such a procedure, without extracting the hidden data. Notice that this scenario is equivalent to invoke the *embed* procedure on the input HI. The obtained output, namely, the marked hyperspectral image HI^M, is then compared with the unaltered hyperspectral image HI, by using the following imperceptibility metrics: *SSIM*, *Q-Index*, *SNR* and *PSNR* (described in Section 4.1).

In Tables 6–8 we report the results achieved through the aforementioned metrics on all the tested hyperspectral images (columns from the second to the fifth), by using the LP and the 3D-MBLP predictors (rows from the second to the sixth) and by considering the data payload D_1 (Table 6), D_2 (Table 7) and D_3 (Table 8).

We stress that the value achieved by a given metric is obtained as follows. Let N be the number of bands of both the cover image HI and marked image HI^M. A given metric is first computed on each pair of bands, namely, the ith band of the cover image and the ith band of the marked image, where $1 \leq i \leq N$. After the values of a given metric has been calculated for each pair of bands, the average of such values is calculated and becomes the final value of the metric (reported in the appropriate table). Notice that we have properly adapted the used metrics, in the way described before, since such metrics were designed to work on bi-dimensional images.

In Fig. 7a we show the value of the SSIM metric (y-axis) for each band of the *Low Altitude* image (x-axis), in which D_1 is hidden. Similarly to Fig. 7a, in Fig. 7b we show the trend of the Q-Index metric (y-index) for each band of the *Lunar Lake* image (x-axis), in which D_1 is hidden. Fig. 7c and Fig. 7d show the trend of the SNR and the PSNR metrics, respectively, for each band of the *Cuprite* and *Moffett Field* images. Notice that in both images the data payload D_2 is embedded.

From the two graphs in Fig. 8, reported on the left, it is possible to observe the slight differences between the histograms of the

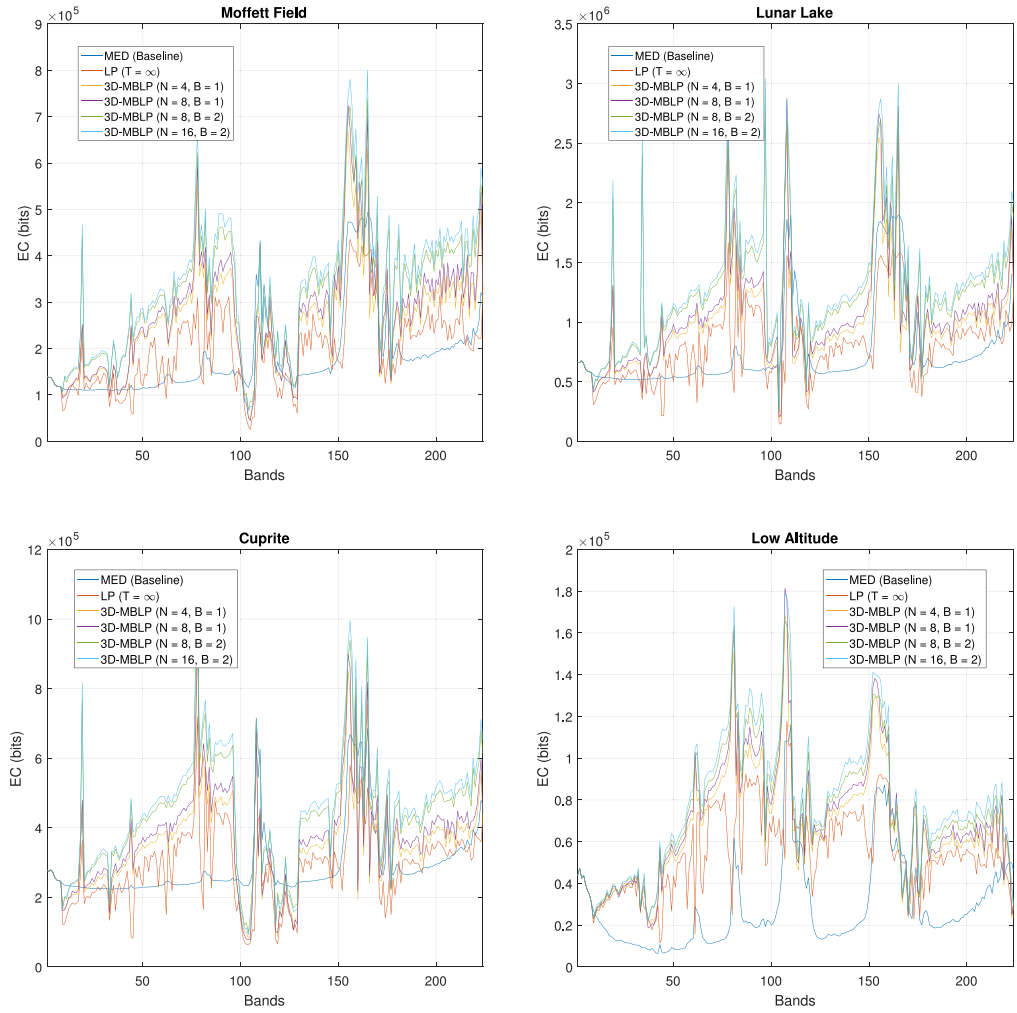


Fig. 6. Band-by-Band Capacity.

Table 6
Imperceptibility Performances by considering the data payload D_1.

	Moffett field	*Cuprite*	*Lunar lake*	*Low altitude*
LP ($T = \infty$)				
SSIM	0.9988	0.9996	0.9950	0.9969
QI	0.9964	0.9999	0.9991	0.9980
SNR	53.73	59.40	56.85	54.32
PSNR	90.32	90.32	90.31	90.32
3D-MBLP ($N = 4, B = 1$)				
SSIM	0.9988	0.9996	0.9921	0.9948
QI	0.9889	0.9999	0.9981	0.9981
SNR	53.73	59.40	56.85	54.32
PSNR	90.32	90.32	90.31	90.32
3D-MBLP ($N = 8, B = 1$)				
SSIM	0.9985	0.9996	0.9915	0.9946
QI	0.9878	0.9999	0.9980	0.9981
SNR	53.73	59.40	56.85	54.32
PSNR	90.32	90.32	90.31	90.32
3D-MBLP ($N = 8, B = 2$)				
SSIM	0.9982	0.9996	0.9931	0.9957
QI	0.9913	0.9999	0.9986	0.9981
SNR	53.73	59.40	56.85	54.32
PSNR	90.32	90.32	90.31	90.32
3D-MBLP ($N = 16, B = 2$)				
SSIM	0.9979	0.9995	0.9927	0.9956
QI	0.9904	0.9999	0.9986	0.9981
SNR	53.73	59.40	56.85	54.32
PSNR	90.32	90.32	90.31	90.32

Table 7
Imperceptibility Performances by considering the data payload D_2.

	Moffett field	*Cuprite*	*Lunar lake*	*Low altitude*
LP ($T = \infty$)				
SSIM	0.9988	0.9996	0.9949	0.9971
QI	0.9963	0.9999	0.9991	0.9982
SNR	54.05	59.73	57.20	54.50
PSNR	90.64	90.64	90.67	90.50
3D-MBLP ($N = 4, B = 1$)				
SSIM	0.9987	0.9996	0.9925	0.9950
QI	0.9902	0.9999	0.9983	0.9982
SNR	54.13	59.78	57.26	54.54
PSNR	90.72	90.70	90.73	90.54
3D-MBLP ($N = 8, B = 1$)				
SSIM	0.9984	0.9996	0.9919	0.9948
QI	0.9891	0.9999	0.9982	0.9983
SNR	54.18	59.82	57.30	54.55
PSNR	90.77	90.74	90.77	90.56
3D-MBLP ($N = 8, B = 2$)				
SSIM	0.9982	0.9996	0.9936	0.9959
QI	0.9924	0.9999	0.9989	0.9983
SNR	54.26	59.90	57.39	54.57
PSNR	90.85	90.81	90.86	90.57
3D-MBLP ($N = 16, B = 2$)				
SSIM	0.9980	0.9996	0.9932	0.9958
QI	0.9918	0.9999	0.9988	0.9983
SNR	54.28	59.91	57.41	54.58
PSNR	90.87	90.83	90.88	90.58

Table 8
Imperceptibility Performances by considering the data payload D_3.

	Moffett field	*Cuprite*	*Lunar lake*	*Low altitude*
LP ($T = \infty$)				
SSIM	*0.9999*	*0.9999*	*0.9998*	*0.9985*
QI	*0.9999*	*0.9999*	*0.9999*	*0.9990*
SNR	*63.75*	*70.43*	*72.47*	*57.57*
PSNR	*100.34*	*101.35*	*105.93*	*93.57*
3D-MBLP ($N = 4, B = 1$)				
SSIM	*0.9999*	*0.9999*	*0.9998*	*0.9979*
QI	*0.9997*	*0.9999*	*0.9999*	*0.9992*
SNR	*64.49*	*71.09*	*73.01*	*58.50*
PSNR	*101.07*	*102.01*	*106.47*	*94.50*
3D-MBLP ($N = 8, B = 1$)				
SSIM	*0.9999*	*0.9999*	*0.9998*	*0.9979*
QI	*0.9997*	*0.9999*	*0.9999*	*0.9993*
SNR	*65.03*	*71.63*	*73.68*	*58.73*
PSNR	*101.62*	*102.55*	*107.14*	*94.74*
3D-MBLP ($N = 8, B = 2$)				
SSIM	*0.9999*	*0.9999*	*0.9999*	*0.9984*
QI	*0.9998*	*0.9999*	*0.9999*	*0.9993*
SNR	*65.83*	*72.42*	*74.51*	*58.97*
PSNR	*102.42*	*103.34*	*107.98*	*94.98*
3D-MBLP ($N = 16, B = 2$)				
SSIM	*0.9999*	*0.9999*	*0.9999*	*0.9985*
QI	*0.9998*	*0.9999*	*0.9999*	*0.9993*
SNR	*66.06*	*72.60*	*74.74*	*59.22*
PSNR	*102.65*	*103.52*	*108.20*	*95.22*

unaltered *Low Altitude* image and the ones that embed the data payloads D_1 and D_2, respectively. Moreover, in the two graphs in Fig. 8, reported on the right, a zoomed portion of such histograms is shown.

4.2.4. Compression performances

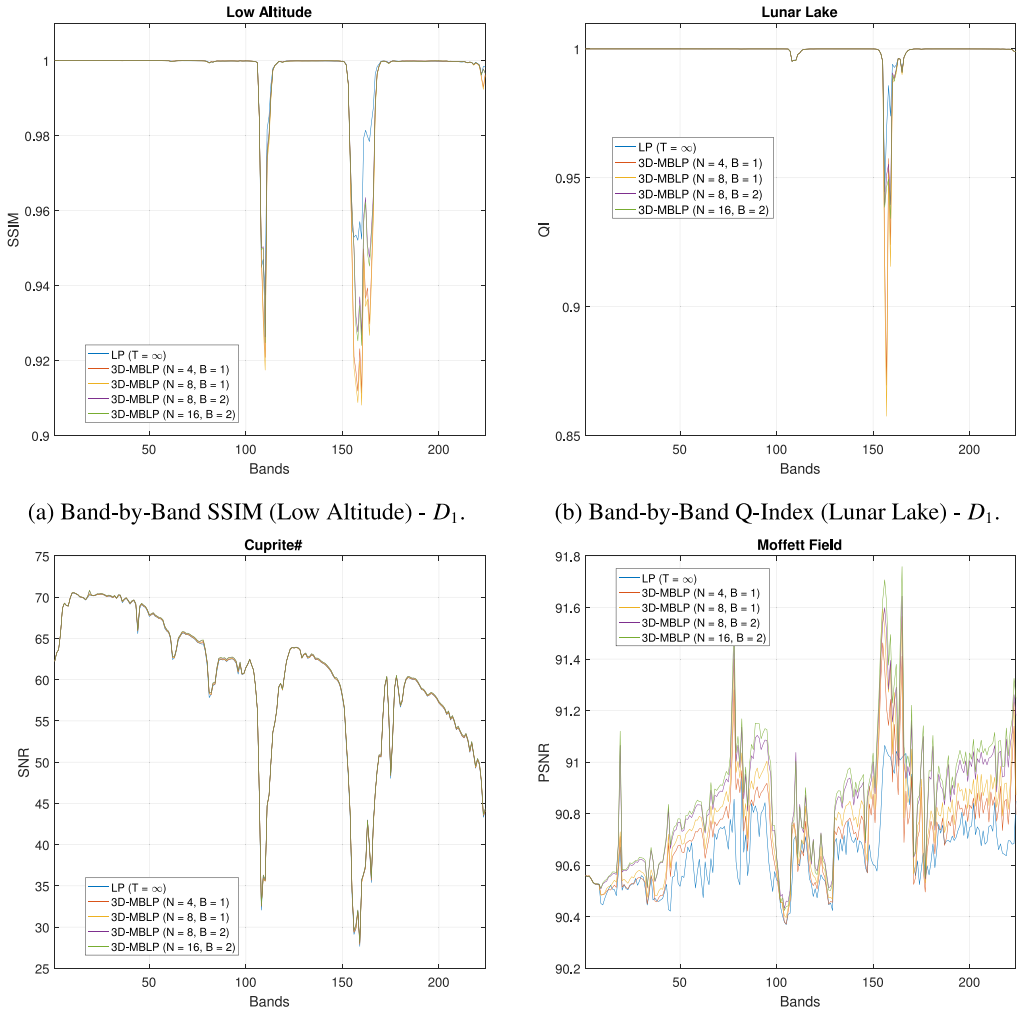
This section highlights the compression performances of the proposed scheme, by exploring how the hiding of a data payload impacts on the compression results. More precisely, we compare the results, in terms of *Bit-Per-Sample (BPS)*, achieved by the compression of each hyperspectral image through the *compression* algorithm, with respect to the results achieved by using our proposal, when the data payloads are embedded, i.e., D_1, D_2 and D_3.

In Table 9 we report the results achieved by comparing the two aforementioned approaches. In detail, in such a table we report the results (in terms of BPS) concerning the pure compression algorithm, as well as those achieved by our proposal (reported in parenthesis), along with the relative percentage change. The results reported in such a table take into account the three data payload, i.e., D_1, D_2 and D_3 for each image (columns from the second to the fifth), besides considering the LP predictor (second row) and the 3D-MBLP predictors (rows from the third to the sixth).

5. Conclusions and future works

In this work we presented a novel *one-pass* framework suitable for hyperspectral images collected through remote sensing facilities. The proposed framework allows, at the same time, the hiding of a data payload and the compression of the resulting marked stream. In particular, the proposed data hiding strategy is based on the *modification of prediction errors (MPE)* technique [36].

Experimental results show that the above framework is able to achieve significant performance improvements in terms of execution time, with respect to the solution which performs data hiding



(a) Band-by-Band SSIM (Low Altitude) - D_1.

(b) Band-by-Band Q-Index (Lunar Lake) - D_1.

(c) Band-by-Band SNR (Cuprite) - D_2.

(d) Band-by-Band PSNR (Moffett Field) - D_2.

Fig. 7. Example of Band-by-Band metric values trends.

Table 9
Compression Performance: BPS — Bits Per Sample.

	Moffett field	*Cuprite*	*Lunar lake*	*Low altitude*
LP ($T = \infty$)	5.3883	4.9917	4.6091	5.5457
Percent Change - D_1 (BPS)	−2.54% (*5.5289*)	−3.02% (*5.1470*)	−3.37% (*4.7699*)	−2.36% (*5.6800*)
Percent Change - D_2 (BPS)	−4.99% (*5.6713*)	−5.68% (*5.2924*)	−6.38% (*4.9232*)	−3.71% (*5.7595*)
Percent Change - D_3 (BPS)	*−0.95% (5.4401)*	*−0.80% (5.0320)*	*−0.47% (4.6309)*	*−2.26% (5.6741)*
3D-MBLP ($N = 4, B = 1$)	4.7420	4.5834	4.1920	5.1420
Percent Change - D_1 (BPS)	−3.44% (4.9109)	−4.43% (4.7960)	−4.98% (4.4119)	−3.15% (5.3094)
Percent Change - D_2 (BPS)	−6.69% (5.0821)	−7.55% (4.9579)	−8.64% (4.5887)	−4.89% (5.4066)
Percent Change - D_3 (BPS)	*−1.10% (4.7947)*	*−0.94% (4.6268)*	*−0,54% (4.2150)*	*−2.56% (5.2773)*
3D-MBLP ($N = 8, B = 1$)	4.7747	4.5937	4.1723	5.1043
Percent Change - D_1 (BPS)	−2.61% (4.9028)	−3.64% (4.7674)	−4.16% (4.3533)	−2.81% (5.2519)
Percent Change - D_2 (BPS)	−6.26% (5.0934)	−7.14% (4.9470)	−8.26% (4.5478)	−4.67% (5.3544)
Percent Change - D_3 (BPS)	*−0.96% (4.8211)*	*−0.83% (4.6323)*	*−0.50% (4.1932)*	*−2.39% (5.2295)*
3D-MBLP ($N = 8, B = 2$)	4.5609	4.5334	4.0587	5.0484
Percent Change - D_1 (BPS)	−3.68% (4.7350)	−3.32% (4.6891)	−4.08% (4.2312)	−2.58% (5.1822)
Percent Change - D_2 (BPS)	−7.87% (4.9503)	−7.39% (4.8953)	−8.88% (4.4541)	−4.58% (5.2909)
Percent Change - D_3 (BPS)	*−1.00% (4.6072)*	*−0.77% (4.5685)*	*−0.49% (4.0788)*	*−2.30% (5.1670)*
3D-MBLP ($N = 16, B = 2$)	4.5183	4.4912	3.9983	4.9785
Percent Change - D_1 (BPS)	−3.12% (4.6638)	−2.77% (4.6191)	−3.43% (4.1401)	−2.34% (5.0976)
Percent Change - D_2 (BPS)	−7.60% (4.8898)	−7.07% (4.8330)	−8.56% (4.3728)	−4.49% (5.2124)
Percent Change - D_3 (BPS)	*−0.95% (4.5618)*	*−0.73% (4.5242)*	*−0.48% (4.0176)*	*−2.20% (5.0906)*

and then compression by means of two separate stages. Moreover, testing activity carried out to assess the performance of our proposal in terms of imperceptibility, shows that the cover image and the stego image are extremely similar, since the hidden data results to be imperceptible. Finally, when dealing with images containing an embedded data payload, the compression performance of our proposal does not show a significant decrease, if compared to the performance achieved by compressing images not affected by data embedding.

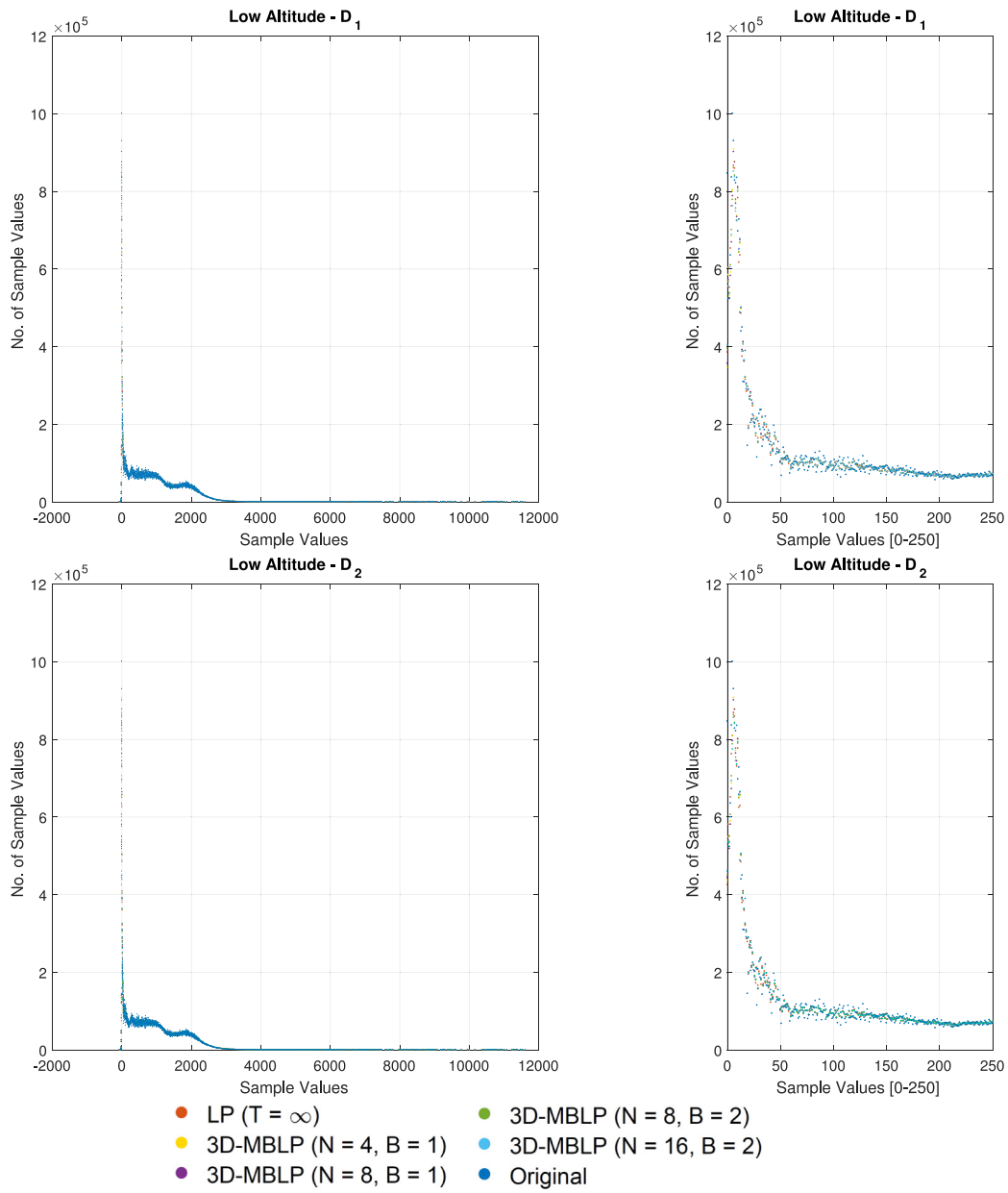


Fig. 8. Histograms of the *Low Altitude* image and the marked ones, produced by our scheme by considering the LP and the 3D-MBLP predictors and the data payloads D_1 and D_2.

Future research activity can follow several different directions. In detail, to further improve the execution time, we intend to explore the possibility of a parallel design of our proposal [45]. Moreover, we intend to explore the possibility of extending our framework to consider also the lossy compression. Finally, in the case of a textual-based payload, we intend to perform further testing activity, by using text compression and encryption approaches on the payload itself, as for example the approach discussed in [46]. In particular, we believe that the use of text compression should minimize the representation of the data, hence minimizing the distortion when the compressed payload will be hidden.

References

[1] D. Manolakis, G. Shaw, Detection algorithms for hyperspectral imaging applications, IEEE Signal Process. Mag. 19 (1) (2002) 29–43.

[2] L. Wang, J. Zhang, P. Liu, K.R. Choo, F. Huang, Spectral-spatial multi-feature-based deep learning for hyperspectral remote sensing image classification, Soft Comput. 21 (1) (2017) 213–221.

[3] P. Liu, K.R. Choo, L. Wang, F. Huang, SVM or deep learning? A comparative study on remote sensing image classification, Soft Comput. 21 (23) (2017) 7053–7065.

[4] R. Pizzolante, B. Carpentieri, Multiband and lossless compression of hyperspectral images, Algorithms 9 (1) (2016) 16.

[5] R. Pizzolante, B. Carpentieri, Visualization, band ordering and compression of hyperspectral images, Algorithms 5 (1) (2012) 76–97.

[6] Z. Liu, K.R. Choo, J. Großschädl, Securing edge devices in the post-quantum internet of things using lattice-based cryptography, IEEE Commun. Mag. 56 (2) (2018) 158–162.

[7] C. Lin, D. He, N. Kumar, K.R. Choo, A. Vinel, X. Huang, Security and privacy for the internet of drones: Challenges and solutions, IEEE Commun. Mag. 56 (1) (2018) 64–69.

[8] C.J. D'Orazio, K.R. Choo, Circumventing ios security mechanisms for APT forensic investigations: A security taxonomy for cloud apps, Future Gener. Comput. Syst. 79 (2018) 247–261.

[9] C. D'Orazio, K.R. Choo, An adversary model to evaluate DRM protection of video contents on ios devices, Comput. Secur. 56 (2016) 94–110.

[10] P. Albano, A. Bruno, B. Carpentieri, A. Castiglione, A. Castiglione, F. Palmieri, R. Pizzolante, I. You, A secure distributed video surveillance system based on portable devices, in: International Conference on Availability, Reliability, and Security, Springer, 2012, pp. 403–415.
[11] P. Albano, A. Bruno, B. Carpentieri, A. Castiglione, A. Castiglione, F. Palmieri, R. Pizzolante, K. Yim, I. You, Secure and distributed video surveillance via portable devices, J. Ambient Intell. Humaniz. Comput. 5 (2) (2014) 205–213.
[12] R. Pizzolante, B. Carpentieri, A. Castiglione, G. D. Maio, The avq algorithm: watermarking and compression performances, in: Intelligent Networking and Collaborative Systems, INCoS, 2011 Third International Conference on, IEEE, 2011, pp. 698–702.
[13] R. Pizzolante, B. Carpentieri, A. Castiglione, A secure low complexity approach for compression and transmission of 3-D medical images, in: 2013 Eighth International Conference on Broadband and Wireless Computing, Communication and Applications, Compiegne, France, October 28-30, 2013, IEEE, 2013, pp. 387–392.
[14] R. Pizzolante, A. Castiglione, B. Carpentieri, A. De Santis, A. Castiglione, Protection of microscopy images through digital watermarking techniques, in: F. Xhafa, L. Barolli, F. Palmieri, M. Koeppen, V. Loia (Eds.), 2014 International Conference on Intelligent Networking and Collaborative Systems, Salerno, Italy, September 10–12, 2014, IEEE, 2014, pp. 65–72.
[15] C. Qin, C.C. Chang, Y.P. Chiu, A novel joint data-hiding and compression scheme based on SMVQ and image inpainting, IEEE Trans. Image Process. 23 (3) (2014) 969–978.
[16] B. Ou, X. Li, Y. Zhao, R. Ni, Y.Q. Shi, Pairwise prediction-error expansion for efficient reversible data hiding, IEEE Trans. Image Process. 22 (12) (2013) 5010–5021.
[17] X. Li, B. Li, B. Yang, T. Zeng, General framework to histogram-shifting-based reversible data hiding, IEEE Trans. Image Process. 22 (6) (2013) 2181–2191.
[18] A. Castiglione, A. De Santis, R. Pizzolante, A. Castiglione, V. Loia, F. Palmieri, On the Protection of fMRI Images in Multi-domain Environments, in: L. Barolli, M. Takizawa, F. Xhafa, T. Enokido, J.H. Park (Eds.), 29th IEEE International Conference on Advanced Information Networking and Applications, AINA 2015, Gwangju, South Korea, March 24-27, 2015, IEEE Computer Society, 2015, pp. 476–481.
[19] R. Pizzolante, A. Castiglione, B. Carpentieri, A. De Santis, F. Palmieri, A. Castiglione, Format-independent protection of DNA microarray images, in: F. Xhafa, L. Barolli, F. Messina, M.R. Ogiela (Eds.), 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2015, Krakow, Poland, November 4–6, 2015, IEEE Computer Society, 2015, pp. 351–357.
[20] R. Pizzolante, A. Castiglione, B. Carpentieri, A. De Santis, A. Castiglione, Reversible copyright protection for DNA microarray images, in: 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2015, Krakow, Poland, November 4–6, 2015, 2015, pp. 707–712.
[21] H. Luo, F.-X. Yu, H. Chen, Z.-L. Huang, H. Li, P.-H. Wang, Reversible data hiding based on block median preservation, Information sciences 181 (2) (2011) 308–328.
[22] J.-X. Wang, Z.-M. Lu, A path optional lossless data hiding scheme based on vq joint neighboring coding, Inform. Sci. 179 (19) (2009) 3332–3348.
[23] X. Wang, J. Ding, Q. Pei, A novel reversible image data hiding scheme based on pixel value ordering and dynamic pixel block partition, Inf. Sci. 310 (2015) 16–35.
[24] W.-J. Yang, K.-L. Chung, H.-Y.M. Liao, Efficient reversible data hiding for color filter array images, Inform. Sci. 190 (2012) 208–226.
[25] R. Pizzolante, A. Castiglione, B. Carpentieri, A. De Santis, F. Palmieri, A. Castiglione, On the protection of consumer genomic data in the internet of living things, Comput. Secur. 74 (2018) 384–400.
[26] M.R. Ogiela, L. Ogiela, U. Ogiela, Biometric methods for advanced strategic data sharing protocols, in: 9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2015, Santa Cantarina, Brazil, July 8–10, 2015, 2015, pp. 179–183.
[27] M.R. Ogiela, U. Ogiela, L. Ogiela, Secure information sharing using personal biometric characteristics, in: Computer Applications for Bio-technology, Multimedia, and Ubiquitous City - International Conferences MulGraB, BSBT and IUrC 2012 Held as Part of the Future Generation Information Technology Conference, FGIT 2012, Gangneug, Korea, December 16–19, 2012. Proceedings., 2012, pp. 369–373.
[28] L. Ogiela, M.R. Ogiela, Bio-inspired cryptographic techniques in information management applications, in: 30th IEEE International Conference on Advanced Information Networking and Applications, AINA 2016, Crans-Montana, Switzerland, 23–25 March, 2016, 2016 pp. 1059–1063.
[29] J. Mielikainen, Lossless compression of hyperspectral images using lookup tables, IEEE Signal Process. Lett. 13 (3) (2006) 157–160.
[30] C.-C. Lin, Y.-T. Hwang, An efficient lossless compression scheme for hyperspectral images using two-stage prediction, IEEE Geosci. Remote Sens. Lett. 7 (3) (2010) 558–562.
[31] E. Auge, J. Santalo, I. Blanes, J. Serra-Sagrista, A. Kiely, et al., Review and implementation of the emerging ccsds recommended standard for multispectral and hyperspectral lossless image coding, in: Data Compression, Communications and Processing (CCP), 2011 First International Conference on, IEEE, 2011, pp. 222–228.
[32] N. Aranki, A. Bakhshi, D. Keymeulen, M. Klimesh, Fast and adaptive lossless on-board hyperspectral data compression system for space applications, in: Aerospace conference, 2009 IEEE, IEEE, 2009, pp. 1–8.
[33] D. Sal, M. Graña, A multiobjective evolutionary algorithm for hyperspectral image watermarking, Comput. Intell. Remote Sens. (2008) 63–78.
[34] H. Fang, Q. Zhou, K. Li, Robust watermarking scheme for multispectral images using discrete wavelet transform and tucker decomposition, J. Comput. Phys. 8 (11) (2013) 2844–2850.
[35] J. Serra-Ruiz, D. Megías, A novel semi-fragile forensic watermarking scheme for remote sensing images, Int. J. Remote Sens. 32 (19) (2011) 5583–5606.
[36] W. Hong, T. Chen, C. Shiu, Reversible data hiding for high quality images using modification of prediction errors, Journal of Systems and Softwar 82 (11) (2009) 1833–1842.
[37] W. Hong, T.-S. Chen, C.-W. Shiu, Reversible data hiding for high quality images using modification of prediction errors, J. Syst. Softw. 82 (11) (2009) 1833–1842.
[38] Z. Ni, Y.-Q. Shi, N. Ansari, W. Su, Reversible data hiding, IEEE Trans. Circuits Syst. Video Technol. 16 (3) (2006) 354–362.
[39] A. Castiglione, R. Pizzolante, F. Palmieri, B. Masucci, B. Carpentieri, A. De Santis, A. Castiglione, On-board format-independent security of functional magnetic resonance images, ACM Trans. Embed. Comput. Syst. 16 (2) (2017) 56.
[40] A. Hore, D. Ziou, Image quality metrics: PSNR vs. SSIM, in: Pattern recognition (icpr), 2010 20th international conference on, IEEE, 2010, pp. 2366–2369.
[41] Z. Wang, A.C. Bovik, A universal image quality index, IEEE Signal Process. Lett. 9 (3) (2002) 81–84.
[42] B. Penna, T. Tillo, E. Magli, G. Olmo, Transform coding techniques for lossy hyperspectral data compression, IEEE Trans. Geosci. Remote Sens. 45 (5) (2007) 1408–1421.
[43] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.
[44] F. Rizzo, B. Carpentieri, G. Motta, J.A. Storer, Low-complexity lossless compression of hyperspectral imagery via linear prediction, IEEE Signal Process. Lett. 12 (2) (2005) 138–141.
[45] R. Pizzolante, A. Castiglione, B. Carpentieri, A. De Santis, Parallel low-complexity lossless coding of three-dimensional medical images, in: 17th International Conference on Network-Based Information Systems, NBiS 2014, Salerno, Italy, September 10–12, 2014, 2014, pp. 91–98.
[46] R. Pizzolante, B. Carpentieri, A. Castiglione, A. Castiglione, F. Palmieri, Text compression and encryption through smart devices for mobile communication, in: Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS, 2013 Seventh International Conference on, IEEE, 2013, pp. 672–677.