remote sensing

Article

Hyperspectral Image Compression Using Vector Quantization, PCA and JPEG2000

Daniel Báscones *, Carlos González and Daniel Mozos

Department of Computer Architecture and Automatics, Computer Science Faculty, Complutense University of Madrid, 28040 Madrid, Spain; carlosgo@ucm.es (C.G.); mozos@ucm.es (D.M.)

* Correspondence: danibasc@ucm.es; Tel.: +34-91-394-7581

Received: 4 May 2018; Accepted: 5 June 2018; Published: 8 June 2018

Abstract: Compression of hyperspectral imagery increases the efficiency of image storage and transmission. It is especially useful to alleviate congestion in the downlinks of planes and satellites, where these images are usually taken from. A novel compression algorithm is presented here. It first spectrally decorrelates the image using Vector Quantization and Principal Component Analysis (PCA), and then applies JPEG2000 to the Principal Components (PCs) exploiting spatial correlations for compression. We take advantage of the fact that dimensionality reduction preserves more information in the first components, allocating more depth to the first PCs. We optimize the selection of parameters by maximizing the distortion-ratio performance across the test images. An increase of 1 to 3 dB in Signal Noise Ratio (SNR) for the same compression ratio is found over just using PCA + JPEG2000, while also speeding up compression and decompression by more than 10%. A formula is proposed which determines the configuration of the algorithm, obtaining results that range from heavily compressed-low SNR images to low compressed-near lossless ones.

Keywords: hyperspectral image compression; dimensionality reduction; PCA; vector quantization; JPEG2000

1. Introduction

Images have been used since their inception to study different subjects of interest. Information can be remotely captured (with the only physical interaction being at most lighting the subject) and then analyzed by powerful algorithms to extract information the naked eye cannot see.

Traditionally, three different wavelengths have been captured per pixel, the building blocks of an image. This is enough to represent all the colors a human can see, since we perceive different colors as a mixture of red, green, and blue. Hyperspectral images extend this concept by sampling hundreds of wavelengths per pixel, making what is called a spectral signature, ranging from below the infrared to above the ultraviolet.

Spectral signatures greatly improve remote sensing capabilities, and have been used to identify materials within an image [1], for agriculture [2], or even for cancer detection [3].

These studies greatly benefit from the massive amounts of data that can be captured. A single image is usually millions of pixels in size, with each containing a couple hundred samples [4]. When dealing with this data, a bottleneck arises around storage, limited in planes and satellites, where capture devices are usually mounted on.

A viable solution is to compress the images, which can be done in a lossy or lossless way [5–7]. If all of the information is of critical importance, and any distortion to the original image could cause errors further down the line, lossless compression is the way to go. For most purposes, however, a moderate loss can be allowed, achieving much higher compression ratios.

A great amount of research has focused on hyperspectral image compression [7]:

- Traditional image compression algorithms have been applied per band [8]. A band is a two dimensional matrix that groups all of the samples of the same wavelength, one per pixel. Each band was compressed as a grayscale image, using different algorithms.
- Ideas from image compression have been extended from 2D to 3D space [9,10], adapting to the 3D-like matrix structure of hyperspectral images.
- Dictionary like methods have been proposed [11] in which spectral libraries are used to aid with compression.
- Dimensionality reduction has been used in the spectral direction, along with image compression on each of the reduced planes [12].

In this paper, a novel technique for lossy compression, composed of multiple steps, is explored:

- First, nonlinear dimensionality reduction is applied in the spectral direction. This is accomplished in two steps [13]: pixels are clustered using the nearest neighbor algorithm, and then Principal Component Analysis (PCA) is performed on each cluster with the same number of target dimensions. The aim of this step is to eliminate spectral correlation, transforming the original bands to Principal Components (PCs). This transform has been shown to preserve relevant image characteristics in the transformed space [14], which is an interesting feature for compression.
- Secondly, each PC is individually compressed using two techniques found in the JPEG2000 standard [15]: Wavelet transforms and Block Coding. Compression is tuned for each PC, allocating more bit depth to the most important PCs.
- Finally, all of the data is packed into the compressed stream, along with the necessary metadata to decompress it. This metadata notably includes Wavelet and Block Coding settings, as well as all of the inverse PCA transforms and the pixel classification, which is arithmetically encoded.

We base the results of our algorithm on two metrics: Signal to noise ratio (SNR) and compression ratio. A third metric is derived from those: distortion-ratio, which measures, given a compression ratio, how good the quality (SNR) is. Thus, it cannot be measured directly, but when given two images compressed at the same ratio, we say that the one with higher SNR has a higher distortion-ratio metric. Results show that an image can be compressed to around $1/200$ of its original size with no perceivable loss of information to the naked eye and up to 2000 times while preserving good SNR.

The rest of the paper is organized as follows: We first take an in-depth look at the algorithm, to see exactly how every step works. Then the implementation is explained, and the results are shown and compared with [12]. Finally, conclusions are drawn from the results, and we hint at possible future research lines in which we think compression could be improved even more.

2. The Algorithm

Hyperspectral images H can be seen as 3-dimensional cubes of data of size $N_x \times N_y \times N_z$. The x and y dimensions are spatial, while the third, z, is spectral and accommodates the different wavelengths. The nature of hyperspectral images makes two types of correlation appear:

- First, we have spectral correlation: bands that are close together have similar values in the same pixels. This is due to the spectrum of the different materials being a continuum, and to our samples being close together ($<$10 nm differences in wavelengths).
- On the other hand we have spatial correlation: neighboring pixels are expected to have similar values, since image features will usually span regions many pixels in size.

Compression in general takes advantage of redundancies in the input data. In this case the spectral redundancies are more pronounced than the spatial ones, and treating them independently can improve our distortion-ratio performance.

Traditional image compression algorithms, when applied per band, do not take into account the spectral correlation, and lose compression ratio. Extensions to the 3-dimensional space assume the correlation is the same both in the spectral and spatial direction, which is not usually the case, and either distort the image more (if the large correlation is assumed) or achieve worse compression ratio (if the small correlation is assumed).

Tailoring the compression algorithm to the specifics of the input data can improve its performance, and that is the core of our approach: extending ideas found in the literature to better suit the nature of hyperspectral data.

Our starting point is the work done in [12]. The idea is to combine dimensionality reduction (PCA) along the z-axis with JPEG2000 compression in the (x, y) plane. Every plane (the reduction will not usually collapse z to a single point) is individually compressed, since the spectral correlation is mostly gone after reduction.

We aim to improve it in two ways: First, we add a vector quantization step prior to the PCA reduction to better adapt to local spectral statistics, by independently reducing clusters of pixels with similar characteristics. Secondly, we do not use a commercially available JPEG2000 solution. Instead, we have followed the standard and implemented the algorithm with a few variations, adapting to our needs (e.g., eliminate certain markers that are useful for partially decoding images, but that bloat image size). Our modified version follows these steps, which can be seen in Figure 1.

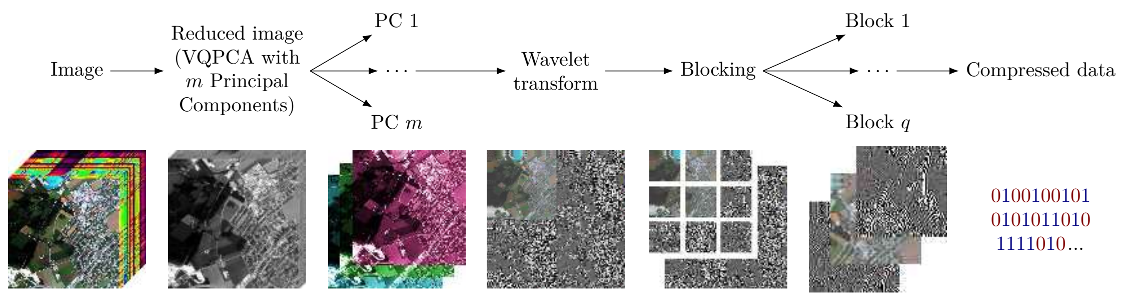


Figure 1. General flow and steps of the algorithm. Steps after separation in PCs are applied per PC.

2.1. Dimensionality Reduction

Linear reduction (such as PCA) assumes a linear correlation model. This works for most cases, and PCA is well-known and used in the literature, but it can be improved. In our case, we decided to use VQPCA (Vector-Quantization PCA) [13].

VQPCA is a piecewise linear reduction model. It is a midway point that benefits from the simplicity of linear models, but offers results close to nonlinear ones. It works by quantizing the pixels (vectors) obtaining a user-defined number of representatives, which partition the z-dimensional space into a Voronoi [16] diagram. Each Voronoi cell is independently reduced by a different PCA, trained using only the points of its own cell. With this, we are exploiting pixel spectral similarities twice: similar pixels are grouped together in Voronoi cells, and then a PCA decides which directions maximize variance within each cluster. The result is that rounding errors are reduced, since applying PCA to clusters of similar pixels ensures less variance, and thus better resolution in the quantization step (Section 2.3).

In our case, we use the K-Means [17] algorithm to group the pixels $p \in \mathbb{R}^n$ (so $N_z = n$) of the image H into c different clusters. This yields centroids $P_1, \ldots, P_c$ and sets $S_1, \ldots, S_c$ corresponding to the Voronoi cells of the Voronoi diagram with centers $P_1, \ldots, P_c$. We then perform PCA on each of the sets S_i, choosing a target dimension m for the reduced space. We obtain matrices $W_i \in \mathcal{M}_{m \times n}$ and $\bar{W}_i \in \mathcal{M}_{n \times m}$ that respectively project the original data onto the reduced space and immerse the reduced data into the original space. With this we obtain the reduced image H':

$$H' = \{W_i(p - P_i) \mid p \in H \cap S_i \quad i = 1, \ldots, c\} \tag{1}$$

with this approach, similar pixels are grouped together before projection, so the PCA algorithm can ignore the variance present between clusters, and better separate the pixels within. The cost is first: we need to store multiple PCA inversion matrices, and second: we also need to store the index i for which cluster each sample belongs to in order to invert the process when using $\bar{W}_i$. Compression is all about reducing size, and this actually increases it, so we might wonder where the benefit is. Since this algorithm achieves better separation than PCA, we might get the same SNR with VQPCA and m dimensions than with PCA and $m+1$ dimensions. The extra dimension saved will compensate the extra space needed by VQPCA.

Note that, in the case that only one cluster c is used, VQPCA is equivalent to PCA. This makes VQPCA a good choice since it can always fall back in a well-tested method in case the clustering does not improve compression. We will later see the effect of the number of clusters on the distortion-ratio performance.

2.2. Wavelet Transform

After reducing H, H' is now a cube of size $N_x \times N_y \times m$. We have removed the z-axis correlation, so now we will process each PC $B_j \in \mathcal{M}_{N_x \times N_y} \quad j = 1, \ldots, m$ separately.

A wavelet transform exploits spatial redundancies in a matrix to separate it into regions with different but consistent patterns, which can later be efficiently encoded. The wavelet transform selected is the CDF 9/7 from the JPEG2000 standard [18]. It will be applied to each PC independently, and we can configure how many times it is recursively applied. More passes will compress the image more, but will also produce results of a lesser SNR.

2.3. Variable Bit-Depth Quantization

Both the dimensionality reduction and the wavelet transform work on floating point numbers. When coding, it is easier to deal with integer arithmetic, since it usually takes up less space, and also because processing integers is faster. So, before coding, our data is quantized.

Quantization means choosing an integer interval ($\left[-2^{q-1}+1, 2^{q-1}-1\right]$ in this case), and mapping the input data to that interval, rounding off any extra decimals. The exponent q is what we call bit-depth. A larger bit-depth means better SNR, but also worse compression ratio. Since not all the PCs B_j have the same amount of information (PCs with lower j have more information about the input data) we can use different bit-depths $q_1, \ldots, q_m$ for the different PCs.

2.4. Coding

After quantization, the block coder from the JPEG2000 standard comes into play. Exploiting the patterns left by the wavelet transform, the image is coded using a binary arithmetic coder. The standard specifies a way of marking the compressed stream so that partial, lower SNR representations of the original data can be recovered without decompressing the whole image. For an image that is sent over the Internet this is useful, since previews can be obtained without receiving all of the data. Analyzing images requires the full picture to be available, so we have decided to not use standard-compliant JPEG2000 compression, but a custom format which eliminates all markers, resulting in a size reduction.

2.5. Metadata

Once coding is done, the original data H is compressed. However, to get the full picture we also need the metadata. Hyperspectral metadata usually contains geographical information, the wavelength of each band, sensor calibration settings and more. This is usually stored in plain text, but we also compress it by storing it in native raw values (byte, short, int, …).

On top of that, we of course need metadata about the algorithm itself. The different configuration parameters are saved along the way (bit-depth values, wavelength pass number, and VQPCA recovery metadata). These are all encoded raw except the dictionary which maps pixels to their Voronoi cell. This information, a matrix of indices $0, \ldots, c-1$, is arithmetically coded using a predictive model.

It works by predicting the index of a pixel based on already processed (in a line-by-line sweep of the image) neighboring pixels. If the prediction is correct, a special symbol 0 is encoded indicating a hit, while if the prediction fails, the full index $i = 1, \ldots, c$ is encoded. The predictive model simply uses the previous symbol to guess the current one.

With that, the full image is compressed.

3. Implementation

The algorithm has been fully implemented in Java, under a command line interface named Jypec (Java hYPErspectral Compressor). The code can be found on GitHub [19].

Both the dimensionality reduction and wavelength steps work on 32-bit float matrices. Intermediate accumulators and calculations use 64-bit double precision numbers. This was chosen since the raw image data, usually 16-bit integers, fit without loss on floats, taking up half the memory we would need if we used double precision. We found no significant difference in distortion-ratio performance when double precision was used throughout.

After quantization, the rest of the operations are done over q_j bit sign-magnitude integers for each PC B_j.

The nearest neighbors algorithm used comes from the SMILE [17] API. Unlike other implementations [20], it can be trained with a subset of samples speeding up computation times.

The PCA algorithm has been implemented from scratch, relying on EJML [21] for fast matrix multiplication.

The wavelet transform implements the lifting [22] scheme to accelerate execution, and coefficients are kept between ± 0.5 with renormalization procedures, to avoid coefficient runoff.

Quantization is done via a deadzone quantizer, which for a depth of q bits maps to $2^q - 1$ values. (One value is lost since the mapping interval for zero is twice as wide as the rest).

Finally, the image is divided in smaller blocks which are then coded according to the JPEG2000 standard. The partitioning is fully deterministic so as to not have to store positional information about the blocks.

We allow fine tuning of the different parameters of the algorithm to test different configurations and see which one obtains the best results.

- Subsampling factor t: Indicates the percentage of pixels used for dimensionality reduction training (which are randomly chosen). The lower it is the faster the algorithm performs (training costs for VQ and PCA increase quadratically and linearly, respectively, with t), but usually at the cost of distortion-ratio performance (see Section 4).
- Number of wavelet passes w: How many times the wavelet is recursively applied to each PC.
- Target dimensions m: Number of dimensions that we are reducing the original image to.
- Target bit depth q: Base bit depth used to process all of the PCs created after reducing H.
- Number of clusters c used for Vector Quantization in VQPCA.
- PC bit shave: This is how variable bit-depth is specified. For each PC, one can indicate how many bits to shave $(s_1, \ldots, s_m)$ from the target bit depth q, creating the different values for $q_1, \ldots, q_m = q - s_1, \ldots, q - s_m$.

4. Experimental Results

We are interested in obtaining a good distortion-ratio performance. That is, given a certain compression ratio we want the minimum distortion possible or, conversely, given a distortion we want the maximum compression ratio.

The metric used to measure distortion is the signal to noise ratio SNR (Equation (2)). This is used to measure the difference between the original image H^a and the result of compressing and then uncompressing it (H^b). SNR is based on the mean square error MSE (Equation (3)) between the samples h_k^a, h_k^b of the hyperspectral images H^a, H^b, both with $K = N_x \times N_y \times N_z$ samples.

$$SNR(H^a, H^b) = 10 \log_{10} \left(\frac{\sigma\left(H^a\right)^2}{MSE(H^a, H^b)} \right) \tag{2}$$

$$MSE(H^a, H^b) = \frac{\sum_{k=1}^{K} \left(h_k^a - h_k^b\right)^2}{K} \tag{3}$$

where $\sigma(H)^2$ (Equation (4)) is the variance of H, used to stabilize the metric when there is too much variance in the input image.

$$\sigma(H)^2 = \frac{\sum_{k=1}^{K} \left(h_k - \mu(H)\right)^2}{K} \tag{4}$$

with $\mu(H)$ being the mean value of the samples $h_k \in H$.

We can see that, under this definition, SNR is not symmetric. From now on, when we refer to an uncompressed image's SNR, H^a will always be the original image, allowing us to compare results in a fair manner.

Compression ratio is simply defined as the relation between the sizes of the original and compressed images:

$$ratio(H^a, H^b) = \frac{size(H^a)}{size(H^b)} \tag{5}$$

Our test image set is comprised of six different hyperspectral images [23], all with a bit depth of 16 bits, and of different sizes: SUW from the gulf of Mexico (360 $\times$ 320 $\times$ 1200); DHO from the Deep Horizon oil spill (360 $\times$ 320 $\times$ 1200), BEL from Beltsville, MD, USA (360 $\times$ 320 $\times$ 600), REN from Reno, NV, USA (356 $\times$ 320 $\times$ 600), CPR from Cuprite, NV, USA (224 $\times$ 614 $\times$ 512), and CUP (350 $\times$ 188 $\times$ 350), a corrected and cropped version of CPR.

Unless otherwise stated, the parameters from Section 3 are set as follows:

- t: We found no relevant effect on distortion-ratio when setting $t > 0.25$ so that is the value used throughout the experiments. It did lower slightly when $0.25 > t > 0.1$, and rapidly degraded after $t < 0.1$, so values under those should be avoided.
- The number of wavelet transform passes w is set to three, following the usual value set in the literature [9,10].
- The number of dimensions m and global bit depth q are not set. We begin by optimizing those in Section 4.1.
- No bit shaving is done at this point. Bit shave effects are studied later in Section 4.3.
- VQPCA is configured to use one cluster c (equivalent to PCA). This will later be optimized in Section 4.2.

4.1. Varying Dimensions and Bit Depth

We start by analyzing the effect of changing the dimensions (m) and bit depth (q) when compressing our image. All images present similar behavior when being compressed, so for brevity we show compression results of the Cuprite image in Figure 2a for SNR and in Figure 2b for compression ratio.

The idea now is to maximize SNR for a given compression ratio. We take the values from Figure 3a,b and create contour maps with 50 lines each. We then intersect each compression ratio contour line with all of the SNR contour lines, and keep the intersection(s) with the highest SNR line. In this way we create a map of points that gives us a configuration for the algorithm which maximizes SNR for a given compression ratio (see Figure 3a). The number 50 is chosen since it gives a good amount of points to later create a regression line, with more points not being significant for the results.

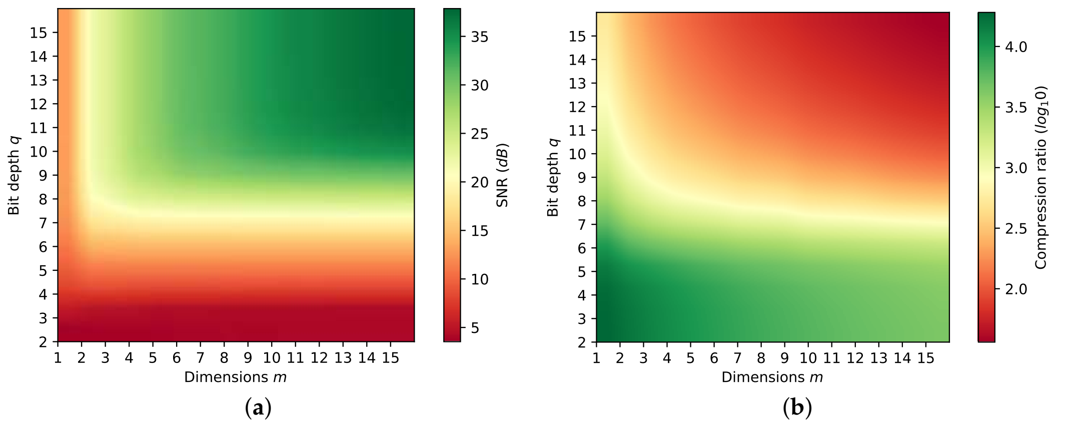


Figure 2. Signal Noise Ratio (SNR) and compression ratio results for the Cuprite image. (**a**) SNR as a function of the dimensions m and bit depth q. Note how, for shallow bit depths, m has little effect in SNR; (**b**) Compression ratio as a function of the dimensions m and bit depth q. An increase of any of them implies a reduction in ratio.

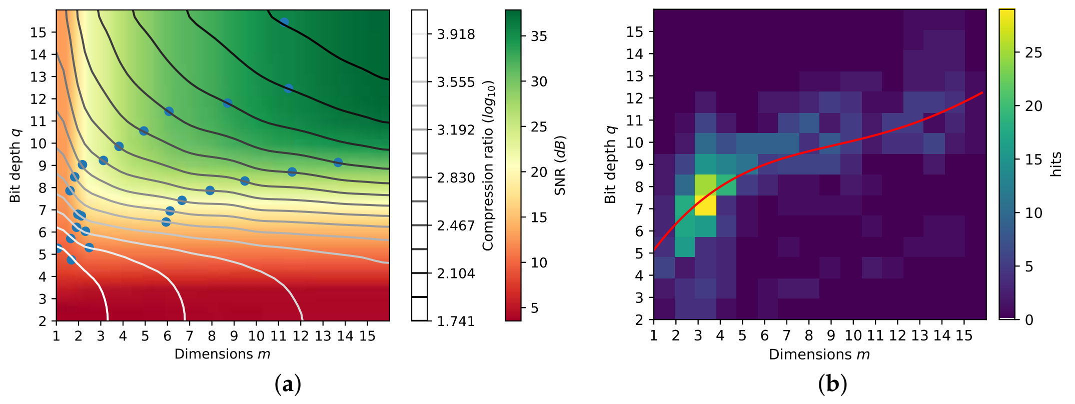


Figure 3. Optimal results for (**a**) the REN image and (**b**) the aggregate of all images. (**a**) Finding out the optimal points where SNR is maximized for a given compression ratio. (Note: not all 50 lines are shown for viewing clarity.); (**b**) Regression showing where the distortion-ratio performance is maximized. Each hit represents an optimal configuration for some image.

We repeat this process for every image in our test library, creating a 2D histogram showing what configurations have been optimal (in terms of maximum SNR for a given compression ratio) more times. Polynomial regression over this histogram gives us a path that, going from highest to lowest compression ratio, tells what configuration to use to get the optimal SNR for that compression ratio (See Figure 3b). We will refer to values from this regression (shown in Equation (6)) as the QR configuration, which gives us optimal values for q and m for an index j (which ranges from zero onwards).

$$QR(j) = \{q(j), m(j)\} = \left\{ \begin{array}{cc} 5+j & j \leq 3 \\ 9+\left\lfloor \frac{j-4}{3} \right\rfloor & j > 3 \end{array} , j+1 \right\} \tag{6}$$

4.2. Optimal Number of Clusters

Now we want to see the effect of changing the number of clusters c of the VQPCA algorithm, following the optimal distortion-ratio path given by the QR configuration. The general trend can be seen in Figure 4a, which shows points at which SNR is maximized given a certain compression ratio. An outlier, (the CPR image) which has a different behavior, is shown in Figure 4b. The rest of the images had behaviors following the general trend.

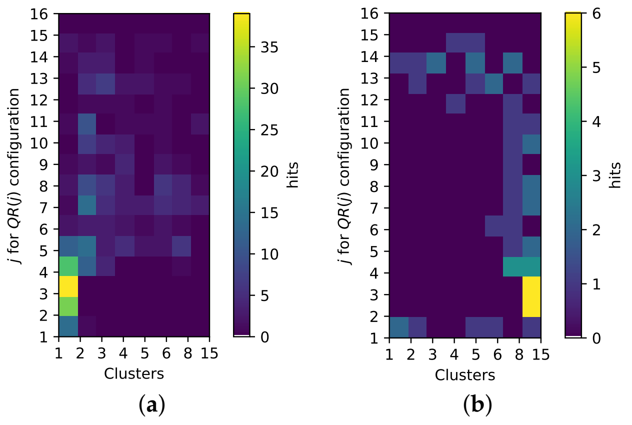


Figure 4. Comparison between the general trend and outlier behavior of the CPR image. Both figures obtained following the same technique used for Figure 3a,b. (**a**) General trend when changing the number of clusters and $QR(j)$ configuration. Results improve slightly with the number of clusters at high values of j; (**b**) Outlier behavior of the CPR image, which is better compressed at lower j values for $QR(j)$ when using a high amount of clusters.

In any case, we now add the parameter c to our QR configuration, creating the QRC configuration, as seen in Equation (7).

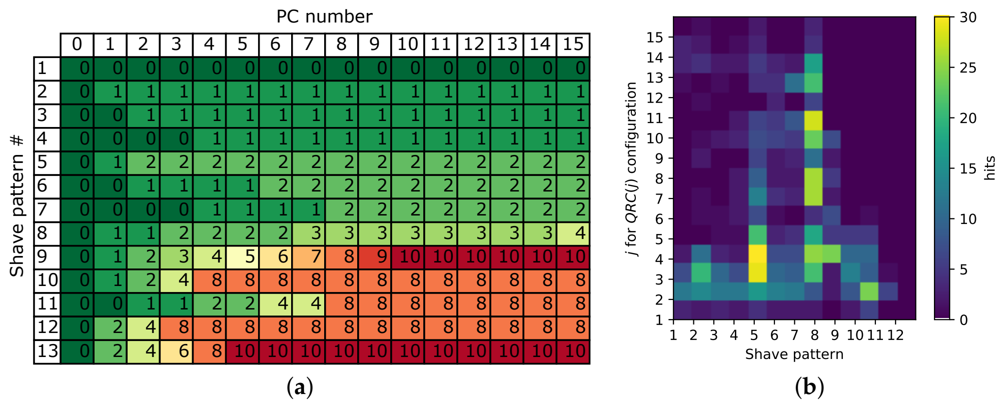
$$QRC(j) = \{QR(j), c(j)\} = \{QR(j), \lfloor j/6 \rfloor\} \tag{7}$$

4.3. Variable Bit-Depth

So far, every PC of the image has been compressed with the same amount of bits. Dimensionality reduction usually keeps more information in the first PCs, progressively adding less detail with subsequent ones. This suggests the possibility of allocating different bit depths $(q_1, \ldots, q_m)$ for the different m PCs.

The way we do this is by specifying the initial bit depth q, and then the bit shave values $s_1, \ldots, s_m$, yielding bitdepths $q_1, \ldots, q_m$ for all m PCs (see Section 3).

We have tried 13 different shave patterns, to see which one yields the best distortion-ratio performance. They can be seen in Figure 5a. Some have been selected based on their properties (e.g., pattern 8 halves the information stored after every 2^ith PC, pattern 9 halves the information stored on every PC), while others are just tests to see how they would perform.


PC number

Shave pattern #	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
5	0	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2
6	0	0	1	1	1	1	2	2	2	2	2	2	2	2	2	2
7	0	0	0	0	1	1	1	1	2	2	2	2	2	2	2	2
8	0	1	1	2	2	2	2	3	3	3	3	3	3	3	3	4
9	0	1	2	3	4	5	6	7	8	9	10	10	10	10	10	10
10	0	1	2	4	8	8	8	8	8	8	8	8	8	8	8	8
11	0	0	1	1	2	2	4	4	8	8	8	8	8	8	8	8
12	0	2	4	8	8	8	8	8	8	8	8	8	8	8	8	8
13	0	2	4	6	8	10	10	10	10	10	10	10	10	10	10	10

j for QRC(j) configuration
Shave pattern
hits
(a)
(b)

Figure 5. Shave bit patterns and their results when compressing. (**a**) Shave patterns and how many bits are shaved on each PC. For each shave pattern and each PC, the number of bits removed when quantizing are shown. Note that the number always increases since the PCs contain progressively less information; (**b**) QRC configuration vs. different bit shave patterns.

We compress our image test bench, varying the pattern as well as the QRC configuration, obtaining the heat map from Figure 5b, showing which configurations obtained optimal distortion-ratio performance more often. In this case, Patterns 5 and 8.

In Figure 6 we see the extent of the increase in distortion-ratio performance, relative to using no bit shaving. For the same SNR, we achieve compression ratios up to 1.8 times better when using the bit shaving pattern 8.

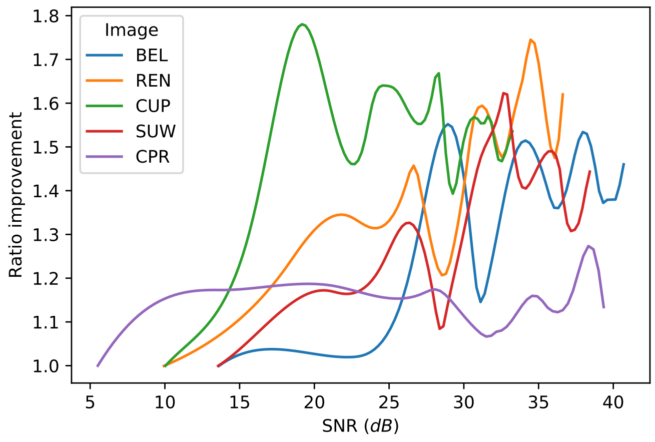


Figure 6. Relationship between the compression ratios of the compressed images at a given SNR when using pattern 8 to shave bits. DHO image is omitted since it was an outlier with about 8× improvement.

4.4. Final Results

In the end, we have optimized the compression parameters q, m, c and s_i, and we can define a configuration $F(j) = \{QRC(j), s_i(j)\} = \{q(j), m(j), c(j), s_i(j)\}$. Increasing j will result in an increase of SNR and a decrease of compression ratio, all while following the optimal distortion-ratio path.

$$F(j) = \{QRC(j), s_i(j)\} = \left\{ QRC(j), \begin{array}{cc} 0 & i = 0 \\ \lfloor \log_2 (j+1) \rfloor & i > 0 \end{array} \right\} \tag{8}$$

Figure 7 shows the results for our test image set, where we have compressed our images with configuration $F(j), j = 0, \ldots, 19$. With the exception of the CPR image, all the others follow a similar slope, but achieve the same qualities at significantly different compression ratios.

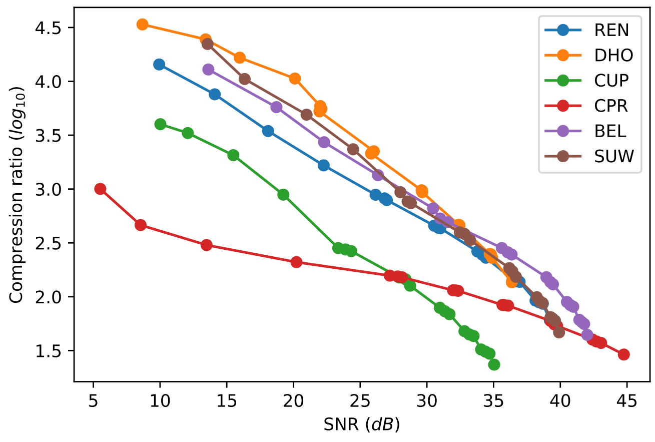


Figure 7. Distortion-ratio results for the optimal parameters from configurations $F(j), j = 0, \ldots, 19$.

Numbers are not always descriptive of the results, so, as an example, we include the results of compressing the SUW image. Figure 8 shows how the SNR progresses from left to right, decreasing the compression ratio at the same time. Depending on what we want to use the image for, we will have to choose which j to set.

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
SNR	13.56	16.34	20.98	24.47	28.01	28.55	29.12	32.80	33.09	33.41	36.31	36.55	37.06	38.55	38.77	38.94	39.47	39.66	39.88	40.13
C.R.	22221	10513	4902	2339	934.6	766.6	598.8	341.3	326.4	293.2	167.5	155.7	136.9	94.68	87.60	83.67	62.99	59.97	56.92	45.24
C.T.	7.94	8.14	8.26	8.95	9.45	9.75	10.10	10.35	11.43	11.79	14.88	15.07	16.86	21.40	22.15	22.50	28.08	29.01	33.44	38.74
D.T.	1.84	1.83	2.21	2.52	3.35	5.72	6.00	6.41	6.72	6.99	8.67	9.39	10.00	12.67	13.42	13.70	16.71	17.60	22.50	24.12

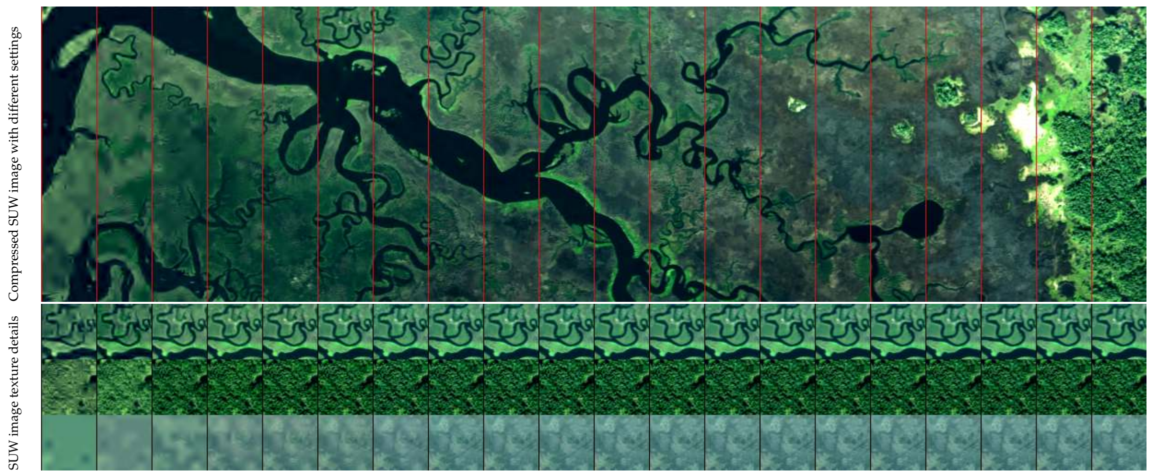

Compressed SUW image with different settings
SUW image texture details

Figure 8. Table shows compression results of the SUW image compressed with F^j configuration for each j shown. From top to bottom, rows are SNR, compression ratio (C.R.), compression time in seconds (C.T.) and decompression time in seconds (D.T.). The first image shows different parts of the SUW image, while the second image focuses on a comparison of three different spots with distinct textures.

5. Discussion

Parameter optimization results show that, for configurations F^j with low values of j, it is best to increase the number of dimensions m and bit depth q at the same rate, starting with $m = 1, q = 5$, suggesting that both have equal impact in distortion-ratio performance. After reaching the $m = 4, q = 8$ mark, it is best to increase m around three times as much as q to keep on the optimal path. Bit depth is thus important only up to a certain point, after which the added bits have little impact compared to adding more PCs.

With respect to c, it is best kept at 1 (where VQPCA is equivalent to PCA) when $j < 5$. After that, adding more clusters does slightly improve distortion-ratio performance. This is however image dependent, and a clear correlation was not found.

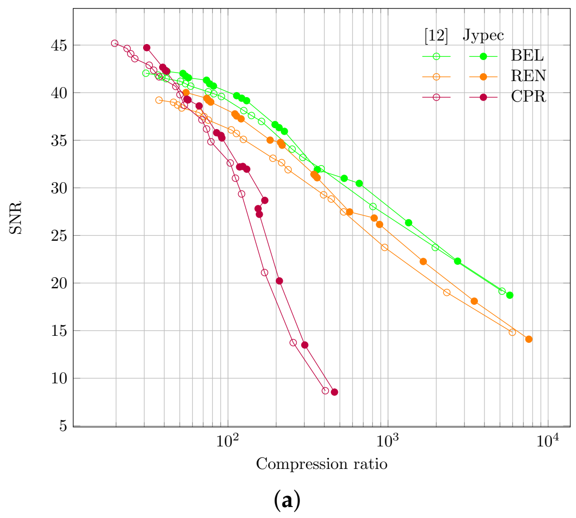
Bit shave is an important tool that highlights the fact that dimensionality reduction concentrates information in the first PCs. From our tests, Pattern 8 clearly achieved the best results, shaving one more bit every time the PC counter goes past a power of two. The amount of bits shaved, despite being small compared to other patterns, is enough to significantly reduce image size, while at the same time preserving good SNR. Compression is also faster when shaving since less bits are coded.

Configurations F^j offer great distortion-ratio performance across all test images. It is worth noting, however, that results are sometimes image dependent, so optimal parameter selection for a concrete image may vary slightly.

5.1. Comparison with Other Algorithms

Comparison of compression algorithms is a tricky subject, since it can only be done when using the same compression metrics and images. We have been able to compare our algorithm (Jypec) with the PCA + JPEG2000 approach from [12], showing an improvement in distortion-ratio performance, as seen in Figure 9.

Execution times (both in compression and decompression) have also improved (Figure 10). This is mainly driven by the bit shaving technique, which saves time when encoding the final bit stream as each band requires fewer bit planes to be coded.



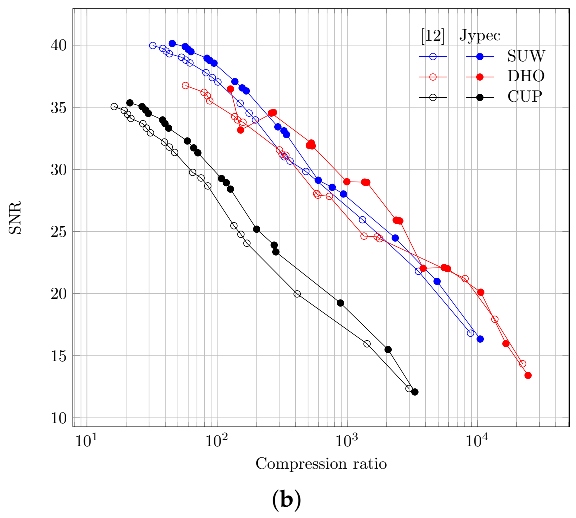


Figure 9. Distortion-ratio performance comparison between the work done in [12] and the algorithm developed here. Except in some isolated cases, Jypec achieves better distortion-ratio performance, having higher SNR for the same compression ratio and higher compression ratio for the same SNR. (**a**) Results for images BEL, REN and CPR; (**b**) Results for images SUW, DHO and CUP.

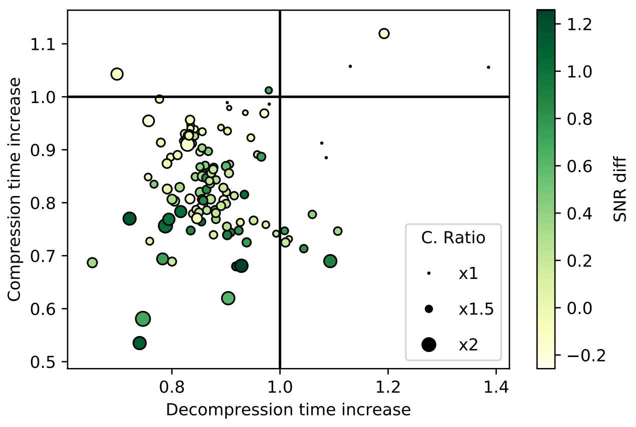


Figure 10. Results of executing PCA + JPEG2000 and Jypec over our test images with parameters from configurations $F^j, j = 0, \ldots, 19$. Shave bits and clusters were only applied for Jypec. Black lines indicate separation between faster decompression (left of vertical line) and faster compression (below horizontal line). Execution times clearly decrease for both compression and decompression. Compression ratio also improves consistently, being always better and almost double in some cases. SNR mostly improves, and where it does not, the increase in compression ratio makes up for it (see Figure 9).

6. Conclusions and Future Research Lines

A novel hyperspectral image compression algorithm is presented, based on the idea of first applying dimensionality reduction along the spectral axis, and then individually compressing each Principal Component (PC).

Vector-Quantization Principal Component Analysis (VQPCA) is chosen for reduction over the well-known PCA algorithm, which is a particular case of VQPCA. This method clusters the input pixels into groups, with each getting its own PCA fitted to improve separation.

After reduction, each of the resulting PCs are individually compressed using ideas from the JPEG2000 standard. A wavelet transform is applied, splitting the PC into four regions with increased local redundancy that allows for better compression, while losing minimal information in the process. A variable bit-depth quantizer transforms each PC from floating point to integer arithmetic, allowing PCs with more information to have greater bit-depths. Finally, a deterministic blocking algorithm, along with a custom version of the JPEG2000 block coder, encode and compress the information into the final bit stream, avoiding any markers that might bloat the final size.

Aside from compressing data, the image as well as the algorithm metadata are also compressed and embedded into the output stream, enabling decompression with just one file.

Results show an improvement in distortion-ratio performance over PCA + JPEG2000, with a 1 to 3 dB increase in SNR for the same compression ratio. This is mainly driven by the variable bit-depth, allocating more bits to the most important PCs. Compression and decompression times also exhibit a speedup of 16% and 12% respectively.

Jypec, the framework developed for testing these ideas, can be a great platform to expand this study. More compression flows can be easily explored in the future, especially by testing other nonlinear dimensionality reduction algorithms, or including different coding techniques aside from JPEG2000. Finally, it is clear that compression is highly dependent on the input data. Studies which optimize compression parameters to obtain the best distortion-ratio performance will be highly valuable to further enhance how we treat hyperspectral data.

Author Contributions: D.B. designed the algorithm; C.G. and D.M. conceived and designed the experiments; D.B. performed the experiments; D.B., C.G. and D.M. analyzed the data; D.B., C.G. and D.M. wrote the paper.

Funding: This work has been supported by the by the Spanish MINECO projects TIN2013-40968-P and TIN2017-87237-P.

Acknowledgments: The authors would like to thank the anonymous reviewers. Their comments and suggestions greatly improved this paper.

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

References

1. Dozier, J. Spectral signature of alpine snow cover from the landsat thematic mapper. *Remote Sens. Environ.* **1989**, *28*, 9–22. [CrossRef]
2. Haboudane, D.; Miller, J.R.; Pattey, E.; Zarco-Tejada, P.J.; Strachan, I.B. Hyperspectral vegetation indices and novel algorithms for predicting green LAI of crop canopies: Modeling and validation in the context of precision agriculture. *Remote Sens. Environ.* **2004**, *90*, 337–352. [CrossRef]
3. Akbari, H.; Halig, L.V.; Schuster, D.M.; Osunkoya, A.; Master, V.; Nieh, P.T.; Chen, G.Z.; Fei, B. Hyperspectral imaging and quantitative analysis for prostate cancer detection. *J. Biomed. Opt.* **2012**, *17*, 760051–7600510. [CrossRef] [PubMed]
4. JPL. Available online: aviris-ng.jpl.nasa.gov/ (accessed on 29 January 2018).
5. Bascones, D.; Gonzalez, C.; Mozos, D. FPGA Implementation of the CCSDS 1.2.3 Standard for Real-Time Hyperspectral Lossless Compression. *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.* **2017**, *11*, 1158–1165. [CrossRef]
6. Báscones, D.; González, C.; Mozos, D. Parallel Implementation of the CCSDS 1.2.3 Standard for Hyperspectral Lossless Compression. *Remote Sens.* **2017**, *9*, 973. [CrossRef]
7. Motta, G.; Rizzo, F.; Storer, J.A. *Hyperspectral Data Compression*; Springer Science & Business Media: New York, NY, USA, 2006; p. 417.
8. Rucker, J.T.; Fowler, J.E.; Younan, N.H. JPEG2000 coding strategies for hyperspectral data. In Proceedings of the 2005 IEEE International Geoscience and Remote Sensing Symposium (IGARSS '05), Seoul, Korea, 29 July 2005; Volume 1, pp. 128–131. [CrossRef]
9. Penna, B.; Tillo, T.; Magli, E.; Olmo, G. Progressive 3-D coding of hyperspectral images based on JPEG 2000. *IEEE Geosci. Remote Sens. Lett.* **2006**, *3*, 125–129. [CrossRef]

10. Christophe, E.; Mailhes, C.; Duhamel, P. Hyperspectral image compression: adapting SPIHT and EZW to anisotropic 3-D wavelet coding. *IEEE Trans. Image Process.* **2008**, *17*, 2334–2346. [CrossRef] [PubMed]
11. Jifara, W.; Jiang, F.; Zhang, B.; Wang, H.; Li, J.; Grigorev, A.; Liu, S. Hyperspectral image compression based on online learning spectral features dictionary. *Multimed. Tools Appl.* **2017**, *76*, 25003–25014. [CrossRef]
12. Du, Q.; Fowler, J.E. Hyperspectral image compression using JPEG2000 and principal component analysis. *IEEE Geosci. Remote Sens. Lett.* **2007**, *4*, 201–205. [CrossRef]
13. Kambhatla, N.; Leen, T.K. Fast non-linear dimension reduction. *Adv. Neural Inf. Process. Syst.* **1994**, 152–159. [CrossRef]
14. Masiello, G.; Serio, C.; Antonelli, P. Inversion for atmospheric thermodynamical parameters of IASI data in the principal components space. *Q. J. R. Meteorol. Soc.* **2012**, *138*, 103–117. [CrossRef]
15. Taubman, D.; Marcellin, M. *JPEG2000 Image Compression Fundamentals, Standards and Practice*; Springer Science & Business Media: New York, NY, USA, 2012; Volume 642, p. 773. [CrossRef]
16. Voronoi, G. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les parallélloèdres primitifs. *J. Reine. Angew. Math.* **1908**, *1908*, 198–287. [CrossRef]
17. Li, H. Available online: github.com/haifengl/smile (accessed on 9 January 2018).
18. International Telecommunication Union (ITU). *JPEG2000 Core Part 1-T.800 (08/2002)*; ITU: Geneva, Switzerland, 2002; Volume 800.
19. Báscones, D. Available online: github.com/Daniel-BG/Jypec (accessed on 17 January 2018).
20. Raff, E. JSAT: Java Statistical Analysis Tool, a Library for Machine Learning. *J. Mach. Learn. Res.* **2017**, *18*, 1–5.
21. Abeles, P. Available online: ejml.org (accessed on 18 December 2017).
22. Sweldens, W. The Lifting Scheme: A Construction of Second Generation Wavelets. *SIAM J. Math. Anal.* **1998**, *29*, 511–546. [CrossRef]
23. Spectir. Available online: www.spectir.com/free-data-samples/ (accessed on 18 January 2018).