Chapter 3

Multitemporal Hyperspectral Image Compression

** The content of this chapter has been published by Yaman Dua, Ravi Shankar Singh, and Vinod Kumar. "Compression of multi- temporal hyperspectral images based on RLS filter." The Visual Computer (2020): 1-11.* `https://doi.org/10.1007/s00371-020-02000-6` *(Springer, IF: 2.601)*

3.1 Introduction

In this chapter, we discuss the compression of 4D multi-temporal HSIs using extended prediction based method. Multi-temporal HSIs are a set of images captured over the same location at different timestamps, also called time-lapse HS imagery. It is used to observe, detect, and analyze the changes in an area over a period of time. It can be considered a 4D data cube as represented in Figure 3.1, where x represents the number of pixels in each row, and y represents the number of pixels in each column forming two dimensions spatial information. z is the number of bands/channels in each image as the third dimension representing spectral information, and t is the number of timestamps or temporal information expressed in the fourth dimension. The size

of multi-temporal HSI depends mainly on the number of timestamps and may vary from a few GBs to TBs in some cases. Such extensive size data incorporates many challenges in storage, transmission, and processing of the images primarily in remote sensing applications. Data compression techniques play an essential role in reducing the enormous size of multi-temporal images. Historical data plays a vital role in many HS applications like change detection, object identification, monitoring & surveillance system, and medical applications like abnormality detection [146]. The large size of 4D HSI possesses a threat to these applications, it affects the performance of data center in keeping a record of any location for an extended period. In recent years, compression of 4D HSI has been adapted as an extension of existing 3D compression algorithms by including temporal dimension. Existing works have used prediction-based lossless compression to reduce this large size.

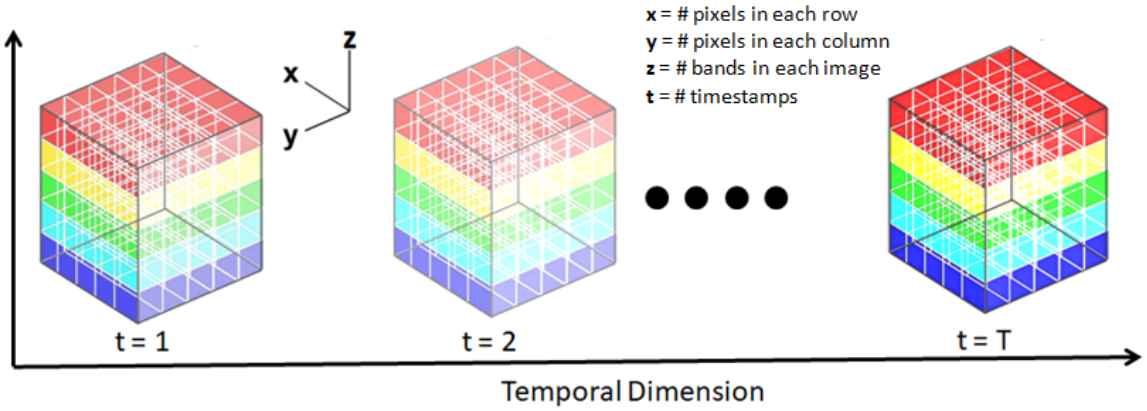

Figure 3.1: Multi-temporal Hyperspectral image

In 4D time-lapse imagery, efficient compression performance has been achieved by using different prediction models for spatial, spectral, and temporal correlation. Shen et al. [76] used this logic for the lossless compression of 4D HSIs using prediction by CLMS filter for FL predictor. Existing FL predictor [120] was extended to 4D data compression by introducing a linear prediction model for spectral and temporal correlation. In this chapter, we extend the RLS filter [147] based HSI compression to compress multi-

temporal images and included temporal bands in the prediction and weight calculation of the RLS learning algorithm. The next section discusses proposed algorithm in detail. We consider modified use of RLS filter to compress multi-temporal HSI along with identification of the optimum number of spectral and temporal bands for prediction.

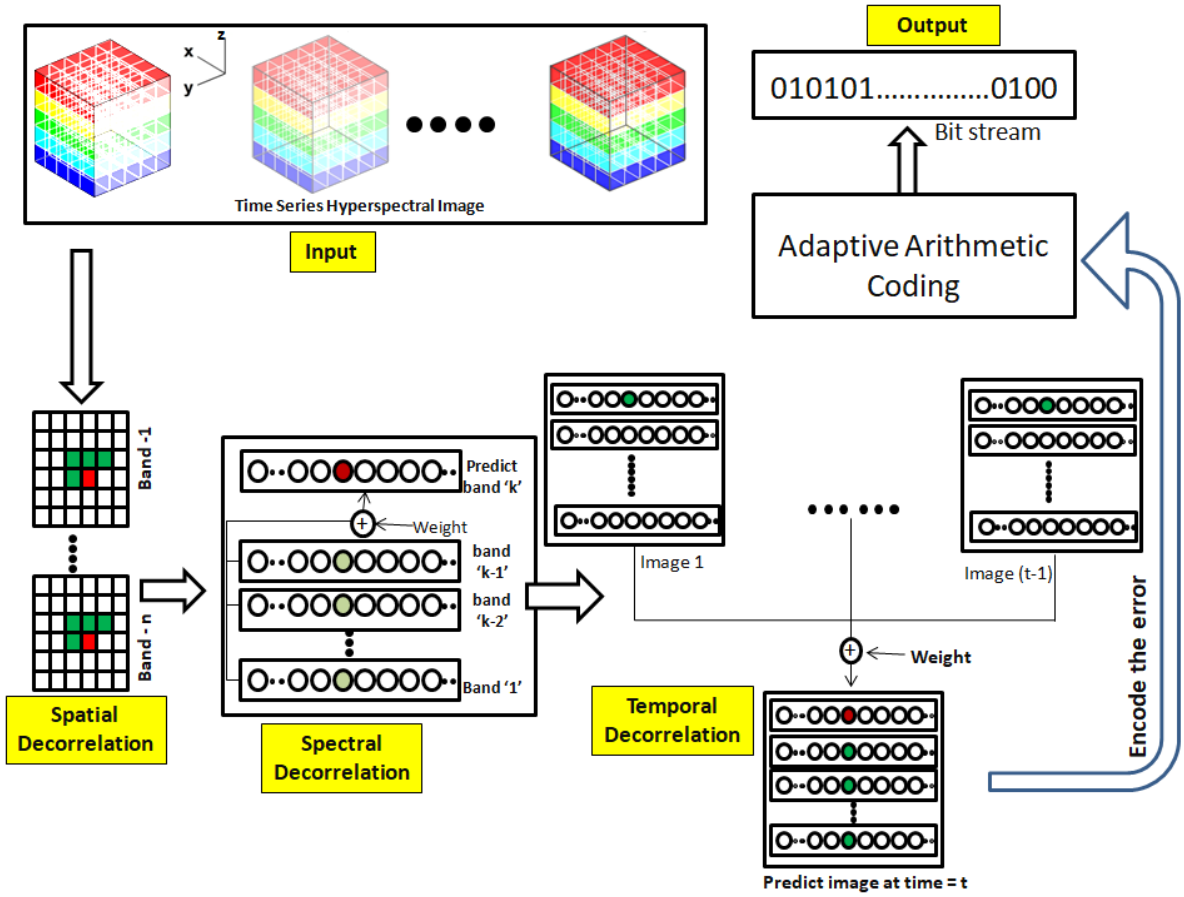

Figure 3.2: Framework of proposed algorithm

3.2 Proposed Methodology

Prediction-based compression has been used for near accurate prediction of pixels by removing correlations in all the four dimensions. We have used RLS filter to calculate weight of predictions at the compression end and entropy coding encodes the error image. The same algorithm has to be used with the same parameters at decompression end during reconstruction after the inverse entropy coding decodes the error image. A

pictorial representation of the algorithm is given in Figure 3.2 that explains the essential steps. The main objective of the proposed algorithm is to reduce the data required for transmission or storage in the form of bit per pixel per band. Four significant steps of the algorithm, say, spatial decorrelation, spectro-temporal decorrelation, calculating prediction coefficients, and entropy coding have been discussed in the following subsections.

Spatial decorrelation

Remote sensing images carry a large amount of information in each pixel. Ground mapping of these pixels has suggested a strong correlation that exists in an image. Pixels in the neighborhood of a pixel share similar properties. In prediction-based algorithms, this neighborhood is not equal to the typical one used in other image processing techniques (9×9 pixels with the target pixel at spatial location $(0, 0)$).

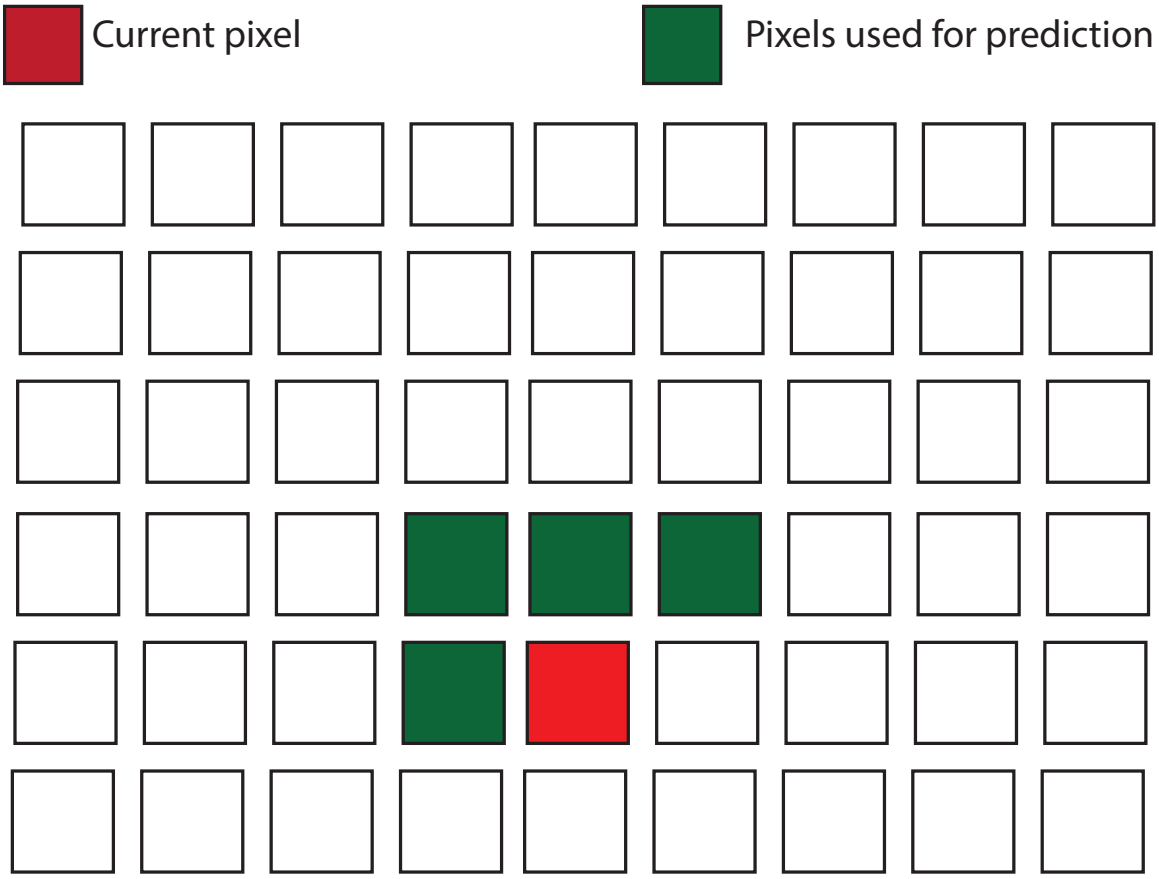

Figure 3.3: Neighboring pixels used for spatial decorrelation

In HSI, it is defined as the n-distance pixels from the target pixel that have already been predicted, where $n \epsilon \{1, min(R, C)$, C = number of columns, and R = number of rows, in a band. It can be explained using Figure 3.3, where the target pixel is shown in red color, and the one-distance neighboring pixels are represented in green color. Spatial decorrelation can be stated as the process of removal of spatial correlations existing in a band of HSI. It is restricted to only one band, where the target pixel has to be predicted. In the proposed algorithm, the value of n equals 1, i.e., four previous pixels have been used for spatial decorrelation.

The spatial decorrelated image $I'(R, C, B)$ can be calculated from input image $I(R, C, B)$, where R = number of rows, C = number of columns in a band, and B = total number of bands in an HSI by the following steps.

1. Calculate arithmetic mean ($\mu(x, y, z)$)of the pixels in its local neighborhood using equation 3.1.

$$\mu(x, y, z) = \frac{[p(x, y-1, z) + p(x-1, y, z) + p(x-1, y+1, z) + p(x-1, y-1, z)]}{4} \tag{3.1}$$

   where $p(x, y, z)$ be the current value of a pixel at (x, y) position on the z^{th} band.

2. Subtract arithmetic mean from the original value of the pixel at position (x, y, z) by using equation 3.2.

$$p'(x, y, z) = p(x, y, z) - \mu(x, y, z) \tag{3.2}$$

3. Repeat the same process for all the bands of HSI, forming a 3-D matrix of size ($R \times C \times B$). This $I'(R, C, B)$ matrix is used as an input signal to the second step of the algorithm, i.e, spectro-temporal decorrelation.

Spectro-Temporal Decorrelation

Multi-temporal HSIs store a broad set of spectral and temporal information. Due to similar reflectance properties of similar materials, the proposed algorithm uses the

strong spectral correlation in the neighboring bands to predict the pixels of target band. If the only spectral correlation is considered, spatially decorrelated pixels of target band (say, b) are generated with the help of $b-1$ already predicted bands at the compression or decompression end. The first band (b=1) of each HSI should be transferred to the decoder without any change. Similarly, temporal correlation from 4D HSI has been removed by using temporal information of the already predicted sequence of images. A target sequence (say, k) is generated from $k-1$ series of pixels at the same spatial and spectral positions. The first reference image (at k=1) is stored or transmitted after spectral and spatial decorrelations.

In this technique, we have combined the pixels from spectral and temporal bands to remove spectral and temporal correlations together, as done by Shen et al. [120]. Thus, the name spectro-temporal decorrelation. It is extracted by generating the value of target pixels from the linear combination of predicted pixels and the weight matrix updated by the RLS filter. The process is explained in the following steps.

1. Take multiple spatial decorrelated HSIs from different time stamps represented by $I_t'(R, C, B)$, where t represents image at time t.
2. Convert the 3D tensor, $I_t'(R, C, B)$, from the spatial decorrelation step into a 2D array of size ($R \times C$, B), by flattening one band's elements into a row
3. Predict the pixel $p_t(x, y, z)$ using equation 3.3

$$p_t(i, z) = \omega^T * p_{t-1}'(i, z) \tag{3.3}$$

where, $p_t(i, z)$ is the predicted pixel value at position (i, z) and at time t; $i \in [1, R \times C]$, $z \in [1, B]$.

ω^T is the transpose of weight vector of length $(b+k)$, where b and k are the number of spectral and temporal bands used for prediction.

The value of vector $\boldsymbol{p'_{t-1}(i,z)}$ is represented using equation 3.4.

$$\begin{aligned}\boldsymbol{p'_{t-1}(i,z)} = \Big[&p'_t(i,z-b),\ldots,p'_t(i,z-3),p'_t(i,z-2),p'_t(i,z-1),\\ &p'_{t-1}(i,z),p'_{t-2}(i,z),p'_{t-3}(i,z),\ldots,p'_{t-k}(i,z)\Big]\end{aligned} \tag{3.4}$$

4. Repeat the step 3 for each pixels of all HSIs in time series data.

Prediction-based on RLS filter

The weight matrix ω^T in equation 3.3 is generated as a function of RLS filter that learns adaptively from the loss/error and is updated after every step reducing the loss. The proposed method's pseudocode is given in Algorithm 3.1, along with the initial value of variables and various representations. The optimization phase of the algorithm is based on minimizing the cost function of RLS filter. Value of δ should be as small as possible to reduce the effect of previous pixels in the weight matrix. In this algorithm, we have selected δ=0.0001 to improve the sensitivity of filter.

Entropy Coding

The prediction coefficients from the extended RLS predictor are not the same as the input image due to the presence of loss function of the filter. The difference between spatially decorrelated image and prediction coefficients is transmitted to the receiver for decompression to make the compression lossless. This difference is called a residual error image, which can be stored in a comparatively lower number of bits per pixels per band, thus giving a better compression performance. Entropy coding technique is used to encode the residual error in predictive compression. Adaptive Arithmetic Coder (AAC) and Golomb Rice Coding (GRC) are two famous entropy coder used to compress HSIs. A lossy coding technique like EZW have also been used to transmit the error via noisy channels [148]. We have used AAC for entropy coding of error matrix that can be sent or stored with a comparatively reduced size. The decompression stage is an exact reverse of the compression procedure given in this article.

Algorithm 3.1: Extended RLS Prediction Algorithm

Result: Encoded bit-stream of prediction error

Input: Sequence of HSIs $I_T(R, C, B)$ of size $(R \times C \times B)$, where number of rows $= R$, number of columns $= C$, number of spectral bands $= B$, and number of temporal sequences $= T$

Representation: $p'_{t-1}(i, z) = [p'_t(i, z-b), \ldots, p'_t(i, z-1), p'_{t-1}(i, z), \ldots, p'_{t-k}(i, z)]]$ is spatial difference input vector, $\omega(i) = [\omega_1(i), \ldots, \omega_{b+k}(i)]$ is weight matrix, $K^T(i)$ is the Kalman Gain, and $e_t(i, j)$ represents prediction error

Initialization: Let $M(0) = \delta I_m$, $\omega(0) = [0]$, $t = 1$, where $\delta = 0.0001$ and I_m is the identity matrix of order m;

for *z = 1 to B* **do**
  for *y = 1 to R* **do**
    for *x = 1 to C* **do**
      Calculate $p'(x, y, z)$ using equation 2
    end
  end
  Convert $I'_t(R, C, B)$ to 2D array of size $(R \times C, B)$ `// by flattening elements of one band into a row of 2D array`
  for *i = 1 to* $R \times C$ **do**
    Calculate $p_t(i, z)$ using equation 3;
    Calculate the residual error $e_t(i, j) = p'_t(i, z) - p_t(i, z)$;
    $K^T(i) = \frac{M(i-1) \cdot (p'_{t-1}(i,z))^T}{1 + (p'_{t-1}(i,z)) \cdot M(i-1) \cdot (p'_{t-1}(i,z))^T}$;
    $M(i) = M(i-1) - K^T(i) \cdot (p'_{t-1}(i, z)) \cdot M(i-1)$;
    `// Calculate gain`
    $\omega(i) = \omega(i-1) + K(i) \cdot e_t(i, j)$
    `// Update weight`
  end
  Send the error $e_t(j)$ to the adaptive arithmetic encoder;
end

3.3 Experiment results

This section provides simulation results of the proposed algorithm and a discussion on various outcomes from those results. The following sub-sections contain detailed information about the dataset used, implementation environment, and simulation results.

Dataset

The algorithm was evaluated on the time-lapse HS imagery dataset [149] available in the public domain.

Figure 3.4: Representation of dataset used in the experiment

Table 3.1: Description of the dataset used to evaluate the algorithm

Dataset/ Attributes	Rows	Columns	Bands per image	Temporal Sequences
NOGUEIRO	1024	1344	33	9
GUALTAR	1024	1344	33	9
LEVADA	1024	1344	33	7
SETE FONTES	1024	1344	33	8

It consists of a total of 4 sets of multi-temporal HSIs of natural scenes with changing illumination [150]. Figure 3.4 shows RGB conversion of all the scenes used in this experiment. Each set of HSIs has been acquired at a time interval of 1-hour. The number of rows is equal to 1024, the number of columns in these images is 1344, and the number of bands in each image is equal to 33, with a bandwidth of 10 nm

ranging from 400-720 nm. Each pixel is represented using 12 bits, saved in MatLab file format, named as SCENE_TIME, e.g. nogueiro_1140.mat. Each pixel value represents spectral radiance in $Wm^{-2}sr^{-1}nm^{-1}$ units. Size of each nine frame dataset = $1024 \times 1344 \times 33 \times 12 \times 9 = 584.72$ MB. Its details are available in Table 3.1.

Implementation environment and evaluation metrics

The proposed algorithm, along with the state-of-the-art techniques like Change Detection [118], CLMS [76], and Extended Fast Lossless (FL) [120], has been tested on a personal computer powered by Windows 10 platform having Intel(R) Core(TM) i3-6006U CPU @ 2.00 GHz, x64-based processor with a 4 GB RAM. Algorithms are implemented in Python 3.7 language using various standard libraries like numpy and scipy to perform vectorized parallelization by itself. The algorithms have been evaluated by comparing the maximum bit-rate obtained from resultant bit-streams. Bit-rate can be defined as the number of bits required to represent a pixel. It is measured in bpp, inversely related to compression performance, i.e., lower the bit-rate better is the performance.

Results

Compression performance results have been calculated by taking an average of bit-rates obtained upon executing the algorithm multiple times over all the images of a scene to remove the redundancy and machine error. The optimal number of spectral bands used for prediction in step 2 of the proposed algorithm is chosen to be '4' after multiple hit-and-trial experimentations. Similarly, the optimal number of temporal bands used in prediction is also selected as '4'. Table 3.2 represents the algorithm's performance on increasing number of temporal scenes for the prediction of target pixels. It can be observed that global optima for compression performance is obtained when the number of temporal bands = 4 for Nogueiro, Gualtar, Sete_Fontes, and Levada dataset.

Table 3.2: Compression Performance for various values of temporal prediction bands (Bit per pixel)

Dataset/# Temporal Bands	1	2	3	4
NOGUEIRO	4.1296	3.7961	3.4426	2.9001
GUALTAR	4.5323	4.4798	3.9452	3.0052
LEVADA	3.9582	3.7928	3.7094	3.5352
SETE FONTES	5.7632	5.7494	5.4291	4.8738

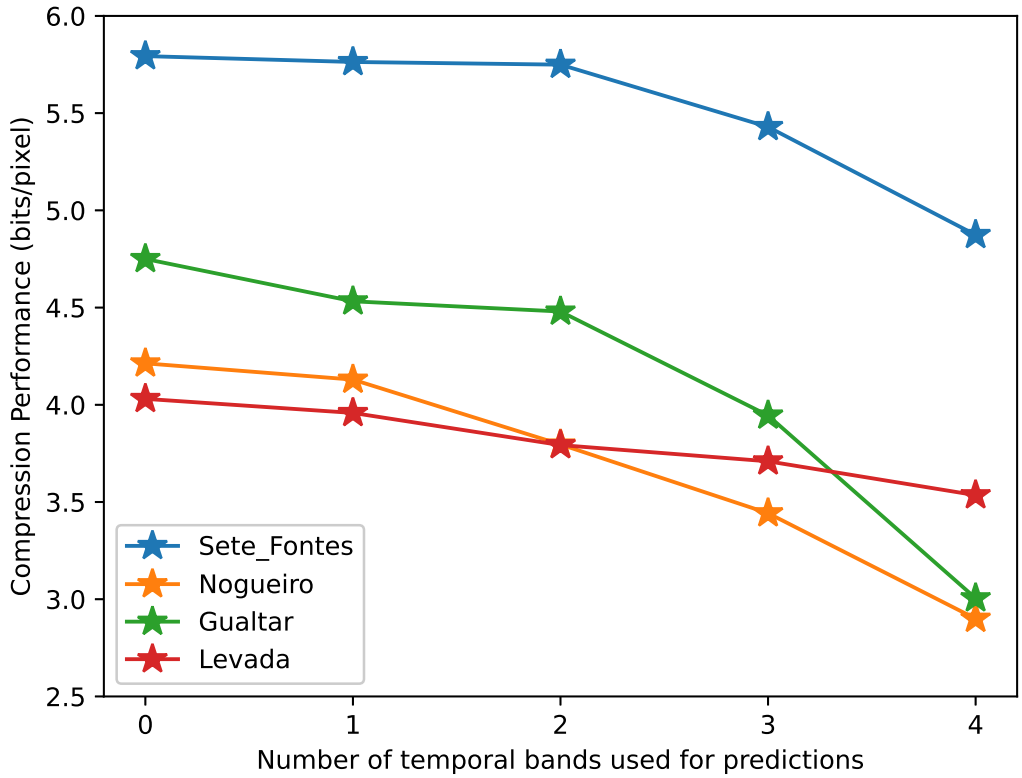

Figure 3.5: Effect of increasing temporal bands on compression performance

Graphically, the same information is represented in Figure 3.5 for individual datasets. The graph shows near similar properties for Nogueiro, Gualtar, and Sete_Fontes, while the curve is more linear for the Levada dataset. The expected increase in performance is also lower than its other counterparts since temporal correlation for one image can be different for the other. This analysis showed that if the optimal number of prediction

bands is fixed beforehand, it may result in a non-optimal solution. So, the number of prediction bands should be selected only when the results of some scenes of the image are obtained.

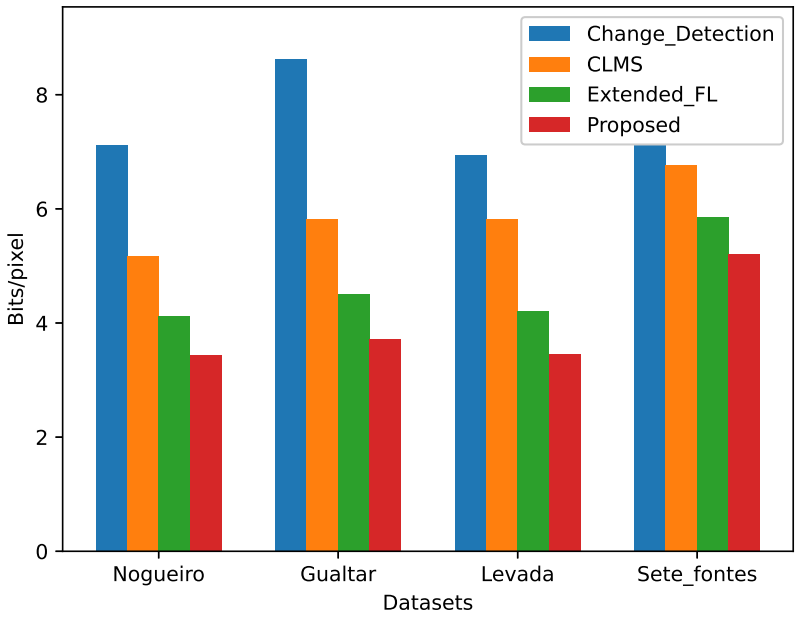

Figure 3.6: Original and reconstructed HSIs of Nogueiro, Gualtar, and Sete_Fontes datasets

The effectiveness of including temporal bands for predicting pixels using the RLS filter has also been shown experimentally in Table 3.3. We have executed different steps of our algorithm on each image of the dataset and calculated the compression performance after each step.

Table 3.3: Compression Performance for different steps of the proposed algorithm (Bit per pixel)

Dataset	Unmodified	Spatial decorrelation	Spectral decorrelation	Temporal decorrelation
NOGUEIRO	9.0284	7.1256	4.4806	3.4426
GUALTAR	9.7200	8.0550	4.6473	3.7177
LEVADA	9.4003	7.1107	4.5441	3.4499
SETE FONTES	10.9340	9.4090	6.2650	5.2031

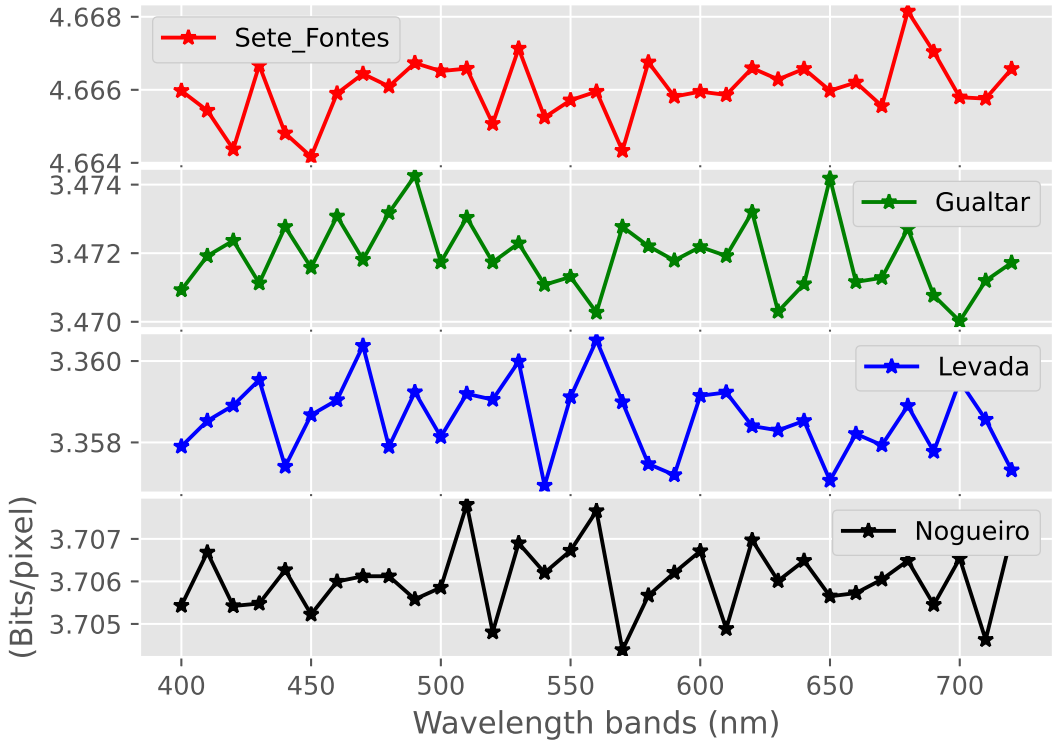

Figure 3.7: Comparison with state of the art algorithms

Table 3.4: Compression performance for standard datasets by different algorithms (Bit per pixel)

Dataset/Algorithm	Change Detection	CLMS	Extended FL	Proposed Method
NOGUEIRO	7.1077	5.1735	4.1127	3.4426
GUALTAR	8.6224	5.8268	4.4990	3.7177
LEVADA	6.9476	5.8191	4.2131	3.4499
SETE FONTES	9.0845	6.7651	5.8461	5.2031

Also, the images have been compressed without including temporal bands in prediction, i.e., removing only spectral correlation. Its result is represented under the column named Spectral Decorrelation. Here, an unmodified image means the original image has been entropy-coded without any decorrelation, and this experiment can visualize the efficiency of AAC. It reduces the number of bits required to store one pixel by 3

bits in Nogueiro, by 2.7 bits in Gualtar, by 2.6 bits in Levada, and by 1.07 bits in Sete_Fontes dataset, since the original image uses 12 bits per pixel. Number of bits per pixel is defined as amount of memory (in bits) required to store one pixel. It is calculated by the ratio of the total size of compressed image to total number of pixels in the image. It can be controlled using compression ratio or the ratio of size of original image (in bits) to the ratio of size of compressed image (in bits). Similarly, the effect of removal of spatial correlation, spectral correlation, and temporal correlation can be observed. The results indicate that spectral correlation is the most influential one. Figure 3.6 represents the original and reconstructed image of band number 28 of Nogueiro, Gualtar, and Sete_Fontes dataset.

Compression performance obtained after executing the state-of-the-art algorithms and proposed algorithm on all the datasets is given in Table 3.4. The proposed method outperforms other algorithms for compression of time-lapse HS imagery in all the datasets. The same has been shown graphically using a bar chart in Figure 3.7. It can be observed from the graph that the proposed algorithm provides a compressed image with minimum bits per pixel. The change is significant due to more number of bands per image, which in turn has a large number of pixels per band. The results are obtained by taking the arithmetic mean of bit-rate of all the scenes in all the algorithms.

The effect of decorrelation on a particular spectral band of an image can't be observed as the performances shown in the above results are calculated as the arithmetic mean value from all bands. Another experiment was conducted to observe the improvement on specific bands. Thus, we also calculated the number of bits required to represent a pixel in different image bands. The bit-rate for all 33 bands of the image is illustrated in Figure 3.8. Wavelength range is used in place of band number in the graph for better understanding. It can be observed that the value of bit per pixel is changed only in third place after the decimal. These results can be used to determine the strength of correlation in a band of any scene of multi-temporal HS imagery.

Figure 3.8: Compression performance for different spectral bands

3.4 Conclusion

In this chapter, the prediction-based compression technique has been used to reduce the size of multi-temporal HSI. It is a lossless compression in which the pixel values of an image are predicted from the combination of already predicted pixels of spectral and temporal bands. The initial condition being the first band of the input image, is transmitted directly to the decompression end. The coefficients are predicted using the weight matrix of recursive least square filter. The experiments have been performed on

four time-lapse HS imagery dataset having 7, 8, 9, 9 scenes, respectively, available in the public domain. Simulation results revealed the efficiency of the proposed method over the existing state of the art algorithms. The experiment also calculates the strength of individual correlations existing in the datasets and the number of optimal spectral and temporal bands used in prediction.

In the future, we plan to develop an automated model for the entire procedure of acquisition, pre-processing, compression, and decompression. We will also develop a model to select the number of temporal prediction bands automatically for a random dataset as the case of spectral bands. Furthermore, the effect of the presence of extensive sequence (counting to 100s) of temporal HSIs on compression performance has to be evaluated.