Error-resilient and low-complexity onboard lossless compression of hyperspectral images by means of distributed source coding

Andrea Abrardo, Mauro Barni, Enrico Magli, Filippo Nencini

***Abstract*—In this paper we propose a lossless compression algorithm for hyperspectral images inspired by the distributed source coding principle. Distributed source coding refers to separate compression and joint decoding of correlated sources, which are taken as adjacent bands of a hyperspectral image. This concept is used to design a compression scheme that provides error resilience, very low complexity, and good compression performance. These features are obtained employing scalar coset codes to encode the current band at a rate that depends on its correlation with the previous band, without encoding the prediction error. Iterative decoding employs the decoded version of the previous band as side information, and uses a cyclic redundancy code to verify correct reconstruction. We develop three algorithms based on this paradigm, which provide different trade-offs between compression performance, error resilience and complexity. Their performance is evaluated on raw and calibrated AVIRIS images, and compared with several existing algorithms. Preliminary results of an FPGA implementation are also provided, which show that the proposed algorithms can sustain an extremely high throughput.**

***Index Terms*—Lossless compression; hyperspectral images; error resilience; distributed source coding.**

I. Introduction

Compression of multispectral and hyperspectral images has recently received a lot of attention. New sensors are generating increasing amounts of data, especially in the spectral dimension, as scenes are imaged at a very fine wavelength resolution. This is particularly useful in terms of potential applications, as spectral features allow to extract important information from the data. However, it also makes the size of the acquired images extremely large. Since many sensors, especially spaceborne ones, cannot store all the data but need to transmit them to a ground station, there is the need of reducing the data size in order to match the available bandwidth.

Image compression techniques can be employed to mitigate this problem, allowing to transmit more scenes in the same amount of time. Several types of compression are possible. In lossless compression, the reconstructed image is identical to the original. In near-lossless compression, the maximum absolute difference between the reconstructed and original image does not exceed a user-defined value. In lossy compression, given a target bit-rate, the reconstructed image must be as similar as possible to the original "on average", i.e., typically in mean-squared error sense. Lossless compression is highly desired to preserve all the information contained in the image; unfortunately, the state-of-the-art algorithms provide limited compression ratios. For example, [1], [2] achieve a compression ratio of about 3.5:1 on 16 bit AVIRIS data. Near-lossless and lossless techniques yield larger size reductions, at the expense of some information loss. E.g., in [3] it is shown that bit-rates of 0.5 and 0.1 bits per pixel (bpp) can be achieved with little or no impact on image classification performance.

In the past, the vast majority of compression algorithms have been optimized for maximum coding efficiency, i.e., reducing the data size as much as possible, with some constraints on the error in case of lossy compression. However, it should be noted that other features are also important. This is especially true for on-board compression of hyperspectral images, which is the subject of this paper. Such features include:

1) **Low encoder complexity.** In the design of the on-board processing unit of an operational system, the selection of a compression algorithm is not only based on the compression efficiency, but also, and perhaps more importantly, on its low complexity.
2) **Error resilience.** Compression algorithms should be error resilient, i.e. capable of dealing with "errors". This requirement is important because of the way the data are processed and transmitted to the ground segment.
3) **Ability to handle raw data.** Compression algorithms should be able to handle raw data, while most current algorithms have only been tested on calibrated data.

As for feature 1, small differences in the compression ratio are usually not very critical, while it is of paramount importance that the on-board encoder has *low complexity*, in order to match the low computational capabilities and the high sensor data rate, possibly providing real-time operation. It is a matter of fact that operational systems do not employ sophisticated state-of-the-art compression algorithms, but rather simpler suboptimal algorithms. A notable example is the 2D lossy compression algorithm recently standardized by the CCSDS (Consultative Committee for Space Data Systems) [4], which is a wavelet-based algorithm similar to JPEG 2000 [5] in many aspects, but with significantly lower complexity and reduced performance [6].

A. Abrardo, M. Barni and F. Nencini are with the Dip. di Ingegneria dell'Informazione, Università di Siena, Via Roma 56, 53100 Siena, Italy, e-mail: (abrardo,barni,nencini)@dii.unisi.it. E. Magli is with Dip. di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy, Ph.: +39-011-5644195, FAX: +39-011-5644099, e-mail: enrico.magli@polito.it.

Regarding feature 2, this requirement is becoming increasingly important as the architecture of on-board processing systems is undergoing a paradigm shift. In particular, until a few years ago it was customary to employ radiation-hardened digital signal processors to perform on-board processing, including compression. Radiation-hardened components have the same functionality of an equivalent standard processor, but they are designed to be insensitive to ionization. Such processors are extremely expensive to design and manufacture; as a result, the available devices do not have state-of-the-art computational capabilities. This has led to a recent trend of employing off-the-shelf processors, as in [7]. Off-the-shelf processors are prone to the effect of radiations; e.g., a typical effect is to occasionally flip some bits in the on-board memory. Moreover, when the data are transmitted to the ground station, there is a small probability that the data are corrupted. A corrupted packet will prevent from decoding other packets that depend on it, causing significant error propagation unless some error resilience is provided.

As a consequence, modern compression algorithms should be able to manage the event of errors in the data. What is meant by "manage" errors is that the image decoding process should not break down completely, nor excessively impair the data, upon occurrence of occasional errors or packet losses. Traditional compression algorithms such as those in [8], [2] are unable to recover upon a single bit error in the data, preventing from decoding the remainder of the compressed file after the error: i.e., if one of the first bits of the file is corrupted, the complete scene will be lost. This undesirable behavior is caused by two aspects of the classical structure of lossless compression algorithms, which first perform a spatial/spectral prediction, and then compress the prediction error samples using an entropy coder. As the entropy coder uses a variable-length code, a bit error will produce the erroneous decoding of a symbol, and the loss of synchronization of the entropy decoder, which generally leads to wrong decoding of all the following symbols. In general, the better the compression, the heavier the error propagation, as there is no redundancy left in the data to resynchronize the decoder; e.g., while a (memoryless) Huffman coder will occasionally resynchronize and start yielding correct symbols, an arithmetic coder is very unlikely to do so, as the encoding process has memory of the past, and the memory will propagate decoding errors to all the subsequent symbols. Besides, the prediction process also has memory, and any error in a prediction error sample will propagate to more and more samples (e.g., to all the subsequent bands) as the inverse predictor is used to reconstruct the image. To manage the event of errors, the compression algorithm should be designed so as to be able to withstand errors, i.e. to limit the scope of error propagation as much as possible. Moreover, its behavior should be predictable. It is anticipated here that, as has been said above, this has a cost in terms of compression efficiency, in that limiting error propagation is only possible if some redundancy is left in the data. The payback is that this allows to employ much less expensive and more powerful on-board processing systems. The importance of error-resilience is also witnesses by its inclusion as a selection criterion for the next generation of lossless and lossy multispectral and hyperspectral compression standards developed by the CCSDS.

At this point, one may wonder how the existing algorithms fare in terms of these requirements. Regarding feature 1, it should be noted that there do exist a few compression algorithms whose complexity would be suitable for on-board processing, e.g. those in [1], [2], [9]. Most other algorithms, including those based on 3D-CALIC [8], [10] and JPEG 2000 [3] have excessive complexity. JPEG-LS [11] is a simple and effective algorithm, but in its current form it can only handle 2D and not 3D images; therefore, many authors consider a simple differential JPEG-LS algorithm, in which JPEG-LS compresses the difference between two adjacent bands, instead of the original bands. Regarding feature 2, this problem is almost completely overlooked in the literature, mainly because it is a very recent requirement. The only algorithm addressing this issue has been proposed in [12], and performs lossy compression using trellis-coded quantization; however, this is out of the scope of this paper, as we consider lossless and not lossy compression. Out of all the remaining algorithms, none of them would be consistently able to withstand a single bit error without impairing the subsequent parts of the data. Therefore, no existing algorithm is able to provide both low complexity and error resilience. As for feature 3, while many compression algorithms, e.g. [2], [9], claim to be suitable for on-board compression thanks to their low-complexity, they have only been tested on calibrated and not raw data. In fact, most lossless and lossy compression papers provide results on calibrated AVIRIS scenes as these scenes are made publicly available by the NASA. However, the raw data generated on-board are known to have rather different characteristics [13], potentially leading to different compression strategies [14]. Therefore, the best algorithms for compression of raw scenes are not necessarily going to be the best on the raw data. In conclusion, there is a strong need of a comparison of different algorithms on the raw as opposed to the calibrated data.

The objective of this paper is to fill this gap, proposing new compression algorithms that provide low complexity *and* error resilience, and testing them on raw and calibrated data. The algorithms are inspired by a compression paradigm called distributed source coding (DSC) [15]. Previous works [16], [17] have shown the feasibility of DSC for low-complexity hyperspectral image compression; however, they suffered a significant loss in compression efficiency with respect to the state-of-the-art, and did not consider error resilience. In this paper we present three new algorithms with improved compression performance and reduced complexity with respect to [16], [17], which provide various degrees of error-resilience, allowing to select the best efficiency-resilience trade-off for a given application.

This paper is organized as follows. In Sect. II we review the DSC concepts and discuss previous work in this area. In Sect. III we describe the basic version of the proposed algorithm. In Sect. IV we describe two algorithms that build on the basic version by providing increasing degrees of error resilience. Their performance evaluation is reported in Sect. V. Finally, conclusions are drawn in Sect. VI.

II. BACKGROUND

In this section we first describe the fundamentals of DSC, and then explain them using the simple example of scalar coset codes.

A. Distributed source coding theory

In recent years, DSC has received an increasing attention from the signal processing community. DSC considers a situation in which two (or more) statistically dependent information sources X and Y must be encoded by separate encoders that are not allowed to talk to each other. Let X be a discrete independent and identically distributed (i.i.d.) information source that can assume M different values in the set $\mathcal{X} = \{0, 1, \ldots, M-1\}$. Then the entropy of X is defined as $H(X) = -\sum_{j=0}^{M-1} p_j^X \log_2 p_j^X$, where $p_j^X = P(X = j)$. The source Y is also defined in the same way using a probability distribution p_j^Y and an alphabet $\mathcal{Y} = \mathcal{X}$. The joint entropy of X and Y is defined as $H(X,Y) = -\sum_{j=0}^{M-1}\sum_{k=0}^{M-1} P(X = j, Y = k) \log_2 P(X = j, Y = k)$. Information theory shows [18] that separately encoding X and Y requires a minimum total rate of $H(X) + H(Y)$. Conversely, joint encoding (e.g., predicting X from Y) requires a total rate equal to $H(X,Y) \leq H(X) + H(Y)$; this is because joint encoding exploits the correlation between X and Y. Thus, separate lossless compression may seem less efficient than joint encoding. Somewhat surprisingly, DSC theory proves that, under certain assumptions, separate encoding is optimal, provided that the sources are *decoded* jointly [19]. For example, with two sources it is possible to perform "standard" encoding of the first source Y (called *side information*) at a rate equal to its entropy $H(Y)$, and "conditional" encoding of the second source X at a rate arbitrarily close to the conditional entropy $H(X|Y)$, which is lower than the entropy $H(X)$, with no information about Y available at the second encoder. This is the scenario considered in this paper.

For completeness, we briefly mention that DSC theory also encompasses lossy compression [20]; it has been shown that, under certain conditions, there is no performance loss in using DSC [20], [21], and that, in the general case, losses are bounded below 0.5 bit per sample (bps) for quadratic distortion metric [22]. In practice, lossy DSC is typically implemented using a quantizer followed by lossless DSC. Lossless and lossy DSC have several potential applications, e.g., coding for non co-located sources such as sensor networks, distributed video coding [23], layered video coding [24], [25], and remote sensing image compression [16], [26], just to mention a few.

Traditional entropy coding of an information source can be performed using one out of many available methods, the most popular being arithmetic coding and Huffman coding. DSC coders are typically implemented using the "binning" concept. We let $\mathcal{X}$ be the alphabet of X (typically $\mathcal{X} = \{0, 1, \ldots, 2^{16} - 1\}$ for unsigned hyperspectral data), and consider a vector X_N of N samples of X taking values in $\mathcal{X}^N$. We want to encode X_N using approximately $H(X|Y)$ bits per sample (bps). Binning amounts to partitioning $\mathcal{X}^N$ into p disjoint sets $\mathcal{C}_i$, $i = 1, \ldots, p$, called "cosets", such that $\mathcal{C}_1 \cup \ldots \cup \mathcal{C}_p = \mathcal{X}^N$. A good partitioning is such that the elements in each coset have maximum distance between each other. Then the encoder transmits only the label of the coset that the actual observed vector X_N belongs to, using $\frac{\lceil \log_2 p \rceil}{N}$ bps. This is clearly an incomplete description of X_N, as the label only describes X_N up to a certain ambiguity level. The decoder receives the coset label, which specifies a list of candidate reconstruction vectors (the coset elements). It picks as estimate of X_N the element of the coset that is closest to the side information Y (more precisely, a vector of N samples of Y) in mean-squared error norm. Ideally, p should be chosen such that the bit-rate is roughly equal to $H(X|Y)$. The interested reader can refer to [15] for an excellent tutorial on DSC, and to [16] for a longer introduction to this subject in the context of hyperspectral image compression.

In practice, coset codes with good geometrical properties can be obtained using channel codes, by representing the source using the syndrome or the parity bits of a suitable code of given rate. The syndrome identifies sets of codewords (i.e. the cosets) with maximum distance properties, so that decoding an ambiguous description of a source, given the side information, incurs minimum error probability. If the correlation between X and Y can be modeled as a "virtual" channel described as $X = Y + W$, with W an additive noise process, a good channel code for that transmission problem is also expected to be a good DSC code [21]. In practice, trellis channel codes [27], turbo codes [28] and low-density parity-check codes [29] have been used; distributed arithmetic codes have also been proposed [30].

B. Scalar coset codes

1) A simple example: Since the channel codes mentioned above have a relatively high encoder complexity, in this paper we employ a scalar coset code, i.e. a code with $N = 1$, which exhibits lower coding performance but significantly lower complexity. We describe the scalar coset code referring to Fig. 1 by means of an example. We assume that the samples to be encoded are represented on three bits, assuming values from 0 to 7. We divide the eight values into four cosets, identified by triangle-up, square, circle and triangle-down. Each coset contains two values taken at maximum distance between each other. Compression operates as follows. Let us say for example that we want to encode the current sample X, which happens to have value equal to 2. If we did not use the side information Y, we would need 3 bits to specify all possible values of X. Instead, we label each coset and send to the decoder only the label of the coset that X belongs to. In this example we have four cosets, hence we need two bits; compared with the three bits of the previous case, the rate has been reduced by one third. The decoder receives the coset label for X, which tells her that X is in $\{2, 6\}$. The decoder employs the correlated side information Y to disambiguate between these two values, by choosing for X the value that is closest to Y. For example, if $Y = 3$, then the decoder will reconstruct $X = 2$. Intuitively, if Y is correlated with X, the probability of error, i.e. that Y is closer to another element of the coset other than the true value of X, is vanishingly small.

This simple example allows to highlight a few important aspects of DSC. One of them is the relation between the

number and size of the cosets. In the example above, instead of taking four cosets with two elements in each, we could have taken two cosets with four elements in each, transmitting only one bit as label for X. The optimal number of coset actually depends on how much correlated X and Y are. The larger the number of cosets, the larger the minimum distance between two elements of the same cosets. Therefore, if X and Y are only mildly correlated, we need a large minimum distance to make sure that the decoder will be able to disambiguate X, and hence we will need more bits for the label. Vice versa, if X and Y are very strongly correlated, we can create less cosets and use less bits. A typical choice is to take the number of cosets equal to a power of two, i.e., 2^k. This choice makes it easy to derive the coset label, as it can be easily shown that the label is simply the k least significant bits of X.

2) Minimum-distance decoding of scalar coset codes: Since scalar coset codes are employed in the proposed algorithm, we discuss in more detail their decoding procedure, which constitutes an integral part of the algorithm description. This also highlights the simplicity and inherent low complexity of the proposed approach.

In practice, minimum-distance decoding is performed as follows. Let $\bar{X} = X\%N_C$ be the received coset label, where N_C is the number of cosets and $\%$ denotes the remainder of the division between two integers, with $0\%N_C = 0$. Given the side information Y, one identifies the two values of X in the given coset $\bar{X}$ that are closest to Y. Since N_C is also the distance between two consecutive elements of a coset, the first candidate can be written as $X_L = \mathrm{floor}(Y/N_C)N_C + \bar{X} - aN_C$, where a can be 0 or 1 and is chosen so that $X_L < Y$. The second candidate is $X_U = X_L + N_C$. The decoder reconstructs $\hat{X} = \arg\min_{U,L}(|Y - X_U|, |Y - X_L|)$. Exact reconstruction ($\hat{X} = X$) occurs if $|X - Y| \leq \mathrm{floor}(N_C/2)$ for N_C odd, and $|X - Y| < (N_C/2)$ for N_C even.

3) Choice of number of cosets: Throughout this description we have made the assumption that the amount of correlation (i.e., $H(X|Y)$) is known *a priori* at the encoder. However, this is generally not true in practical settings. The only viable way to cope with this is to slightly betray the DSC principle, and allow communication between the two sources X and Y in order to choose the number of cosets, i.e. the parameter k, to be used for coset coding. One can choose k such that the rate is approximately equal to $H(X|Y)$; this minimizes the bit-rate, but is prone to occasional residual errors, as the Slepian-Wolf theorem guarantees that the error probability is vanishingly small, but not exactly zero. Alternatively, k can be designed for the worst case, allowing zero-error reconstruction, i.e., exactly lossless compression.

4) Role of the side information: From the discussion above, it is clear that the side information Y should be as close as possible to X in mean-squared error sense. Statistically, this allows to employ the smallest number of bits for the coset label, while still allowing perfect reconstruction. In practice, in many cases the "simplest" side information is not the most effective one. We show this with reference to the hyperspectral image compression scenario. While two adjacent bands X and Y are typically highly correlated, the correlation may have a linear structure, such that an improved side information $Y' = aY + b$

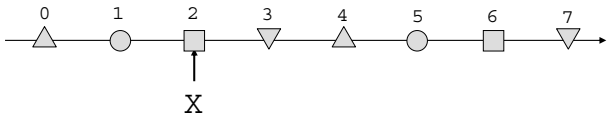

Fig. 1. DSC using scalar coset codes.

may be closer to X in mean-squared error sense. With DSC, it is possible to exploit this kind of structure without explicitly sending the parameters a and b to the decoder. The encoder can choose a transmission rate approximately equal to $H(X|Y')$, which is smaller than $H(X|Y)$. Along with the coset label, the encoder also periodically sends a cyclic redundancy check (CRC) parity string for the decoder perusal. For example, the encoder can send a 32-bit CRC string computed over a block of 256 consecutive samples. The decoder, which does not know the exact value of Y', tries different possible values for Y' for all the samples in the block, e.g. using Y as initial guess, until the decoded samples of the block are such that their CRC string exactly matches the received string. In this way, there is a bit-rate saving at the encoder, in that the parameters of the estimator that relates Y and Y' do not have to be transmitted, allowing to employ sophisticated correlation models.

5) Error resilience: Another important aspect regards error resilience. One reason why existing algorithms are not error-resilient is that the predictor of the current sample X is a specific value that can be computed in a causal way only if all previous pixels of the image have been decoded without errors. The decoder reconstructs X as the predictor plus the prediction error; hence, if this latter is corrupted by errors, this will yield an erroneous reconstruction of X. With DSC, this is not the case. In fact, *any* side information Y that is sufficiently close to X will allow the decoder to reconstruct the correct value. Let us suppose that the side information $Y = 3$ has been corrupted, and a wrong value $Y' = 1$ is available; in this example, $Y' = 1$ will yield $X = 2$. As long as Y' is closer to X than to any other element of the same coset, the decoder will provide the correct value of X. Therefore, the DSC approach attenuates error propagation by allowing to decode a sample from a corrupted predictor, provided that the label of the sample is not corrupted. Moreover, in terms of entropy coding, a scalar coset code is a fixed-length code, as it employs K bits for each sample; therefore, the effect of a single bit error is ideally limited to the sample affected by that error.

C. Prior work

For remote sensing image compression, which is addressed in this paper, the idea is to consider each band X_i of a multispectral or hyperspectral image as an individual source [16], [17]. Adjacent bands are correlated, and this raises the problem of exploiting their correlation. Classical algorithms do so by employing interband prediction, i.e., performing modeling at the encoder. On the contrary, DSC allows to perform little or no modeling at the encoder, with lower encoder complexity and error-resilience, and achieving ideally

the same bit-rate as a classical joint encoder. Indeed, in DSC it is the decoder that is burdened with the modeling task; however, in remote sensing applications this is not an issue as the computational constraints at the ground segment are not nearly as tight as on-board. It should be noted that in [26] a wavelet-based hyperspectral compression algorithm is described, which applies a 2D discrete wavelet transform to each band, and then applies DSC between the wavelet coefficients of adjacent bands. This algorithm can perform both lossless and lossy compression, but its complexity is high. Moreover, a number of papers have been published on the subject of video compression using DSC; the interesting reader is referred to [31] for a survey of recent results.

III. Proposed algorithm: A1

In order to satisfy the requirements outlined so far and allow to trade off between compression efficiency and error resilience, we developed three algorithms: the first algorithm (hereafter referred to as A1) focuses on coding efficiency, the third one (A3) on error resilience, while the second (A2) attempts to trade off between the two requirements. In this section we describe the A1 algorithm, which achieves the best coding efficiency, and constitutes the basis for the next two algorithms.

All algorithms have been designed assuming that the data are processed band by band. In particular, the algorithms independently encode non-overlapped blocks of 16×16 samples in each band. Therefore, it is assumed that one block of 16×16 samples is available in the current and previous bands. This is akin to a band-sequential format, with a requirement of 16 lines of data available in memory. This choice has been made for two reasons. Firstly, it makes it easier to parallelize the algorithm in hardware implementation, as multiple 16×16 blocks available in memory can be encoded concurrently, thus highly increasing the sustainable throughput. Secondly, it allows to achieve a high degree of spatial error resilience thanks to the block structure, coupled with the DSC built-in error resilience in the spectral direction.

A. Encoder

1) Overview of encoding and decoding process: The general outline of the proposed encoding/decoding strategy is described in the following. The description refers to a generic image band i; all bands are treated in the same way, with the exception of the first one that is either transmitted uncompressed (as is done in the present paper), or coded with any available 2D lossless compression scheme.

As a first step, in order to account for the spatially varying nature of the image, the bands are partitioned into non-overlapping blocks of size 16×16. The first block of the band is also transmitted uncompressed, for reasons that will be explained later on. Subsequent blocks are processed as follows. Let $x_{m,n,i}$ denote a generic pixel in m-th line, n-th column, and i-th band. We assume that $x_{m,n,i}$ can be predicted from the corresponding pixel of the previous band $x_{m,n,i-1}$ as:

$$x_{m,n,i} = f(x_{m,n,i-1}) + v_{m,n} \tag{1}$$

where $v_{m,n}$ is the prediction error, whose variance determines the amount of information between two consecutive bands (the higher the variance, the lower the correlation, hence resulting in a lower compression ratio). As in [16] we assume an affine relationship between $x_{m,n,i}$ and $x_{m,n,i-1}$, that is:

$$f(x_{m,n,i-1}) = \mu_i + \alpha(x_{m,n,i-1} - \mu_{i-1}); \tag{2}$$

where α is a constant to be determined and μ_{i-1} and μ_i are the average pixel values of the current block (μ_i) and the co-located block in the previous band (μ_{i-1}). The encoder estimates the value of α, μ_i and μ_{i-1} by inspecting the i-th and $(i-1)$-th bands. Note that α may vary from block to block, allowing a fine adaptation to the local characteristics of the image. Then the encoder computes the maximum value of the prediction error $v_{m,n}$ for all pixels in the current block. Once such maximum error is known, the encoder can determine the number of least significant bit-planes to be transmitted to the decoder (such a number in turns determines the distance between the elements of a coset as explained in section II-B). Finally the encoder computes a CRC of the pixels in the block and sends it to the decoder together with the bit-planes.

The decoder operates as described in section II-B. However, before coset decoding, it must first predict the pixel values in the i-th band from those of the $(i-1)$-th band. The predicted values, in fact, play the role of the side information and are needed to correctly reconstruct the actual pixel values. Since the decoder does not know the particular prediction function used by the encoder, it tries to guess it. For each guess the pixels of the block are reconstructed and the CRC computed again. This process terminates upon occurrence of a CRC that matches the one included in the compressed file. If the CRC is long enough, the probability that a wrong guess will produce a correct CRC is negligible.

2) Detailed encoder description: In the following a detailed description of the steps outlined above is given.

1) For each 16×16 block, the first step carried out by the encoder consists in estimating $x_{m,n,i}$ from $x_{m,n,i-1}$. To do so, it adopts a Least-Squares estimator assuming the form
$$\alpha = \frac{\sum_m \sum_n (x_{m,n,i-1} - \mu_{i-1}) \cdot (x_{m,n,i} - \mu_i)}{\sum_m \sum_n (x_{m,n,i-1} - \mu_{i-1}) \cdot (x_{m,n,i-1} - \mu_{i-1})}; \tag{3}$$
where the sums are taken over all the pixels of the block. With respect to the scheme adopted in [16], where the encoder tries to minimize the maximum prediction error, we opted for a Least-Squares estimator, since it is much faster to compute and better fits the new encoding strategy described below (item 4), for which the average error is more relevant than the maximum one.
2) The value of α found according to equation (3) is quantized by means of a uniform scalar quantizer with 256 levels in the $[0, 2]$ range. Quantization is necessary to allow the decoder to guess the correct value of α in a finite and small number of steps. As a consequence, the predicted values of the pixels within the block are computed as
$$\widetilde{x}_{m,n,i} = \mu_i + \widehat{\alpha}[x_{m,n,i-1} - \mu_{i-1}]; \tag{4}$$

The average values μ_i and μ_{i-1} are rounded off to the nearest integer.

3) A prediction error vector is calculated for each block as

$$e_{m,n} = x_{m,n,i} - \widetilde{x}_{m,n,i}; \tag{5}$$

where we drop the subscript i to emphasize that this is the error vector for the current block. The maximum absolute error in $e_{m,n}$ is used to compute the number k of LSB planes that should be transmitted to ensure that the distance between the elements of each coset is larger than twice the maximum error (see section II-B). In particular we have:

$$k = \left\lfloor \log_2(\|e_{m,n}\|_\infty) \right\rfloor + 2. \tag{6}$$

where $\|\cdot\|_\infty$ denotes the $\mathcal{L}_\infty$ norm, i.e., $\max_{m,n}|e_{m,n}|$. A few words are in order here to explain the above formula. To assure recovery at the decoder side, the maximum error of the block has to be strictly lower than half of the distance among two consecutive cosets, i.e. $|e_{m,n}|_\infty < 2^{k-1}$. This equation should be rewritten as $k > \log_2(|e_{m,n}|_\infty) + 1$. If $|e_{m,n}|_\infty$ is not a power of two, the lowest integer value that satisfies the equation is $\lceil \log_2(|e_{m,n}|_\infty) + 1 \rceil = \lfloor \log_2(|e_{m,n}|_\infty) + 1 \rfloor + 1 = \lfloor \log_2(|e_{m,n}|_\infty) \rfloor + 2$ (for example, if $|e_{m,n}|_\infty$ is 9, k is equal to 5). Otherwise, when $|e_{m,n}|_\infty$ is a power of two, the $\log_2(|e_{m,n}|_\infty) + 1$ term is an integer value and the equation is verified with $k = \Big(\log_2(|e_{m,n}|_\infty) + 1\Big) + 1 = \log_2(|e_{m,n}|_\infty) + 2$. These two results are combined in (6) into a single formula.

4) Only the $k-1$ LSB planes of each pixel in the block are transmitted. For the k-th LSB plane the following map is computed

$$M_{m,n} = \begin{cases} 0 & |e_{m,n}| < 2^{k-2} \\ 1 & \text{otherwise} \end{cases} \tag{7}$$

and stored for transmission to the decoder. In addition to the map $M_{m,n}$, the k-th LSB of those pixels for which $M_{m,n} = 1$ is also stored. This step constitutes a main deviation from the scheme used in [16] and the general approach outlined in section II-B. To explain the rationale behind it, we observe that setting the distance between cosets according to the maximum prediction error is a rather pessimistic approach. Actually in most cases the prediction error will be much smaller than the maximum, thus making it possible to transmit only 2^{k-1} LSBs. In order to avoid a reconstruction error, the position of those few pixels with a prediction error larger than 2^{k-2} is stored in the $M_{m,n}$ map and the corresponding k-th LSB transmitted separately. Note that, since the error distribution is not uniform, the $M_{m,n}$ map will be highly sparse and hence easily compressible. To take advantage of this, the positions of those $M_{m,n}$ entries that are equal to one are coded with a differential Huffman coding scheme applied to a zig-zag scanned version of the map. The above arguments are exemplified in Fig. 2 where a typical distribution of the prediction error is plotted and compared with the distance between two consecutive elements of a coset obtained by fixing $k-1$ LSBs. As can be seen, the number of pixels for which the prediction error exceeds half the distance between two coset elements is very small. The $M_{m,n}$ map is also depicted in the figure: the map is highly sparse thus allowing a high compression gain.

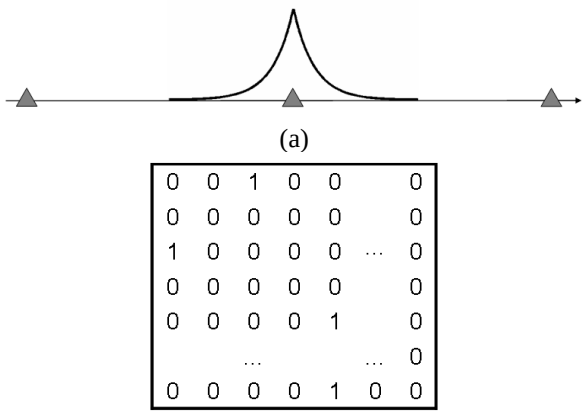

Fig. 2. (a) Representation of the prediction error distribution among two consecutive cosets and (b) an example of the $M_{m,n}$ map obtained by the analysis of a block.

In the decoding process the correct pixel value is recovered from $k-1$ LSBs when the prediction error is lower than 2^{k-2} or from k LSBs when $M_{m,n}$ is equal to 1.

5) In addition to the LSBs and the $M_{m,n}$ map, for each block the number k and 32 parity-check bits obtained by applying a CRC code to the values of the pixels in the block are stored in the bit-stream.

At the end of the encoding process a codestream formed as in Fig. 3 is produced.

The CRC code must have good error detection properties, in order to not hinder the decoding process, and must be simple to compute to avoid increasing the computational burden of the encoder and decoder, as the CRC computation represents a significant part of both. We have attempted to use an extremely fast error detection algorithm, such as the so-called "Internet checksum", i.e. the hash computed from the header bytes of the Internet protocol, which is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header. This checksum is extremely fast, but does not have sufficiently good error detection performance. In our scheme we used a 32-bit CRC with polynomial $d(x) = x^{32} + x^{31} + x^8 + 1$, [32], whose complexity is about 5.75 instructions per input byte. This CRC is only about 3.5 times slower than the Internet checksum, but almost 30 times faster than the CRC employed in [16], yielding a significant computational saving. An even faster CRC implementation exists with 3.25 instructions/byte;

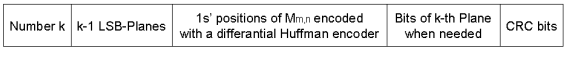

Fig. 3. Content of a compressed block of the A1 algorithm.

however, this latter requires large look-up tables. While this is not an issue on a software implementation, the look-up table would not fit in the internal memory of a hardware implementation, leading to reduced throughput.

B. Decoder

The first step the decoder needs to carry out, is the prediction of the pixel values in the i-th band from those of the $i-1$-th band. However, the decoder does not know the parameters of the predictor used by the encoder, hence it has to estimate them. Namely, for each block it has to estimate the values of μ_i and $\widehat{\alpha}$ appearing in (4); the value of μ_{i-1} is already available since the $i-1$-th band has already been reconstructed.

The value of μ_i is estimated by resorting to the pixels of spatially contiguous blocks already decoded, and to those of the previous band. Let us indicate with $\mu_{i(u)}$ and $\mu_{i(l)}$ the average values of the blocks above and on the left of the current block, and with $\mu_{i-1(u)}$ and $\mu_{i-1(l)}$ the corresponding values in the previous band. The value of μ_i is estimated by the following equation

$$\widehat{\mu}_i = \frac{1}{2}\left[\mu_{i(u)}\left(1+\frac{\mu_{i-1}-\mu_{i-1(u)}}{\mu_{i-1(u)}}\right)+\mu_{i(l)}\left(1+\frac{\mu_{i-1}-\mu_{i-1(l)}}{\mu_{i-1(l)}}\right)\right] \tag{8}$$

The use of the mean value of the left block explains why the first block of each band is encoded uncompressed. This allows to compute $\widehat{\mu}_i$ for the second block, setting $\mu_{i-1(u)}$ and $\mu_{i(u)}$ to zero.

For each value of $\widehat{\mu}_i$, a set of 256 uniform levels from 0 to 2 is used to estimate $\widehat{\alpha}$. In turn, all the candidate predictors are combined with the received LSBs, to reconstruct the correct pixel values in the current band. Of course the decoder does not know a priori whether the predictor it is currently using corresponds to the one employed by the encoder. To understand it, the reconstructed values are validated by checking if CRC obtained from the reconstructed pixels is identical to the received CRC.

When this condition is verified, the reconstruction process of the block is terminated and the decoder proceeds to the reconstruction of the subsequent block.

IV. Improving error resilience: A2 and A3 algorithms

As outlined in Sect. I, achieving a good compression rate may not be enough in modern on-board compression systems, for which low complexity and error resilience are also important. In this section we present two modifications of the basic A1 algorithm described so far that aim to improve its robustness against errors.

Specifically, we introduce two new schemes; the former achieves a trade-off between coding gain and error resilience, in that it permits to avoid error propagation in most, but not all, the cases. The latter permits to always avoid error propagation in the presence of isolated errors, at the expenses of a more significant reduction of compression efficiency.

A. Error model

In order to make a sensible design for error resilience, one has to make some assumptions regarding the error model. In this paper, we consider two different kinds of errors. The first class is given by errors in the on-board memory. These errors can be modeled as an occasional isolated bit-flipping. The second class consists of errors in the downlink communication stage. Many systems employ the CCSDS packet telemetry recommendation [33]. In this recommendation, similarly to most wireless communication systems, the source packets are broken down into transfer frames, which are the basic transmission units. Transfer frames are checked for errors individually, as they have a CRC. Therefore, the size of a transfer frame determines the minimum amount of data that are lost upon occurrence of a single bit error during the transmission. The maximum size, for the CCSDS standard, is 16384 bits, and is also constrained by the channel code employed in a particular mission. Thus, the maximum amount of data that are lost corresponds to few compressed 16×16 blocks. For the purpose of this paper, this means that, for transmission errors, it is reasonable to assume that a complete block has been lost, and that the co-located block in the previous and next bands is available. Note that the telemetry standard also provides the syntax necessary to resynchronize and restart the decoder, after an error, upon reception of a new set of uncorrupted compressed blocks. Thus, we assume that the decoder knows what blocks are corrupted, and can restart decoding upon reception of a new frame or source packet.

B. Algorithm 2

As we have shown in the previous section, if an error is present in the $(i-1)$-th band, the A1 algorithm will result in a reconstruction error in the i-th band, which will propagate through all the remaining bands. This means that all co-located blocks in the subsequent bands would be lost, as the decoder would be unable to reconstruct them. Algorithm A1 does have some built-in error resilience, in that the block-based approach will limit the propagation of the reconstruction error to a 16 $\times$ 16 block. However, the DSC approach can perform even better, as any side information that is sufficiently close to x will allow the decoder to correctly reconstruct the bitstream. The idea, then, is to use the $(i-2)$-th band as a back-up side information to be used when the $(i-1)$-th band is corrupted by errors. In this way spatial error propagation is avoided due to the block-based nature of the encoding/decoding process, and spectral error propagation is mitigated.

We now describe how we modified the A1 algorithm so as to take advantage of the availability of the $(i-2)$-th band when the $(i-1)$-th is not available.

First of all the encoder applies to the $(i-2)$-th band the same procedure used to predict the i-th band from the $(i-1)$-th one. In this way, the encoder computes the predictor parameters μ_{i-2}, α', the prediction error $e'_{m,n}$ between the i-th and the $(i-2)$-th band, the number k' of LSB planes necessary to reconstruct the i-th band from the $(i-2)$-th one, and the corresponding map $M'_{m,n}$. At this point three different cases are considered:

- $k' < k$;
- $k' = k$;
- $k' > k$.

When k' is lower than k (a rather rare case indeed, given that usually the correlation between bands tends to decrease when the spectral distance increases), k LSBs are enough to recover the current band values either from $x_{m,n,i-1}$ or $x_{m,n,i-2}$, so nothing has to be done and the encoder goes on as in the A1 algorithm. When $k' > k$ the encoder decides that error propagation can not be avoided and transmits only the bits that are required to ensure perfect reconstruction from $x_{m,n,i-1}$ as in algorithm A1. Note however that this situation does not occur often since with hyperspectral images band $i-2$ is typically close to band $i-1$, and most of the times we will have $k' = k$.

When $k' = k$ (which as we said is by far the most frequent case), the algorithm A2 continues to transmit the $k-1$ LSBs of the pixels of the block, however it now builds a new composite map $M^c_{m,n}$ by bit-wise OR-ing $M_{m,n}$ and $M'_{m,n}$:

$$M^c_{m,n} = M_{m,n} | M'_{m,n}. \tag{9}$$

where the symbol " $|$ " denotes the bit-wise "OR" operator. As with A1, the k-th LSB is transmitted for the pixels for which $M^c_{m,n} = 1$.

The new map allows to recover $x_{m,n,i}$ both from $x_{m,n,i-1}$ and $x_{m,n,i-2}$, because k LSBs are transmitted when either $e_{m,n}$ or $e'_{m,n}$ is greater than 2^{k-2}. In fact, it may happen that the k-th LSBs are transmitted even for some pixels for which $k-1$ bit would be enough. This is often going to be the case when decoding is performed by relying on the $(i-1)$-th band. While this is clearly not a problem from a reconstruction point of view, it represents a source of redundancy in the compressed file that slightly decreases compression performance. In particular, the $M^c_{m,n}$ map obviously contains a larger number of ones, hence it is less compressible than $M_{m,n}$ (or $M'_{m,n}$), thus causing a slight reduction of coding efficiency.

The decoder of A2 algorithm is the same used by A1. If the decoder detects an error in a block of the i-th band, reconstruction of the corresponding block in band $i+1$ is carried out by using band $i-1$ as side information, limiting spectral error propagation.

C. Algorithm 3

Algorithm 2 fails to recover $x_{m,n,i}$ from $x_{m,n,i-2}$ if $k' > k$ because additional LSB-planes would be required. To overcome this limitation and increase the robustness of the compressed bitstream at the expense of compression rate, we introduce a third scheme (A3) that permits to recover the current band even when $k' = k+1$. The encoder of the A3 algorithm defines a new map, hereafter referred to as $M''_{m,n}$:

$$M''_{m,n} = \begin{cases} 0 & |e'_{m,n}| < 2^{k-2} \\ 1 & 2^{k-2} \le |e'_{m,n}| < 2^{k-1} \\ 2 & |e'_{m,n}| \ge 2^{k-1} \end{cases} \tag{10}$$

$M''_{m,n}$ and $M_{m,n}$ are used to generate a composite map $M^c_{m,n}$:

$$M^c_{m,n} = \max\{M_{m,n}, M''_{m,n}\}. \tag{11}$$

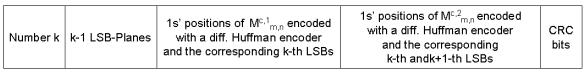

Fig. 4. Content of a compressed block of the A3 algorithm.

where, at each pixel position, the maximum is taken between the co-located entries of the two maps. At this point two maps are extracted from $M^c_{m,n}$

$$M^{c,1}_{m,n} = \begin{cases} 1 & if \;\; M^c_{m,n} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

$$M^{c,2}_{m,n} = \begin{cases} 1 & if \;\; M^c_{m,n} = 2 \\ 0 & \text{otherwise} \end{cases} . \tag{13}$$

The two maps are separately coded with a differential Huffman scheme and are stored in the compressed file with the k-th LSB when $M^{c,1}_{m,n}$ is equal to 1 and with the k-th and $(k+1)$-th LSBs when $M^{c,2}_{m,n}$ is 1 (see Fig. 4). Map $M^{c,1}_{m,n}$ allows to decode from band $i-2$ as in algorithm A2, while map $M^{c,2}_{m,n}$ allows one to use the $(k+1)$-th LSB of band $i-2$.

The structure of the A3 decoder differs from that of A2 only when $k' = k+1$. When this condition is verified, the $M^{c,2}_{m,n}$ map is uncompressed and the recovered $k-th$ and $k+1-th$ LSBs are used to reconstruct all pixels of the block with the same procedure described in III-B.

V. Results

A. Dataset description

In our experiments we employed a few scenes acquired by the AVIRIS sensor; AVIRIS covers the 0.41-2.45 μm spectrum in 10-nm bands. The instrument consists of four spectrometers that are flown at 20 km altitude with 17 m resolution. Each image has 512 lines, 224 bands and 680 lines.

In particular, seven raw images have been employed. These scenes are referred to as *Sc0*, *Sc3*, *Sc10*, *Sc11*, *Sc18*, *Hawaii* and *Maine*. The first five scenes are represented on 16 bits, and have been acquired in 2006 over Yellowstone, WY, while the last two ones are on 12 bits. These images are made publicly available by NASA, and can be downloaded at `http://compression.jpl.nasa.gov/hyperspectral/`. For the 16-bit scenes, the corresponding calibrated images have also been employed.

Moreover, we have considered the radiance data of four scenes from the 1997 acquisitions, namely *Cuprite*, *Jasper Ridge*, *Moffett Field* and *Lunar Lake*, which are publicly available on the Internet at `aviris.jpl.nasa.gov`. These images are calibrated, and have been used in many papers addressing hyperspectral image compression; hence their use facilitates comparisons with other algorithms.

B. Compression efficiency

For the 2006 raw and calibrated images, the proposed algorithms have been compared with the following algorithms.

- The JPEG-LS standard.
- The look-up table based algorithm (LUT) proposed in [2].

- The locally averaged interband scaling look-up table (LAIS-LUT) method proposed in [1].
- The FL algorithm proposed in [13], which employs the previous band for prediction, but adapts the predictor coefficients using recursive estimation. Note that, for this method, the last scene of each image is not included.
- The BG block-based compression algorithm proposed in [34], which employs a simple block-based predictor followed by an adaptive Golomb code. We have implemented the BG algorithm as described in [34], with the only difference that we subtract the mean value of the block in the current and previous band before computing the prediction gain; this yields slightly better bit-rates than those reported in [34].

The results are shown in Tab. I (the bit-rates for FL and LAIS-LUT have been taken from [35]). The average bit-rates on the raw and calibrated scenes are also reported (note that the average over the raw scenes does not include the 12-bpp scenes). JPEG-LS clearly performs poorly, as it does not exploit the spectral correlation. FL has the highest compression efficiency, scoring a 2.6 bpp gain with respect to JPEG-LS. The LUT and LAIS-LUT algorithms do not perform as well as expected on the raw data, as also reported in [36]. The BG algorithm is particularly interesting because it has the same predictor structure than the proposed algorithms, but encodes the prediction residuals instead of using coset codes. Its average performance is only 0.14 bpp worse than FL, which is remarkable, as FL employs a significantly more sophisticated adaptive predictor. The A1 algorithm, which has the best compression efficiency among the proposed algorithms, has a 0.48 performance gap with respect to BG. For the same predictor structure, this is the price to be paid for using the coset codes instead of encoding the prediction residuals. As anticipated, A2 and A3 have a larger gap, up to 0.77 bpp with respect to BG; however, as will be seen later on, this is compensated by the additional error resilience. Interestingly, the algorithm design proposed in this paper is significantly better than previous DSC algorithms [16], as A1, A2 and A3 have a performance gain of 0.7, 0.62 and 0.41 bpp respectively with respect to [16] (which does not target error resilience). In summary, A1 and A2 are better than LUT, and A3 only slightly worse. The performance gap with respect to a full-featured algorithm such as FL is between 0.62 and 0.91. This gap is compensated by the significant degree of error resilience, and the ability to perform parallel encoding exploiting the block-based structure. Inspection of the results on calibrated data shows similar trends. On these data LUT and LAIS-LUT are only slightly better than on the raw data with respect to the proposed algorithms, so that LUT and A1 have almost the same average bit-rate. The performance loss of A1/A2/A3 with respect to BG is roughly the same as on the raw data, and BG is 0.26 bpp worse than FL. The performance gain of the proposed algorithms with respect to [16] ranges between 0.67 and 0.36 bpp.

For the four 1997 calibrated images, the following additional algorithms have been employed for comparison.

- The SLSQ-HEU prediction-based algorithm proposed in [9].
- The BH block-based compression algorithm proposed in [34].
- The C-DPCM algorithm proposed in [37].
- The spectral relaxation labelled prediction (S-RLP) and spectral fuzzy-matching pursuit (S-FMP) algorithms proposed in [38].

The bit-rates on the 1997 images are shown in Tab. II. On these scenes, LUT and LAIS-LUT are known to perform particularly well, with average bit-rates down to 4.61 bpp. More sophisticated algorithms such as C-DPCM, S-RLP and S-FMP also achieve similar performance. For comparison, FL achieves 4.99 bpp, which is similar to SLSQ-HEU. BH and BG are only slightly worse, with 5.18 and 5.15 bpp. A1, A2 and A3 have a performance loss of respectively 0.41, 0.55 and 0.72 bpp with respect to BG. The gain with respect to [16] is between 0.67 and 0.36 bpp.

TABLE II
BIT-RATES (BPP) ACHIEVED BY VARIOUS ALGORITHMS FOR LOSSLESS COMPRESSION ON THE COMPLETE AVIRIS IMAGES.

	Jasper Ridge	*Lunar Lake*	*Cuprite*	*Moffett*	Average
JPEG-LS	7.67	6.87	7.02	7.41	7.24
SLSQ-HEU	4.97	4.97	4.95	5.00	4.97
C-DPCM	4.62	4.75	4.68	4.62	4.67
LUT	4.95	4.71	4.65	5.05	4.84
LAIS-LUT	4.68	4.53	4.47	4.76	4.61
S-RLP	4.65	4.69	4.69	4.67	4.68
S-FMP	4.63	4.66	4.66	4.63	4.65
FL	5.04	4.97	4.95	4.99	4.99
BH	5.23	5.11	5.11	5.26	5.18
BG	5.18	5.13	5.12	5.18	5.15
A1	5.60	5.51	5.50	5.64	5.56
A2	5.73	5.65	5.65	5.75	5.70
A3	5.93	5.78	5.78	5.99	5.87
[16]	6.27	6.17	6.16	6.32	6.23

In summary, as expected the proposed algorithms suffer a performance loss with respect to state-of-the-art techniques. Algorithm A1 has a loss of 0.4-0.5 bpp with respect to BG, which is the most similar algorithm. The loss with respect to FL is 0.15-0.25 bpp larger. In return of this loss, A1 and BG exhibit spatial error resilience, with a containment of an error inside a 16×16 block, and A1 has some spectral error resilience, as detailed in Sect. V-C. Moreover, the block-based structure of the proposed algorithms is amenable to parallel implementation.

C. Error resilience

Out of all the algorithms that have been used to assess the compression performance, only BH, BG and the proposed algorithms would be able to provide some error resilience. In particular, they spatially confine the effect of an error to the corrupted 16x16 block, affecting all subsequent bands. All the other algorithms have not been designed for error resilience, thus a single bit error or packet loss would render the subsequent portion of the compressed file undecodable.

The proposed DSC approach also provides different degrees of spectral error containment. When spectral containment is successful, only one pixel is going to be missing in the spectral

TABLE I
BIT-RATES (BPP) ACHIEVED BY VARIOUS ALGORITHMS FOR LOSSLESS COMPRESSION ON THE 2006 RAW AND CALIBRATED AVIRIS IMAGES.

Algorithm	Raw scenes								Calibrated scenes					
	Sc0	*Sc3*	*Sc10*	*Sc11*	*Sc18*	*Hawaii*	*Maine*	Average	*Sc0*	*Sc3*	*Sc10*	*Sc11*	*Sc18*	Average
JPEG-LS	9.18	8.87	7.32	8.50	9.30	4.58	4.50	8.63	6.95	6.68	5.19	6.24	7.02	6.42
LUT	7.13	6.91	6.25	6.69	7.20	3.27	3.44	6.84	4.81	4.62	3.95	4.34	4.84	4.51
LAIS-LUT	6.78	6.60	6.00	6.30	6.82	3.05	3.19	6.50	4.48	4.31	3.71	4.02	4.48	4.20
FL	6.23	6.10	5.65	5.86	6.32	2.64	2.72	6.03	3.96	3.83	3.40	3.63	3.94	3.75
BG	6.46	6.31	5.65	6.05	6.40	3.03	3.17	6.17	4.29	4.16	3.49	3.90	4.23	4.01
A1	6.92	6.78	6.10	6.53	6.92	3.49	3.65	6.65	4.81	4.69	4.01	4.41	4.77	4.54
A2	7.01	6.87	6.17	6.61	6.98	3.62	3.76	6.73	4.91	4.78	4.09	4.52	4.86	4.63
A3	7.26	7.11	6.31	6.78	7.22	3.74	3.91	6.94	5.16	5.02	4.24	4.71	5.12	4.85
[16]	7.61	7.48	6.80	7.25	7.61	4.13	4.29	7.35	5.49	5.35	4.65	5.09	5.46	5.21

vectors that have components in the affected block, as the decoder is able to restart from the next correctly received band. When it fails, all the subsequent pixels are going to be missing, as in the case of BH and BG. In other terms, the proposed algorithms yield a spectral vector which is either almost perfect, and can hence be used for information extraction, or is significantly impaired. For BH and BG, the spectral vector would always be impaired.

As has been seen in Sect. V, whether algorithms A1, A2 and A3 will be able to contain spectral error propagation depends on the values of k and k'. For A1, containment occurs when $k' < k$, which is relatively rare. For A2, it occurs when $k' \leq k$, which is a rather frequent case, while for A3 it occurs when $k' \leq k+1$, which almost always occurs. We have evaluated the occurrence frequency of these conditions over the 16-bit raw data included in the test set. The results are reported in Tab. III, and reflect the remarks above, highlighting the effectiveness of the proposed approach in containing errors in the spectral direction, besides the spatial error containment.

TABLE III
PERCENTAGE OF SUCCESSFUL SPECTRAL ERROR CONTAINMENTS FOR 16-BIT RAW DATA.

A1	A2	A3
8%	67%	96%

D. Complexity

The complexity of the proposed algorithms has been compared with that of existing schemes. As a global complexity measure, we have taken the running time of a software implementation of the encoders, in C language, on a Linux workstation. All programs have been compiled with `gcc`, enabling full optimization. The results are reported in Tab. IV, and are normalized with respect to the complexity of JPEG-LS. As can be seen, all proposed algorithms are faster than JPEG-LS, while providing better compression performance. The complexity of A1 is significantly smaller, while A2 is still less complex and more efficient than JPEG-LS, and provides a significant degree of error resilience. The LUT algorithm is slightly more complex than JPEG-LS; our results are in line with those reported in [2]. The BG algorithm is also in the same order of complexity of the proposed algorithms; however, it does not provide error resilience.

TABLE IV
COMPLEXITY OF VARIOUS ENCODERS.

Algorithm	Complexity
JPEG-LS	1
A1	0.57
A2	0.87
A3	0.89
BG	0.73
LUT	1.14

As has been seen, the A1, A2 and A3 algorithms have very low complexity. In addition to that, they can be easily parallelized by having different 16x16 blocks compressed at the same time. This can be achieved very easily on an FPGA.

Preliminary performance tests have been conducted on an engineered version of the algorithms run on space-qualified hardware. We only briefly report on the results, which have been worked out by Carlo Gavazzi Space SpA, and are fully described in [39].

First, the C software has been optimized and run on a TMS320C6701 EVM board. It has been found that algorithm A1 can sustain a throughput of 440 ksample/s, and algorithm A2 365 ksample/s, just using a software implementation.

Then, the algorithms have been written in VHDL language, and tested on a Xilinx Virtex 4 FPGA, using a 9-stage pipelined architecture. A single instance of the algorithm achieves a throughput in excess of 80 Msample/s, and as many instances can be used as fit into the FPGA. As a result of the pipelined architecture, the performance of the A1 and A2 algorithms is roughly the same. This shows that the proposed techniques are very effective for on-board compression, as they can handle very high data rates, providing support for real-time image compression.

E. Discussion

The results presented above lead to the following remarks.

- In terms of complexity, all the three proposed algorithms are very competitive. While they do exploit the spectral correlation, they are faster than JPEG-LS, which is a low-complexity 2D image compression algorithm. A throughput as high as 80 Msample/s has been demonstrated on an FPGA, and this can be increased even further exploiting parallelism.
- In terms of error resilience, the proposed algorithms are significantly better than any existing scheme, and

provide sevelar possible trade-offs between robustness and complexity.

- In terms of compression performance, there is a loss with respect to algorithms based on 3D prediction, which is the price to be paid for error resilience. On the raw AVIRIS dataset, the loss ranges from 0.48 to 0.77 bpp with respect to BG. However, the gain with respect to a 2D algorithm such as JPEG-LS is still as high as 2 bpp. This shows that the proposed algorithms are still a convenient choice with respect to a 2D algorithm, in terms of both compression efficiency and complexity, while providing error resilience as an added-value.

VI. Conclusions

We have proposed three compression algorithms which aim at providing error-resilient lossless compression of hyperspectral images. The algorithm design has been inspired by the DSC principle and, in particular, the concept of coset codes is employed to achieve error resilience.

The algorithms prove the effectiveness of the underlying principle, significantly outperforming previous work in this area [16] (note that the algorithm in [16] did not provide error resilience).

Performancewise, the main feature of the proposed algorithms is their low complexity, which is smaller than that of JPEG-LS. An FPGA implementation has demonstrated a throughput as high as 80 Msample/s, which can be further increased running multiple instance of the algorithm in parallel.

The three algorithms achieve different trade-offs between compression efficiency and error resilience. There is still a limited performance gap with respect to a predictive coder based on the same structure, which is compensated by the error resilience. As the high sensor data rates of present and future hyperspectral missions call for simple and fast compression techniques, the proposed algorithms represent good options for on-board compression, with the added benefit of error resilience.

References

[1] B. Huang and Y. Sriraja, "Lossless compression of hyperspectral imagery via lookup tables with predictor selection," in *Proc. SPIE, vol. 6365*, 2006.

[2] J. Mielikainen, "Lossless compression of hyperspectral images using lookup tables," *IEEE Signal Processing Letters*, vol. 13, no. 3, pp. 157–160, Mar. 2006.

[3] B. Penna, T. Tillo, E. Magli, and G. Olmo, "Transform coding techniques for lossy hyperspectral data compression," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 5, pp. 1408–1421, May 2007.

[4] *Image Data Compression*, CCSDS-122.0-B-1 Blue Book, November 2005.

[5] D.S. Taubman and M.W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards, and Practice*, Kluwer, 2001.

[6] *Image Data Compression*, CCSDS-120.1-G-1 Green Book, June 2007.

[7] K. Anderson, "Low-cost, radiation-tolerant, on-board processing solution," in *Proc. of IEEE Aerospace Conference*, 2005.

[8] X. Wu and N. Memon, "Context-based lossless interband compression - extending CALIC," *IEEE Transactions on Image Processing*, vol. 9, no. 6, pp. 994–1001, June 2000.

[9] F. Rizzo, B. Carpentieri, G. Motta, and J.A. Storer, "Low-complexity lossless compression of hyperspectral imagery via linear prediction," *IEEE Signal Processing Letters*, vol. 12, no. 2, pp. 138–141, Feb. 2005.

[10] E. Magli, G. Olmo, and E. Quacchio, "Optimized onboard lossless and near-lossless compression of hyperspectral data using CALIC," *IEEE Geoscience and Remote Sensing Letters*, vol. 1, no. 1, pp. 21–25, Jan. 2004.

[11] M.J. Weinberger, G. Seroussi, , and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1309–1324, Aug. 2000.

[12] G.P. Abousleman, T.-T. Lam, and L.J. Karam, "Robust hyperspectral image coding with channel-optimized trellis-coded quantization," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 4, pp. 820–830, Apr. 2002.

[13] M. Klimesh, "Low-complexity lossless compression of hyperspectral imagery via adaptive filtering," in *The Interplanetary Network Progress Report*, 2005.

[14] A.J. Pinho, "An online preprocessing technique for improving the lossless compression of images with sparse histograms," *IEEE Signal Processing Letters*, vol. 9, no. 1, pp. 5–7, Jan. 2002.

[15] Z. Xiong, A.D. Liveris, and S. Cheng, "Distributed source coding for sensor networks," *IEEE Signal Processing Magazine*, vol. 21, no. 5, pp. 80–94, Sept. 2004.

[16] E. Magli, M. Barni, A. Abrardo, and M. Grangetto, "Distributed source coding techniques for lossless compression of hyperspectral images," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, 2007.

[17] E. Baccaglini, M. Barni, L. Capobianco, A. Garzelli, E. Magli, F. Nencini, and R. Vitulli, "Low-complexity lossless compression of hyperspectral images using scalar coset codes," in *Proceedings of Picture Coding Symposium*, 2007.

[18] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley, New York, 1991.

[19] D. Slepian and J.K. Wolf, "Noiseless coding of correlated information sources," *IEEE Transactions on Information Theory*, vol. 19, no. 4, pp. 471–480, July 1973.

[20] A. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Transactions on Information Theory*, vol. 22, no. 1, pp. 1–10, Jan. 1976.

[21] S.S. Pradhan, J. Chou, and K. Ramchandran, "Duality between source coding and channel coding and its extension to the side information case," *IEEE Transactions on Information Theory*, vol. 49, no. 5, pp. 1181–1203, May 2003.

[22] R. Zamir, "The rate loss in the Wyner-Ziv problem," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 2073–2084, Nov. 1996.

[23] R. Puri and K. Ramchandran, "PRISM: a "reversed" multimedia coding paradigm," in *Proc. of IEEE International Conference on Image Processing*, 2003, pp. 617–620.

[24] Q. Xu and Z. Xiong, "Layered Wyner-Ziv video coding," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3791–3803, Dec. 2006.

[25] H. Wang and A. Ortega, "Scalable predictive coding by nested quantization with layered side information," in *Proceedings of IEEE International Conference on Image Processing*, 2004, pp. 1755–1758.

[26] N.-M. Cheung, C. Tang, A. Ortega, and C.S. Raghavendra, "Efficient wavelet-based predictive Slepian-Wolf coding for hyperspectral imagery," *Signal Processing*, vol. 86, no. 11, pp. 3180–3195, Nov. 2006.

[27] S.S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): Design and construction," *IEEE Transactions on Information Theory*, vol. 49, no. 3, pp. 626–643, Mar. 2003.

[28] J. Garcia-Frias and Y. Zhao, "Compression of correlated binary sources using turbo codes," *IEEE Communications Letters*, vol. 5, no. 10, pp. 417–419, Oct. 2001.

[29] A. Liveris, Z. Xiong, and C. Georghiades, "Compression of binary sources with side information at the decoder using LDPC codes," *IEEE Communications Letters*, vol. 6, no. 10, pp. 440–442, Oct. 2002.

[30] M. Grangetto, E. Magli, and G. Olmo, "Distributed arithmetic coding," *IEEE Communications Letters*, vol. 11, no. 11, pp. 883–885, Nov. 2007.

[31] C. Guillemot, F. Pereira, L. Torres, T. Ebrahimi, R. Leonardi, and J. Ostermann, "Distributed monoview and multiview video coding," *IEEE Signal Processing Magazine*, vol. 24, no. 5, pp. 67–76, Sept. 2007.

[32] D.C. Feldmeier, "Fast software implementation of error detection codes," *IEEE/ACM Transactions on Networking*, vol. 3, no. 6, pp. 640–651, Dec. 1995.

[33] *Packet Telemetry*, CCSDS-102.0-B-5 Blue Book, November 2000.

[34] M. Slyz and L. Zhang, "A block-based inter-band lossless hyperspectral image compressor," in *Proc. of IEEE Data Compression Conference*, 2005, pp. 427–436.

[35] A.B. Kiely and M.A. Klimesh, "Exploiting calibration-induced artifacts in lossless compression of hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing, submitted 2008. Available at compression.jpl.nasa.gov/hyperspectral/*.

[36] E. Magli, "Multiband lossless compression of hyperspectral images," *IEEE Transactions on Geoscience and Remote Sensing, to appear 2009*.

[37] J. Mielikainen and P. Toivanen, "Clustered DPCM for the lossless compression of hyperspectral images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 12, pp. 2943–2946, Dec. 2003.

[38] B. Aiazzi, L. Alparone, S. Baronti, and C. Lastri, "Crisp and fuzzy adaptive spectral predictions for lossless and near-lossless compression of hyperspectral imagery," *IEEE Geoscience and Remote Sensing Letters*, vol. 4, no. 4, pp. 532–536, Oct. 2007.

[39] A. Bertoli and R. Grimoldi, *Implementation and Performace Evaluation of Parallel Algorithm for Lossless Hyperspectral Image Compression*, 2009, Final report of ESA ITI project "PILL".