PERFORMANCE IMPACT OF PARAMETER TUNING ON THE CCSDS-123 LOSSLESS MULTI- AND HYPERSPECTRAL IMAGE COMPRESSION STANDARD

Estanislau Augé[a], Jose Enrique Sánchez[a], Aaron Kiely[b], Ian Blanes[a], and Joan Serra-Sagristà[a]

[a]*Dept. of Information and Communications Engineering*
Universitat Autònoma de Barcelona
Edifici Q, UAB, E-08193 Cerdanyola del Vallès (Barcelona), Spain
Email: {Estanislau.Auge,JoseEnrique.Sanchez}@e-campus.uab.cat, {ian.blanes,joan.serra}@uab.cat
[b]*NASA Jet Propulsion Laboratory (JPL)*
California Institute of Technology, Pasadena, CA, USA
Email: Aaron.B.Kiely@jpl.nasa.gov

Multispectral and hyperspectral image data payloads have large size and may be challenging to download from remote sensors. To alleviate this problem, such images can be effectively compressed using specially designed algorithms.

The new CCSDS-123 standard has been developed to address onboard lossless coding of multispectral and hyperspectral images. The Standard is based on the Fast Lossless (FL) algorithm, which is composed of a causal context-based prediction stage and an entropy-coding stage that utilizes Golomb power-of-two codes. Several parts of each of these two stages have adjustable parameters.

CCSDS-123 provides satisfactory performance for a wide set of imagery acquired by various sensors, but end-users of a CCSDS-123 implementation may require assistance to select a suitable combination of parameters for a specific application scenario. To assist end-users, this paper investigates the performance of CCSDS-123 under different parameter combinations and addresses the selection of an adequate combination given a specific sensor.

Experimental results suggest that prediction parameters have a greater impact on the compression performance than entropy-coding parameters.

Keywords: *remote sensing, lossless image coding, predictive coding, multi- and hyperspectral imagery, CCSDS 123.0-B-1, configuration parameters.*

1 Introduction

Remote-sensing instruments collect huge amounts of data to be transmitted to ground stations. Due to the constraints of the downlink channel and to minimize the transmission time, it is beneficial to compress such data onboard prior to transmission. However, these instruments are usually limited in memory and computing capacity, and, as a consequence, only low complexity coding algorithms are appropriate.

The Consultative Committee for Space Data Systems (CCSDS) [1] is a multinational forum for the development of communications and data systems standards for spaceflight. It is composed of the world's major space agencies, such as NASA, ESA and CNES, among others. The Multispectral and Hyperspectral Data Compression (MHDC) working group of the CCSDS has recently developed the new CCSDS-123 standard [2], which is based on the Fast Lossless (FL) compression algorithm [3] and intended for onboard lossless coding of multi- and hyperspectral imagery. The standard achieves state-of-the-art compression performance on images collected by a wide variety of sensors. It has low computational complexity, which facilitates implementation in resource-constrained scenarios.

CCSDS-123 includes several user-specified configuration parameters, and it may not always be easy for users to determine suitable values for these parameters. This article illustrates the impact of different parameters values on the compression performance and the efficiency of the standard on a large corpus of images. Experiments and results provided may offer instructive hints to end-users.

The remainder of this paper is structured as follows: Section 2 provides an overview of the compressor and the images used in the experiments, Section 3 reports the experimental results, and Section 4 presents some conclusions.

2 Method and materials

2.1 The CCSDS-123 standard

Fig. 1 illustrates the two stages of the CCSDS-123 compressor: a predictor and an encoder. The encoder may use either a *sample-adaptive* or *block-adaptive* entropy coder. The latter option, which is defined in a previous CCSDS standard [4], is provided to enable the reutilization of existing implementations, typically at the cost of a slight decrease in performance. The block-adaptive entropy coder has been intentionally left outside the scope of this article to limit the number of experimental results that would have been needed to analyze all cases.

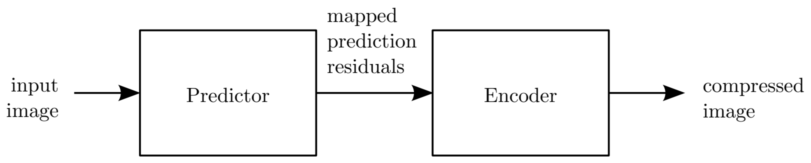


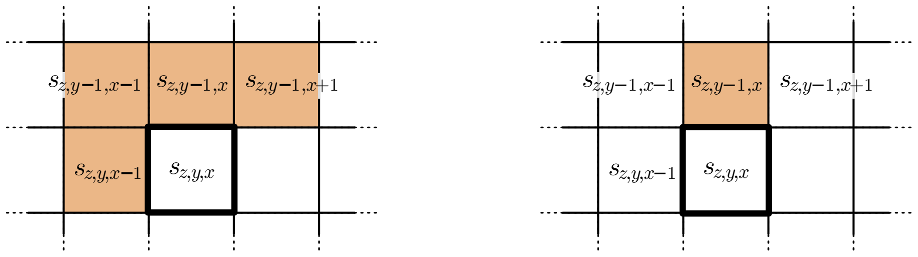
Figure 1: Overview of a CCSDS-123 implementation.

To facilitate an understanding of the experiments and results, we now briefly describe the CCSDS-123 algorithm, emphasizing the different parameters that can be adjusted. To improve readability, parameters discussed in this section are highlighted using bold typography.

Predictor

The predictor's task is to calculate a prediction $\hat{s}_{z,y,x}$ (at column x, row y, band z) for the sample $s_{z,y,x}$ — the one to be predicted — and map the difference between these two quantities to a non-negative integer *mapped prediction residual* value $\delta_{z,y,x}$, which is losslessly encoded in the encoding stage. Prediction depends on previously encoded sample values in the current and P previous (i.e., lower-indexed) bands. The **number of prediction bands**, P, used by the predictor can take values in the range 0 to 15.

The prediction of $s_{z,y,x}$ is computed as an offset relative to the scaled value of a *local sum* $\sigma_{z,y,x}$ calculated from samples in the neighborhood of $s_{z,y,x}$. One can think of the local sum as a scaled preliminary estimate of the value of $s_{z,y,x}$. The user's choice of **local sum type** — either *neighbor-oriented* or *column-oriented* — determines how the local sum $\sigma_{z,y,x}$ is calculated. Fig. 2 illustrates the neighborhood used in each case. The neighbor-oriented local sum (Fig. 2a) is equal to the sum of $s_{z,y-1,x-1}$, $s_{z,y-1,x}$, $s_{z,y-1,x+1}$ and $s_{z,y,x-1}$, while the column-oriented local sum (Fig. 2b) is equal to four times the vertically adjacent sample $s_{z,y-1,x}$.



(a) Neighbor-oriented local sum (b) Column-oriented local sum

Figure 2: Samples used to calculate the local sum $\sigma_{z,y,x}$.

Differences between local sum values and scaled values of previously encoded samples are arranged in a *local difference vector* $\mathbf{U}_{z,y,x}$ whose definition depends on the user's choice of **prediction mode** — either *full* or *reduced* prediction mode may be used. Under full prediction mode, the local difference vector is defined as

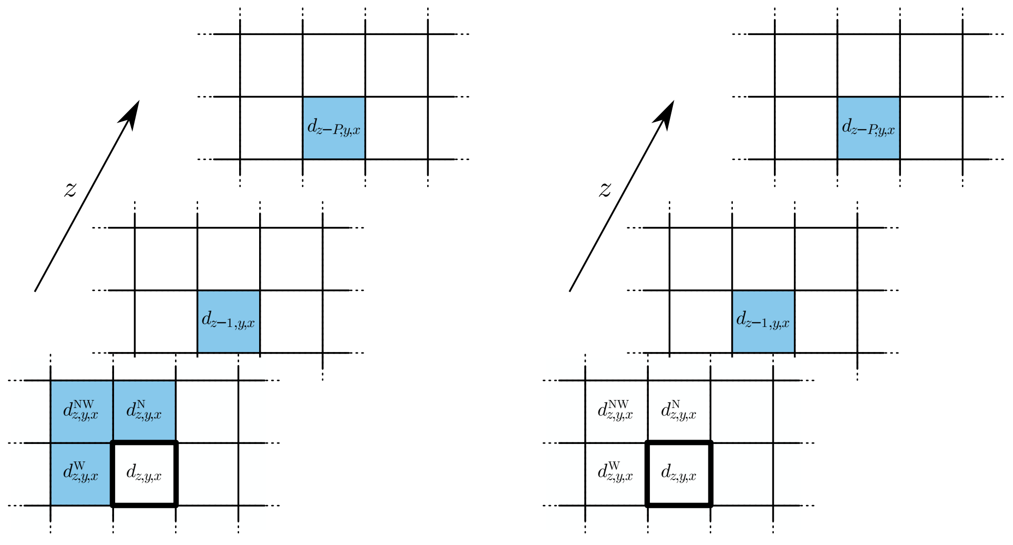
$$\mathbf{U}_{z,y,x} = \begin{bmatrix} d^{\mathrm{N}}_{z,y,x} \\ d^{\mathrm{W}}_{z,y,x} \\ d^{\mathrm{NW}}_{z,y,x} \\ d_{z-1,y,x} \\ d_{z-2,y,x} \\ \vdots \\ d_{z-P,y,x} \end{bmatrix} = \begin{bmatrix} 4s_{z,y-1,x} - \sigma_{z,y,x} \\ 4s_{z,y,x-1} - \sigma_{z,y,x} \\ 4s_{z,y-1,x-1} - \sigma_{z,y,x} \\ 4s_{z-1,y,x} - \sigma_{z-1,y,x} \\ 4s_{z-2,y,x} - \sigma_{z-2,y,x} \\ \vdots \\ 4s_{z-P,y,x} - \sigma_{z-P,y,x} \end{bmatrix}.$$

Under reduced prediction mode, the definition of $\mathbf{U}_{z,y,x}$ omits the first three components but is otherwise the same.

The *directional local differences*, $d^{\mathrm{N}}_{z,y,x}$, $d^{\mathrm{W}}_{z,y,x}$ and $d^{\mathrm{NW}}_{z,y,x}$, used under full prediction mode, represent directional offsets between the local sum for the to-be-predicted sample and the scaled values of adjacent samples in the same band; the labels "N", "W", and "NW" are intended to suggest compass directions. The *central local differences*, $d_{z-1,y,x}, d_{z-2,y,x}, \ldots, d_{z-P,y,x}$ are differences between the local sum and scaled sample values in previous bands. Fig. 3 depicts the local differences used in prediction under both the full (Fig. 3a) and reduced (Fig. 3b) prediction modes.

The prediction is calculated by means of the inner product between the local difference vector $\mathbf{U}_{z,y,x}$ and the *weight vector* $\mathbf{W}_{z,y,x}$. Specifically, the inner product $\hat{d}_{z,y,x} = \mathbf{W}^{\mathrm{T}}_{z,y,x} \cdot \mathbf{U}_{z,y,x}$ is a scaled prediction of the value of $d_{z,y,x}$. Loosely speaking, the weight vector serves as a measure of the effectiveness of each component of the local difference vector in predicting the sample being coded. The value of the **weight component resolution** parameter, Ω, determines the resolution of weight vector components; each component is a signed quantity that can be represented using $\Omega + 3$ bits.

The initial values of weight vector components are determined by the choice of **weight initialization method**, either *default* or *custom*. A standard set of initial weight components is used under the default initialization method. Under the custom initialization option, each initial component may be explicitly specified using a **weight initialization table** with precision determined by the value of the **weight initialization resolution** parameter, Q.



(a) Full prediction mode (b) Reduced prediction mode

Figure 3: Local differences used under the different prediction modes. Only differences depicted in blue are used in each case.

The **register size** parameter, R, identifies the size, in bits, of the register used to perform the main prediction calculation, which is to compute the *scaled predicted sample value*, $\tilde{s}_{z,y,x}$, from $\hat{d}_{z,y,x}$. Using a smaller register increases the chance that an overflow will occur, leading to a poor prediction for the affected sample, but the resulting (poorly predicted) sample is still losslessly encoded. Knowledge of the register size allows the decompressor to duplicate the prediction calculation and accurately reconstruct each sample value.

Following prediction of a sample, the *scaled prediction error* is then computed as $e_{z,y,x} = 2s_{z,y,x} - \tilde{s}_{z,y,x}$, and the components of the weight vector are adaptively updated based on the sign of $e_{z,y,x}$. The rate at which the weight vector adapts to the input image is proportional to $2^{-\rho_{y,x}}$. The *weight update scaling exponent* $\rho_{y,x}$ changes over time following a schedule determined by three user-controlled parameters: the **weight update scaling exponent initial parameter**, $v_{\min}$, the **weight update scaling exponent final parameter**, $v_{\max}$, and the **weight update scaling exponent change interval**, t_{inc}. A smaller value of $\rho_{y,x}$ produces larger weight increments, yielding faster adaption to source statistics but worse steady-state prediction. The value of the interval t_{inc} controls how quickly the predictor proceeds from a smaller to a larger value of the weight update scaling exponent.

The *predicted sample value*, $\hat{s}_{z,y,x} = \lfloor \frac{\tilde{s}_{z,y,x}}{2} \rfloor$ is used to compute the *prediction residual* $\Delta_{z,y,x} = s_{z,y,x} - \hat{s}_{z,y,x}$. The mapped prediction residual $\delta_{z,y,x}$ is computed from $\Delta_{z,y,x}$, $\hat{s}_{z,y,x}$, and the least significant bit of $\tilde{s}_{z,y,x}$. This $\delta_{z,y,x}$ is the final output of the predictor, from which the original sample value can be recovered by the decompressor.

Encoder

The entropy coder losslessly encodes the mapped prediction residual values produced by the predictor. The **sample encoding order** selected by the user determines the order in which samples are encoded in the compressed bitstream. This order can affect the compressed image size when the block-adaptive entropy coder is used, but not when the sample-adaptive entropy coder is used, and thus we make no further analysis of this option. However, note that the sample encoding order may affect memory requirements for an implementation, e.g., additional buffering might be necessary to provide an encoding order that does not match the order in which samples are delivered by an imaging instrument.

The overall sample-adaptive entropy coding procedure is very similar to that used by the JPEG-LS image compressor [5]. The encoder maintains two internal variables, an *accumulator* $\Sigma_{z,y,x}$ and a *counter* $\Gamma_{y,x}$. The ratio $\Sigma_{z,y,x}/\Gamma_{y,x}$ is an estimate of the mean value of $\delta_{z,y,x}$, and this ratio determines the particular Golomb-power-of-two (GPO2) code selected to encode $\delta_{z,y,x}$. After encoding $\delta_{z,y,x}$, the counter is incremented by 1 and the accumulator is incremented by the value of $\delta_{z,y,x}$. The counter and accumulator are both halved whenever the counter reaches a value of $2^{\gamma^*}-1$, where γ^* is the **rescaling counter size** parameter. A smaller value of γ^* increases the relative influence of more recently encoded samples in selecting the GPO2 code.

The counter's initial value in each band is 2^{γ_0}, where γ_0 is the **initial count exponent** parameter. The initial accumulator value in each band is determined by the user-specified sequence $\left\{k'_0, k'_1, \ldots, k'_{N_Z-1}\right\}$, referred to as the **accumulator initialization table**. The accumulator initialization in each band z depends on γ_0 and k'_z and is defined so that the initial GPO2 code parameter in the band is equal to k'_z. When all accumulator initialization table elements have the same value, denoted by K, this value is referred to as the **accumulator initialization constant**.

The maximum encoded length of any sample is controlled via the **unary length limit** parameter $U_{\max}$. If the GPO2 code parameter $k_{z,y,x}$ selected to encode $\delta_{z,y,x}$ would result in a GPO2 codeword length exceeding $U_{\max}+k_{z,y,x}$ bits, then $\delta_{z,y,x}$ is instead encoded as a sequence of $U_{\max}$ zeros followed by the D-bit unsigned binary representation of $\delta_{z,y,x}$, where D is the bit depth of the input image. Thus, the maximum number length of any encoded sample is $U_{\max}+D$ bits. Limiting the codeword length in this way may simplify an implementation and also helps to limit the cost of encoding occasional outlier samples for which prediction performs poorly, e.g., due to register overflow.

Summary of customizable parameters

Table 1 summarizes the compression parameters tested and the default settings used in the experiments.

2.2 Corpus of images

Our experiments have been performed on the image corpus defined by the MHDC working group for the purposes of algorithm evaluation and testing. It contains 101 images collected from 14 different sensors. Table 2, an excerpt from [6], includes a summary of the characteristics of each sensor. The corpus includes a variety of image types, from ultraspectral sounder images captured by IASI and AIRS sensors, through hyperspectral images captured by CASI, SFSI, AVIRIS and Hyperion sensors, to multispectral images captured by MODIS, Landsat, Vegetation, MSG, Pleiades and SPOT5 sensors. The table also indicates whether images have been calibrated or are raw images.

Table 1: Description of the parameters evaluated in the experimental section. Default settings are set based on the authors' experience.

Predictor

Parameter Name	*Symbol*	*Default Value*	*Description*
Number of Prediction Bands	P	3	Number of previous bands used to perform prediction
Register Size	R	64	Size of register used in prediction calculation
Local Sum Type		*column-oriented* for CRISM, Hyperion, M3, MODIS day and night; *neighbor-oriented* otherwise	Identifies neighborhood used to calculate local sums
Prediction Mode		*reduced* for AIRS, AVIRIS 12-bit, CASI, CRISM, Hyperion, IASI, M3, MODIS day and night; *full* otherwise	Indicates whether directional local differences are used in the prediction calculation
Weight Component Resolution	Ω	19	Determines number of bits used to represent each weight vector component
Weight Initialization Method		*default*	Determines initial values of weight vector components
Weight Initialization Table	$\{\Lambda_z\}$	(unused)	Defines the initial weight components under *custom* initialization.
Weight Initialization Resolution	Q	(unused)	Determines the precision of the initial weight components under *custom* initialization.
Weight Update Scaling Exponent Initial Parameter	$v_{\min}$	-1	Determines initial rate at which predictor adapts weight vector to input
Weight Update Scaling Exponent Final Parameter	$v_{\max}$	3	Determines final rate at which predictor adapts weight vector to input
Weight Update Scaling Exponent Change Interval	t_{inc}	2^6	Determines the interval between increments to the weight update scaling exponent

Entropy Coder

Parameter Name	*Symbol*	*Default Value*	*Description*
Unary Length Limit	$U_{\max}$	18	Limits the maximum length of any encoded sample
Rescaling Counter Size	γ^*	6	Determines the interval between rescaling of counter and accumulator
Initial Count Exponent	γ_0	1	Sets initial counter value
Accumulator Initialization Table	$\{k_z'\}$	(unused)	Sets an initial accumulator value for each band
Accumulator Initialization Constant	K	3	Sets initial accumulator value in all bands

Table 2: Corpus Information. Characteristics of several sensors are provided.

Instrument Type	*Image Type*	*Bit Depth*	*Number of Bands*	*Width*	*Height*
AIRS	raw	12-14	1501	90	135
AVIRIS	raw	12	224	{614, 680}	512
AVIRIS	raw	16	224	680	512
CASI	raw	12	72	405	{2852, 1225}
CRISM	raw	12	545	{320, 640}	{420, 450, 480, 510}
CRISM	raw	12	74	64	2700
Hyperion	raw	12	242	256	{1024, 3187, 3176, 3242}
IASI	calibrated	12	8461	66	60
Landsat	raw	8	6	1024	1024
M3	raw global	12	86	320	512
M3	raw target	12	260	640	512
MODIS	night, raw	12	17	1354	2030
MODIS	day, raw	12	14	1354	2030
MODIS	500m, raw	12	5	2708	4060
MODIS	250m, raw	12	2	5416	8120
MSG	calibrated	10	11	3712	3712
Pleiades	HR, Simulated	12	4	224	{2456, 3928, 2448}
SFSI	raw	12	240	496	140
SPOT5	HRG, processed	8	3	1024	1024
Vegetation	raw	10	4	1728	{10080, 10193}

3 Experimental results

This section presents results from experiments conducted to assess the influence of different parameters on compression performance. All results have been obtained using Emporda [7], a free software implementation of CCSDS-123. Emporda has been cross-validated against three other independently developed implementations of the standard.

To avoid the combinatorial explosion that would arise from joint evaluation of all possible values of all parameters, default values are set for each parameter based on the authors' experience. Thus, when the effect of a given set of interrelated parameters is assessed, the default values reported in Table 1 are used for the remaining parameters.

While experimental results have been produced for the complete image corpus, due to space constraints, results are shown only for a few representative images, each from a different sensor. The same images are shown for all experiments, with exceptions whenever showing different images yields a more heterogeneous view of the results.

3.1 Predictor parameters

This section presents results from four experiments exercising predictor parameters.

The first experiment tests the impact of changing the number of previous bands P used in prediction. Fig. 4 shows the compressed data rate for different sensors as a function of P. As we might expect, setting $P = 0$ (i.e., not using any previous bands for prediction) yields the worst results, often by a dramatic amount. However, we generally observe diminishing marginal returns as we increase P; the bit rate curve tends to flatten, or in the case of the AVIRIS and CASI images, compressed data rate actually increases beyond $P > 2$ possibly due to the occurrence of overfitting. For the images in the corpus, evidently there is not much motivation to use values of P near the maximum allowed, 15.

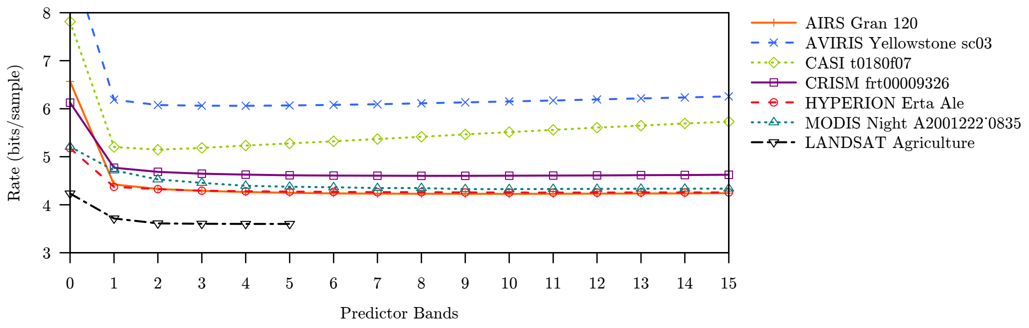
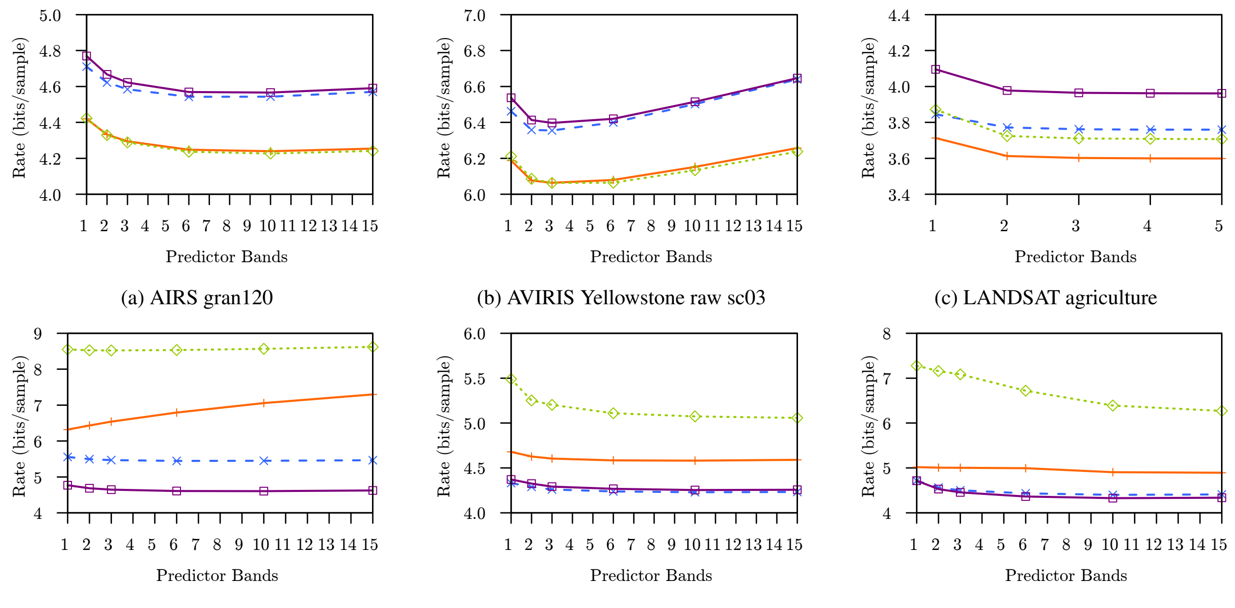


Figure 4: Compressed bit rate performance as a function of P.

We next consider the choice of prediction mode and local sum type. The reason for defining two different prediction modes (full and reduced) as well as two different local sum types (column- and neighbor-oriented) in the CCSDS-123 standard is so that the same prediction framework can be used to provide effective prediction for image data from different types of imagers. Specifically, the use of column-oriented local sums and reduced mode is intended to provide more effective compression for raw images from pushbroom imagers that produce streaking artifacts parallel to the alongtrack (y) direction[1] [6]. For images without such artifacts, such as calibrated images or imagery from whisk-broom instruments, we generally expect improved performance by using neighbor-oriented local sums.

Fig. 5 shows compressed data rate as a function of P for six different images under all four combinations of prediction mode and local sum type. The relative performance of the four combinations generally follows what one would expect based on the presence or absence of streaking artifacts in the input image. On the CRISM, Hyperion, and MODIS night images, which exhibit streaking artifacts ranging from moderate to severe, column-oriented local sums combined with reduced mode gives the best performance. On the AIRS, AVIRIS, and LANDSAT images, which do not exhibit streaking artifacts, neighbor-oriented local sums outperform column-oriented local sums and the choice of full or reduced mode has a relatively small impact on performance.



(a) AIRS gran120 (b) AVIRIS Yellowstone raw sc03 (c) LANDSAT agriculture

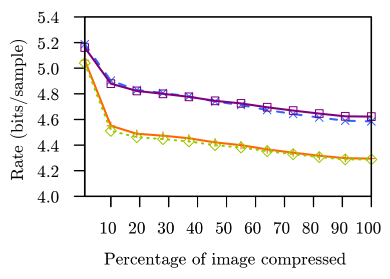
(d) CRISM frt00009326_07_sc167 (e) Hyperion ErtaAle (f) MODIS night A2001222_0835

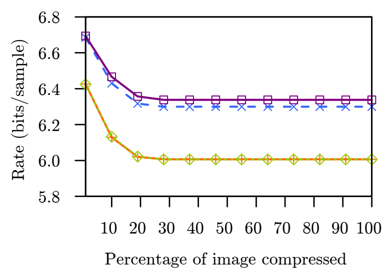
Figure 5: Compressed bit rate for different choices of prediction mode and local sum type. Each curve represents a different combination using the following key: —+— *full+neighbor*, - × - *full+column*, ···◇··· *reduced+neighbor*, or —□— *reduced+column*.

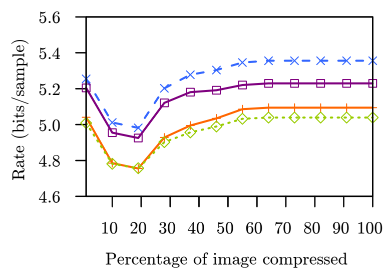
For a fixed value of P, the use of full prediction mode requires three additional components in the weight vector compared to reduced mode. Thus, we would expect slower adaptation of this larger weight vector, and so for a given image it could be the case that full prediction mode provides a benefit over reduced mode, but only after processing a sufficient number of samples. It appears that this effect is typically not significant. Fig. 6 shows the cumulative running average compressed data rate as compression proceeds through an image. With the exception of Fig. 6f, the relative performance of the different choices of prediction mode and local sum type does not change substantially during compression.

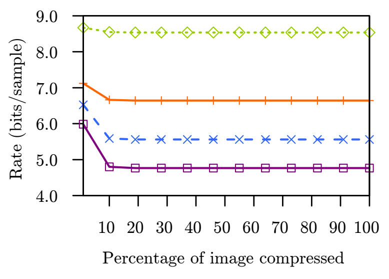
The second experiment addresses the influence of parameters $v_{\min}$, $v_{\max}$ and t_{inc}, which affect the rate at which the predictor adapts to the image. The results are presented in Fig. 7, which shows the compressed data rate as a function of $v_{\max}$ for different values of t_{inc} and $v_{\min}$. In general, t_{inc} and $v_{\min}$ have little impact on compression performance, and $v_{\max}$ is the

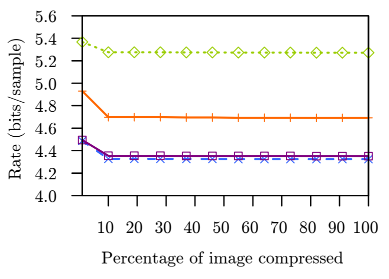
[1]MODIS is in fact a "whisk-push" imager that exhibits streaking artifacts parallel to the cross-track direction. Thus, following the recommendation in the first note on [2, p. 3-1], in our experiments we transpose each band of MODIS images prior to compression.


(a) AIRS gran120


(b) AVIRIS Yellowstone raw sc03


(c) CASI t0477f06


(d) CRISM frt00009326_07_sc167


(e) Hyperion ErtaAle

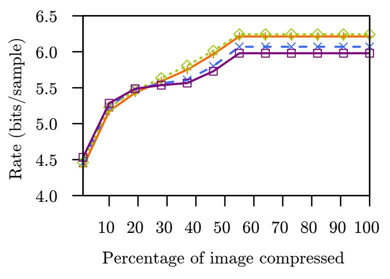

(f) MODIS day A2001222_0835

Figure 6: Cumulative running average bit rate. Each curve represents a different combination of prediction mode and local sum type using the following key: —+— *full+neighbor*, - × - *full+column*, ···◇··· *reduced+neighbor*, or —□— *reduced+column*.

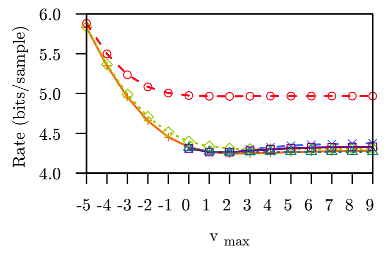
most important parameter in this experiment. As $v_{\max}$ increases, compression performance improves up to a point, then worsens, but to a much smaller degree. For the images shown here, the values of t_{inc} and $v_{\min}$ have little impact except for the AIRS image. Given the small spatial size of this image (90×135), the combination of a large value of t_{inc} and a large difference $v_{\max} - v_{\min}$ ensures that the weight update scaling exponent does not reach its final value until near the end of the image, and thus for the case of $t_{\text{inc}} = 2^{10}$ and $v_{\min} = -6$ the data rate is significantly higher.

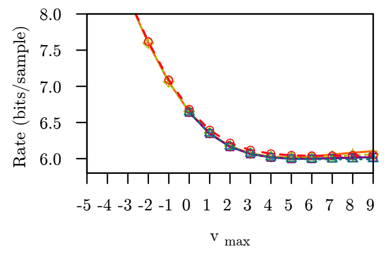
As further evidence of the relatively small impact of t_{inc} and $v_{\min}$, Fig. 8 shows that the compressed data rate as a function of $v_{\min}$ is remarkably flat when the value of $v_{\max}$ chosen for each image is the one that yields the best compression performance for the last 40% of the image in the y dimension.

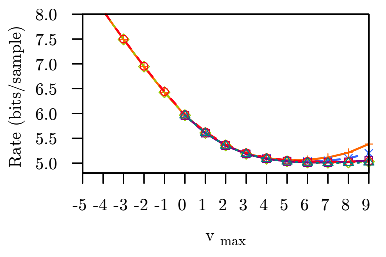
The third experiment addressing the impact of predictor parameters on compression effectiveness considers the influence of Ω and R on compression performance. The results of Fig. 9 suggest that Ω has a noticeable impact on the compression performance, while R does not. This suggests that, for these images, there are only an insignificant number of predictions that require a large value of R to prevent overflow. As for Ω, it can be seen that results for $\Omega = 8$ are very close to those of $\Omega = 12$ and $\Omega = 16$, suggesting that using the largest values of Ω is not necessary to achieve a good performance.

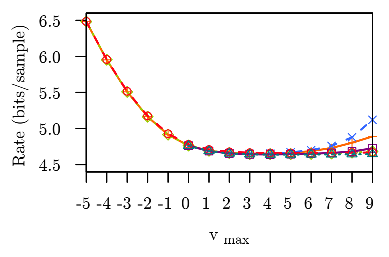
The choice of compression parameters affects not only compression performance but also complexity. This warrants some consideration in a fourth experiment; we have already seen that increasing complexity (e.g., using very large values of P) does not always improve compression performance.

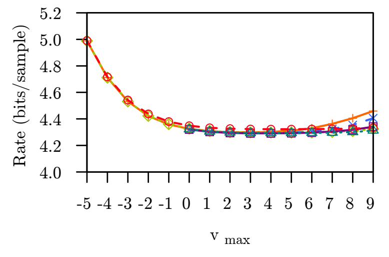
While a thorough analysis of complexity is well beyond the scope of this paper, we note some basic relationships between arithmetic operations required in the prediction calculation and the choice of P, prediction mode, and local sum type parameters. Calculation of the local sum includes additional terms when neighbor-oriented local sums are used. The


(a) AIRS gran120


(b) AVIRIS Yellowstone raw sc03


(c) CASI t0477f06


(d) CRISM frt00009326_07_sc167


(e) Hyperion Cuprite

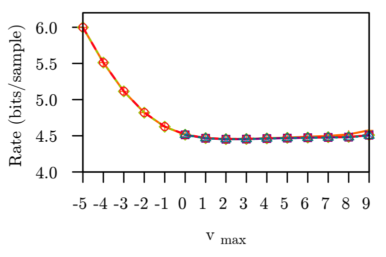
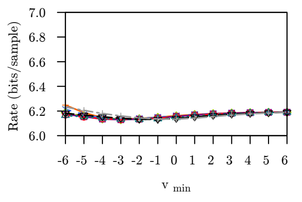
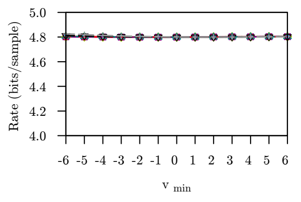
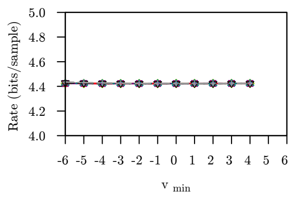

(f) MODIS night A2001222_0835

Figure 7: Bit rate for different choices of parameters that affect the adaptability of the predictor to the image. Each curve represents a different combination of t_{inc} and v_{min} using the following key: —+— $\{t_{\text{inc}} = 2^4, v_{\text{min}} = -6\}$, – × – $\{t_{\text{inc}} = 2^4, v_{\text{min}} = -1\}$, ···◇··· $\{t_{\text{inc}} = 2^7, v_{\text{min}} = -6\}$, —□— $\{t_{\text{inc}} = 2^7, v_{\text{min}} = -1\}$, – ○ – $\{t_{\text{inc}} = 2^{10}, v_{\text{min}} = -6\}$, and ···△··· $\{t_{\text{inc}} = 2^{10}, v_{\text{min}} = -1\}$.


(a) AVIRIS Yellowstone raw sc00


(b) CASI t0180f07


(c) Hyperion LakeMonona

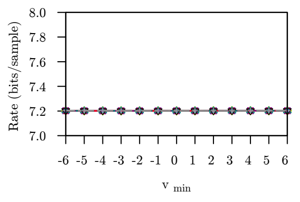

(d) MODIS 500m

Figure 8: Bit rate depending on v_{min} and t_{inc} for a fixed v_{max}. Values of t_{inc} are represented using the following key: —+— 2^4, – × – 2^5, ···◇··· 2^6, —□— 2^7, – ○ – 2^8, ···△··· 2^9, —▽— 2^{10}, and —+— 2^{11}.

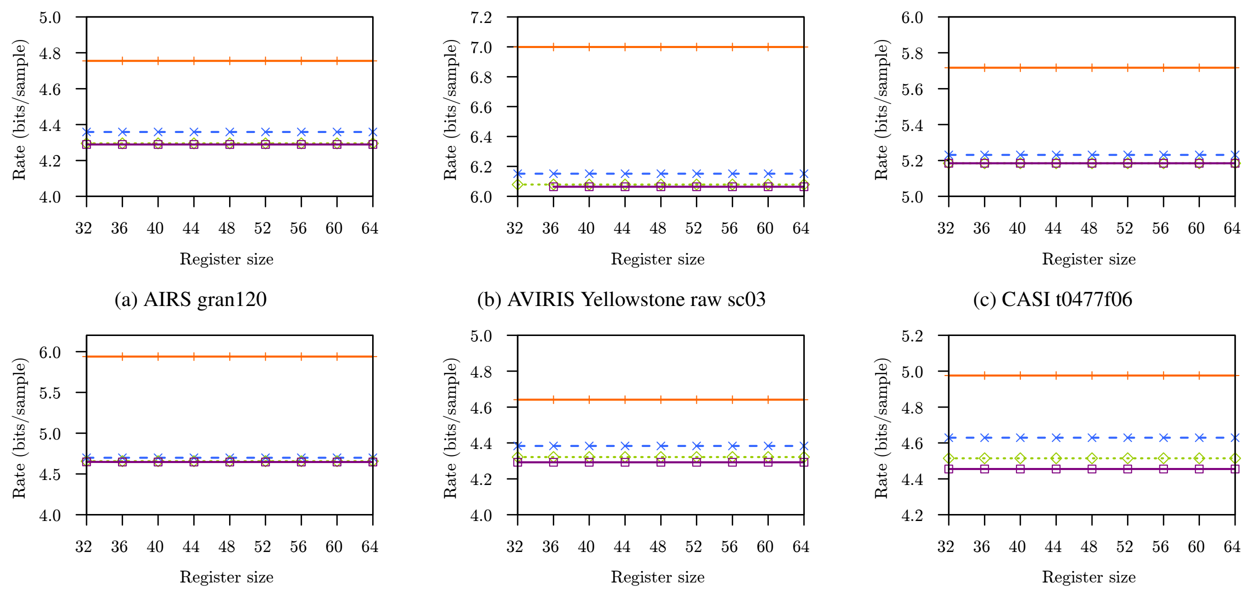
number of components in the weight vector and local difference vector is P under reduced mode and $P+3$ under full mode. As these vectors grow longer, the prediction operation requires additional calculations. Thus, prediction complexity is lowest under reduced prediction mode using column-oriented local sums and $P = 0$.

As a rough complexity measure, Fig. 10 shows the compression software execution time relative to the time required under the least complex choice (reduced prediction mode using column-oriented local sums and $P = 0$). As expected, neighbor-oriented local sums and full prediction mode are each slower than their alternatives, and compression time is approximately linear in P.

3.2 Entropy coder parameters

This section presents results from three experiments exercising encoder parameters.

The first one relates to the accumulator, its initial value, which depends on γ_0, and the rescaling interval, which depends



(a) AIRS gran120 (b) AVIRIS Yellowstone raw sc03 (c) CASI t0477f06

(d) CRISM frt00009326_07_sc167 (e) Hyperion ErtaAle (f) MODIS night A2001222_0835

Figure 9: Bit rate for different values of R and Ω. Each curve represents a different value of Ω using the following key: —+— 4, - ✕ - 8, ···◇··· 12, and —□— 16.

on γ^*. Fig. 11 suggests that the impact of these parameters on compressed data rate performance is absolutely marginal.

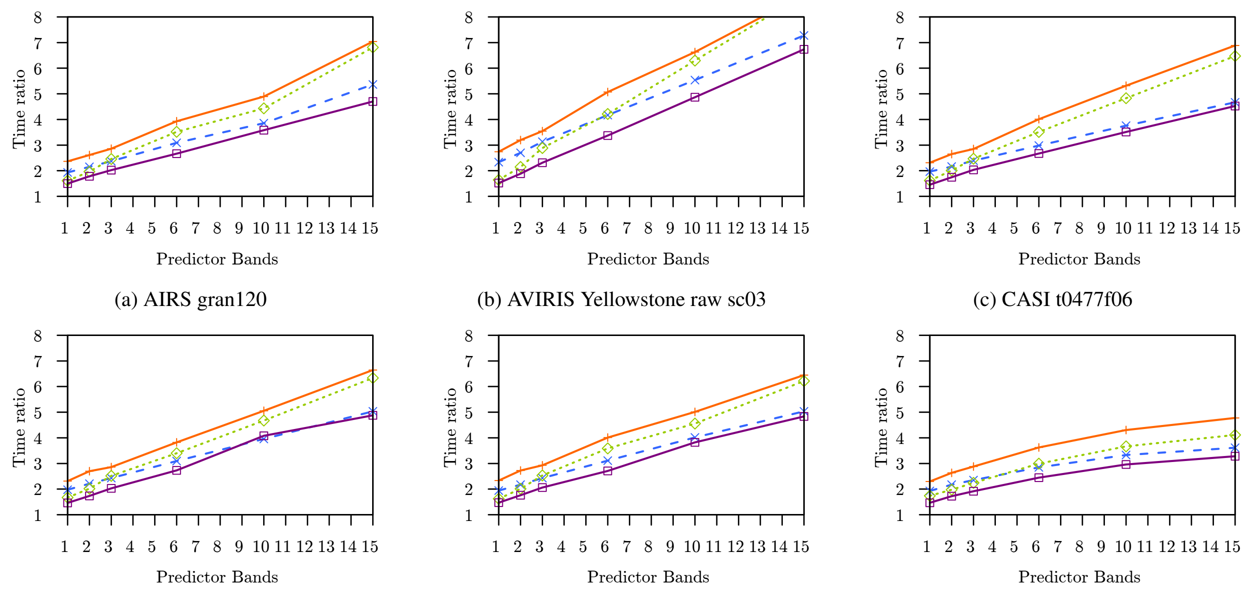
The second experiment, whose results are shown in Fig. 12, considers the influence of parameter K. It can be noticed that, again, the compression ratio does not seem to depend much on K. A single exception would be the AIRS image, but the differences are not significant.

The final experiment regarding the encoder considers the impact of parameter $U_{\max}$ in the coding performance. Fig. 13 reports the results only for an AIRS image, because the results for the other images are also constant. A much finer scale for the bit rate is employed in Fig. 13b, showing the very small differences on the bit rate produced by the parameter $U_{\max}$.

3.3 Image segmentation

Data transmitted over space communications channels are vulnerable to corruption in the form of bit errors and/or data loss. While such events may occur with low probability, even a single bit error in a compressed image generally results in corruption of reconstructed samples extending to the end of the image.

To mitigate the impact of data corruption on the communications channel, one can partition a large image into smaller images that can be independently decompressed. We refer to each of these smaller images as a *segment* of the larger image (Fig. 14). We assume that communications protocols employed by the spacecraft incorporate mechanisms (e.g., a packetization structure) that allow the beginning of the next segment to be identified following a data corruption event, and thus, the impact of data corruption is limited to the affected segment. We note that the CCSDS-123 standard does not directly address image segmentation (the term "segment" is not part of the standard); in the view of the standard, each such segment is simply a separate image.



(a) AIRS gran120 (b) AVIRIS Yellowstone raw sc03 (c) CASI t0477f06

(d) CRISM frt00009326_07_sc167 (e) Hyperion ErtaAle (f) MODIS night A2001222_0835

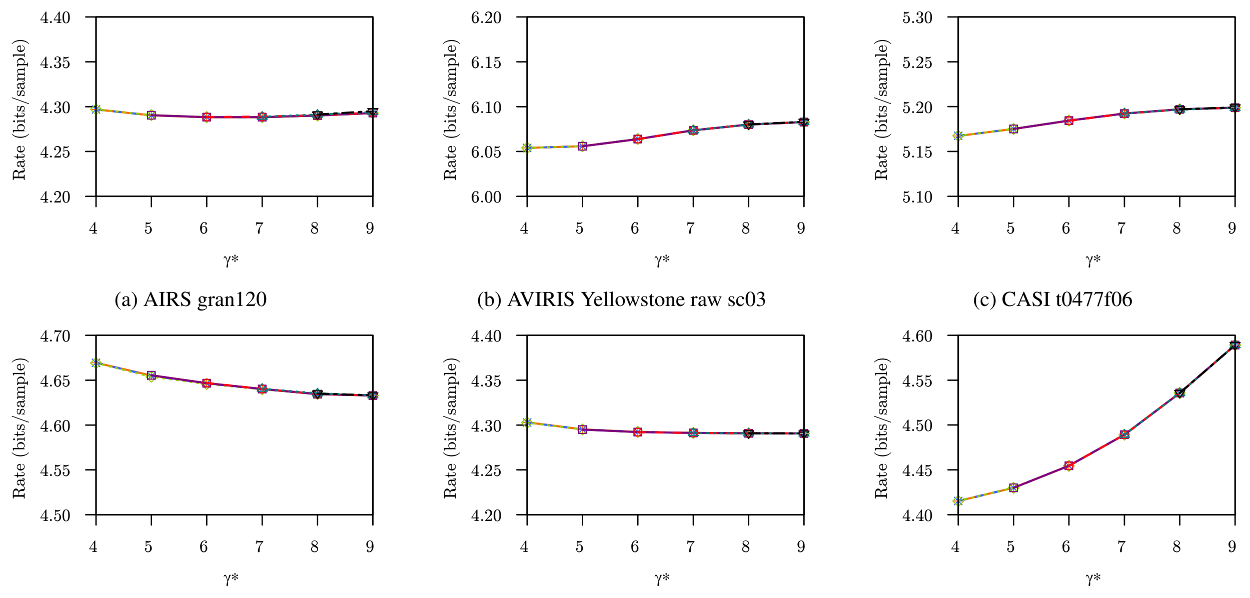
Figure 10: Software compression time, normalized by the time required to perform compression using $P = 0$, reduced prediction mode, and column-oriented local sums. Each curve represents a different combination of prediction mode and local sum type using the following key: —+— *full+neighbor*, - × - *full+column*, ···◇··· *reduced+neighbor*, or —□— *reduced+column*.

Using smaller segments provides increased robustness to data corruption, but it reduces compression effectiveness because (1) each compressed segment includes the overhead cost of an image header, (2) samples at segment boundaries have fewer neighbors for use in prediction, and (3) the predictor and entropy coder take some time to adapt, and so samples near the beginning of a segment will, on average, not be compressed as effectively as later samples. To mitigate this compression performance reduction due to segmentation, the CCSDS-123 standard allows information about the state of the encoder at the end of one segment to be optionally included in the header of the next segment to control the initialization of compressor state variables. Thus, the impact of (3) is reduced at the expense of a slight increase in (1).

Segments can be formed by partitioning an image along any (or all) three axes, but for our discussion we consider only partitioning along the y-axis, which is a natural scenario that might arise for a pushbroom imager. We are therefore interested in the impact of the segment height on compression performance, and we compare different alternatives in passing encoder state information from one segment to the next.

Information about the predictor state can be passed from one segment to the next via $v_{\min}$ (which can be used to convey the weight update scaling exponent, $\rho_{y,x}$) and the optional weight initialization table, which can be used to convey the weight vectors, either exactly or approximately depending on the choice of the weight initialization resolution. Information about the entropy coder state can be passed from one segment to the next via γ_0, which can retain the approximate value of the counter, and the optional accumulator initialization table, which can be used to convey the approximate value of the accumulator for each band. For our experiments, Table 3 defines seven cases that differ in the amount and type of compressor state information passed from one segment to the next.

Two different experiments have been conducted. The first one illustrates the impact of segment height on compressed



(a) AIRS gran120 (b) AVIRIS Yellowstone raw sc03 (c) CASI t0477f06

(d) CRISM frt00009326_07_sc167 (e) Hyperion ErtaAle (f) MODIS night A2001222_0835

Figure 11: Bit rate for different values of γ^* and γ_0. Each curve represents a different value of γ_0 using the following key: —+— 1, – ✕ – 2, ···◇··· 3, —□— 4, – ⊖ – 5, ···△··· 6, —▽— · 7, and —+— · 8.

Table 3: Cases considered for segmentation experiments.

Case	*Weight Initialization Method*	*Q*	$\rho_{y,x}$ *(through* v_{min}*)*	*Accumulator Initialization Table and* γ_0	*Description*
Case 0	Default	–	–	Not Used	Complete reset between segments
Case 1	Default	–	–	Used	Only entropy coder state is kept
Case 2	Custom	$\Omega+3$	Not Passed	Not Used	Only weights are kept
Case 3	Custom	$(\Omega+3)/2$	Not Passed	Not Used	Only weights are kept, but at half resolution
Case 4	Custom	$\Omega+3$	Not Passed	Used	Weights and entropy coder state are kept
Case 5	Custom	$\Omega+3$	Passed	Not Used	Weights and weight update scaling exponent are kept
Case 6	Custom	$\Omega+3$	Passed	Used	Weights, weight update scaling exponent, and entropy coder state are kept

bit rate for the different test cases. Fig. 15 shows the results as a percentage increase in bit rate compared to the image compressed without segmentation. Passing entropy coder state information from one segment to the next appears to have little impact on compression performance. But, at least for some images, there is a notable difference obtained by exploiting predictor state information to initialize the weight vectors and the weight update scaling exponent. In addition, the use of a lower resolution weight initialization table does not seem to produce significant variations.

Fig. 16 shows the results of the second experiment, which measures the relative increase in compressed bit rate when the number of prediction bands is varied while the segment height is fixed at 65. The ranking of the seven different test cases is unchanged from the previous experiment, which reinforces the apparent benefit of retaining state information from the predictor but not the entropy coder. As the number of prediction bands increases, the absolute differences between cases rise substantially thus amplifying the effects of resetting the coder state.

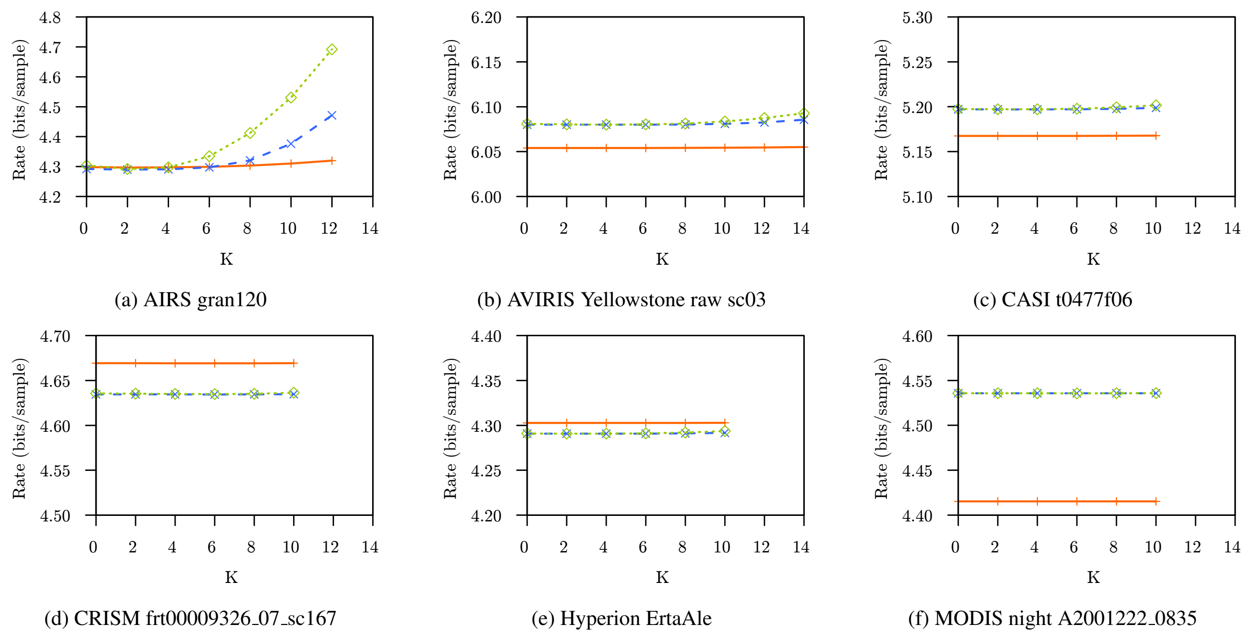


Figure 12: Bit rate as a function of K. Each curve represents a different combination of γ^* and γ_0 using the following key: $\{\gamma^* = 4, \gamma_0 = 3\}$, $\{\gamma^* = 8, \gamma_0 = 3\}$, and $\{\gamma^* = 8, \gamma_0 = 7\}$.

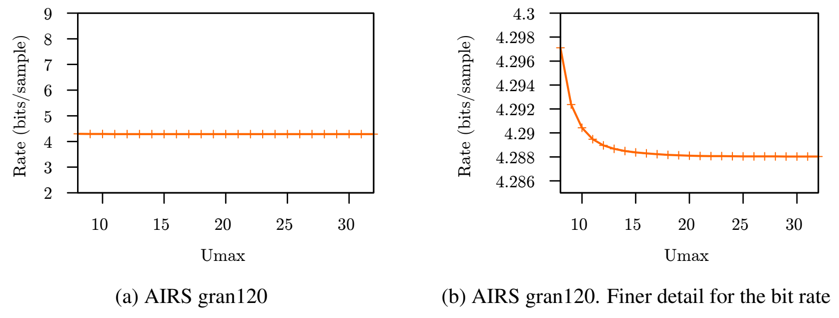


Figure 13: Impact of $U_{\max}$ on compressed bit rate.

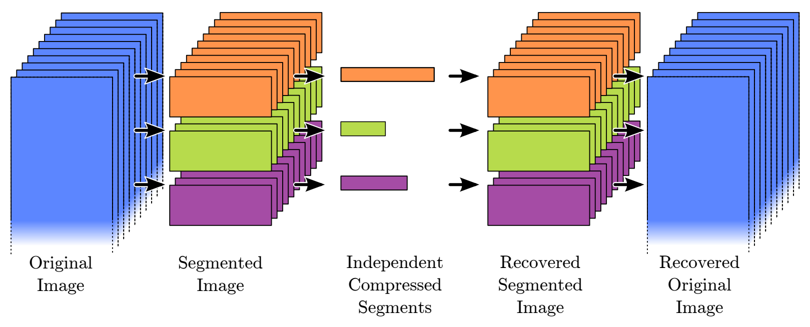
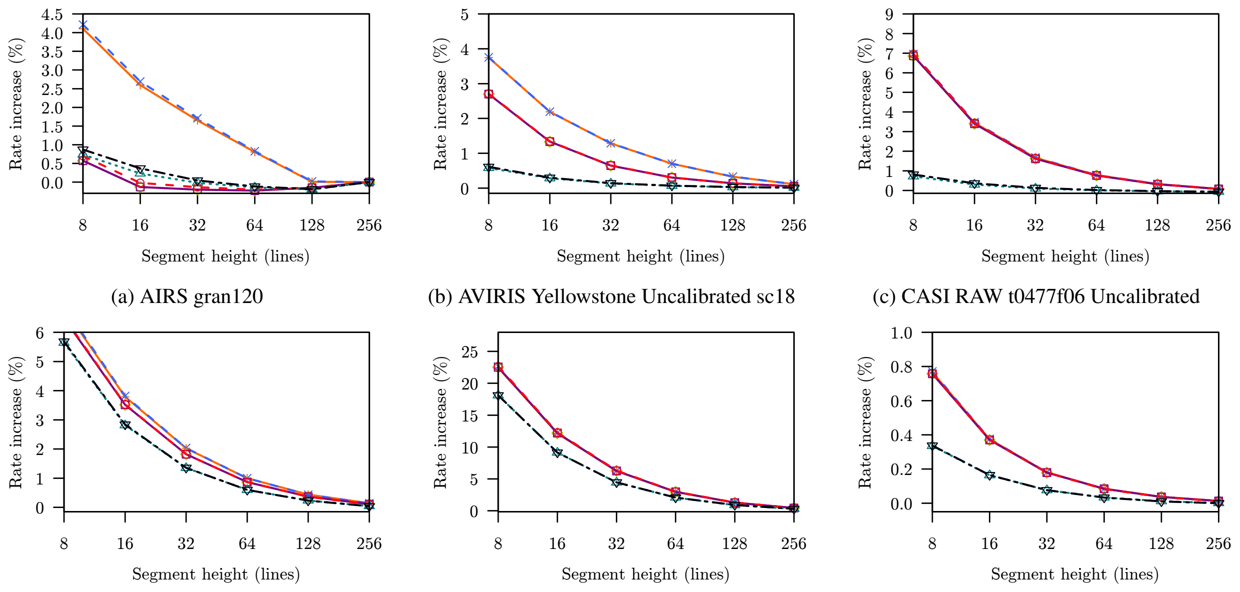


Figure 14: Overview of the segmentation process.

4 Conclusion

This article examines the impact of parameter settings on the overall performance of the CCSDS-123 standard. The results suggest that predictor parameters have much more impact than the entropy coder parameters. In particular, the number of



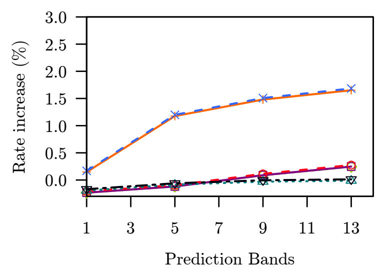
(a) AIRS gran120 (b) AVIRIS Yellowstone Uncalibrated sc18 (c) CASI RAW t0477f06 Uncalibrated

(d) HYPERION MtStHelens raw (e) M3 TARGET M3TargetA (f) MODIS 500m MOD01_A2001123_630

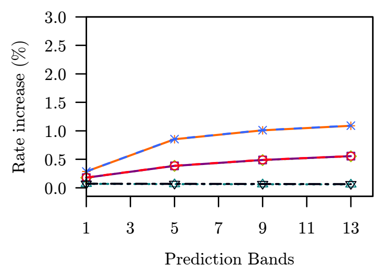
Figure 15: Impact of segment height on compression performance when using segmentation. Each curve represents a different case using the following key: —+— Case 0, - × - Case 1, ···◇··· Case 2, —□— Case 3, - ⊖ - Case 4, ···△··· Case 5, and —▽— Case 6.

prediction bands, the local sum type, the prediction mode, and the value of $v_{\max}$. Based on the image corpus considered, it appears that obtaining good compression performance depends primarily on the optimization of a few parameters, in particular, the local sum type, the predictor mode, and the value of $v_{\max}$; in the other cases, the improvement obtained by parameter optimization is relatively small. The computational complexity is primarily influenced by the number of prediction bands P, the local sum type, and the prediction mode; reducing the number of prediction bands is likely to be the most promising approach to reduce computational complexity while maintaining high compression performance. Not excluding a thorough characterization of the data to be coded, the following broad principles can be derived. If a sensor produces streaking artifacts, *column-oriented* local sum type and *reduced* prediction mode seem appropriate; otherwise *neighbor-oriented* local sum type seems convenient, with *full* or *reduced* prediction modes. For the parameter $v_{\max}$ a value between 2 and 7 seems adequate. As for the number of prediction bands, $P \simeq 3$ seems to provide a reasonable trade-off between performance and complexity.

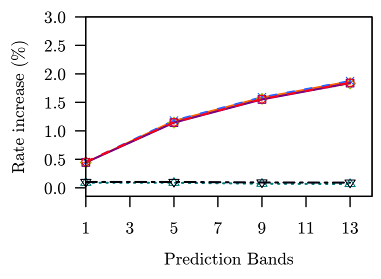
This article also examines the partitioning of images into independently decompressible segments to limit the impact of data corruption that may arise on communications channels. Coding performance suffers as we reduce the size of such segments, but this effect can be mitigated by using the final weight vectors and weight update scaling exponent from one segment to control the initialization of these state variables in the next segment.



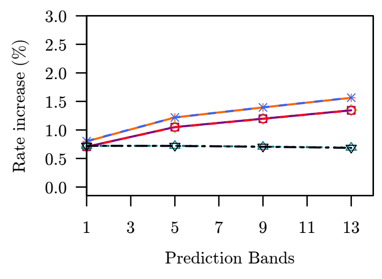
(a) AIRS gran120



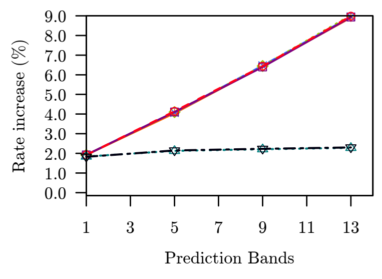
(b) AVIRIS Yellowstone Uncalibrated sc18



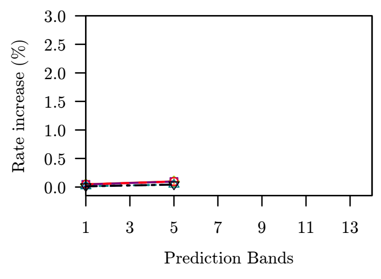
(c) CASI RAW t0477f06 Uncalibrated



(d) HYPERION MtStHelens raw



(e) M3 TARGET M3TargetA



(f) MODIS 500m MOD01_A2001123_630

Figure 16: Impact of the number of prediction bands on compression performance when using segmentation. Each curve represents a different case using the following key: —+— Case 0, – ✳ – Case 1, ···◇··· Case 2, —□— Case 3, – ⊖ – Case 4, ···△··· Case 5, and —▽-· Case 6.

5 Acknowledgments

This work has been partially supported by the Spanish Government, by FEDER, and by the Catalan Government, under grants TIN2009-14426-C02-01, TIN2012-38102-C03-03 (LIFE-VISION) and 2009-SGR-1224. The research conducted by A. Kiely was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

An earlier version of this work has been published in the conference proceedings of the ESA CNES On-Board Payload Data Compression Workshop 2012.

References

[1] Consultative Committee for Space Data Systems (CCSDS). [Online]. Available: http://www.ccsds.org

[2] ——, *Lossless Multispectral & Hyperspectral Image Compression CCSDS 123.0-B-1*, ser. Blue Book. CCSDS, May 2012. [Online]. Available: http://public.ccsds.org/publications/archive/123x0b1ec1.pdf

[3] M. Klimesh, "Low-complexity lossless compression of hyperspectral imagery via adaptive filtering," *IPN Progress Report*, vol. 42-163, pp. 1–10, 2005. [Online]. Available: http://ipnpr.jpl.nasa.gov/progress report/42-163/163H.pdf

[4] Consultative Committee for Space Data Systems (CCSDS), *Lossless Data Compression CCSDS 121.0-B-2*, ser. Blue Book. CCSDS, May 2012. [Online]. Available: http://public.ccsds.org/publications/archive/121x0b2.pdf

[5] G. S. M. Weinberger and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," *IEEE Trans. Image Process*, vol. 9, no. 8, pp. 1309–1324, 2000.

[6] M. Klimesh, A. Kiely, and P. Yeh, "Fast lossless compression of multispectral and hyperspectral imagery," in *2nd Int. WS on On-Board Payload Data Compression (OBPDC)*. ESA, October 2010.

[7] GICI, "Emporda," http://gici.uab.cat/GiciWebPage/downloads.php, 2012.

Figure 1: Overview of a CCSDS-123 implementation.

Figure 2: Samples used to calculate the local sum $\sigma_{z,y,x}$.

Figure 3: Local differences used under the different prediction modes. Only differences depicted in blue are used in each case.

Figure 4: Compressed bit rate performance as a function of P.

Figure 5: Compressed bit rate for different choices of prediction mode and local sum type. Each curve represents a different combination using the following key: —+— *full+neighbor*, - × - *full+column*, ··◇·· *reduced+neighbor*, or —⊟— *reduced+column*.

Figure 6: Cumulative running average bit rate. Each curve represents a different combination of prediction mode and local sum type using the following key: —+— *full+neighbor*, - × - *full+column*, ··◇·· *reduced+neighbor*, or —⊟— *reduced+column*.

Figure 7: Bit rate for different choices of parameters that affect the adaptability of the predictor to the image. Each curve represents a different combination of t_{inc} and v_{min} using the following key: —+— $\{t_{\text{inc}} = 2^4, v_{\text{min}} = -6\}$, - × - $\{t_{\text{inc}} = 2^4, v_{\text{min}} = -1\}$, ··◇·· $\{t_{\text{inc}} = 2^7, v_{\text{min}} = -6\}$, —⊟— $\{t_{\text{inc}} = 2^7, v_{\text{min}} = -1\}$, - ⊖ - $\{t_{\text{inc}} = 2^{10}, v_{\text{min}} = -6\}$, and ··△·· $\{t_{\text{inc}} = 2^{10}, v_{\text{min}} = -1\}$.

Figure 8: Bit rate depending on v_{min} and t_{inc} for a fixed v_{max}. Values of t_{inc} are represented using the following key: —+— 2^4, - × - 2^5, ··◇·· 2^6, —⊟— 2^7, - ⊖ - 2^8, ··△·· 2^9, —▽—· 2^{10}, and —+—· 2^{11}.

Figure 9: Bit rate for different values of R and Ω. Each curve represents a different value of Ω using the following key: —+— 4, - × - 8, ··◇·· 12, and —⊟— 16.

Figure 10: Software compression time, normalized by the time required to perform compression using $P = 0$, reduced prediction mode, and column-oriented local sums. Each curve represents a different combination of prediction mode and local sum type using the following key: —+— *full+neighbor*, - × - *full+column*, ··◇·· *reduced+neighbor*, or —⊟— *reduced+column*.

Figure 11: Bit rate for different values of γ^* and γ_0. Each curve represents a different value of γ_0 using the following key: —+— 1, - × - 2, ··◇·· 3, —⊟— 4, - ⊖ - 5, ··△·· 6, —▽—· 7, and —+—· 8.

Figure 12: Bit rate as a function of K. Each curve represents a different combination of γ^* and γ_0 using the following key: —+— $\{\gamma^* = 4, \gamma_0 = 3\}$, - × - $\{\gamma^* = 8, \gamma_0 = 3\}$, and ··◇·· $\{\gamma^* = 8, \gamma_0 = 7\}$.

Figure 13: Impact of U_{max} on compressed bit rate.

Figure 14: Overview of the segmentation process.

Figure 15: Impact of segment height on compression performance when using segmentation. Each curve represents a different case using the following key: —+— Case 0, - × - Case 1, ··◇·· Case 2, —⊟— Case 3, - ⊖ - Case 4, ··△·· Case 5, and —▽—· Case 6.

Figure 16: Impact of the number of prediction bands on compression performance when using segmentation. Each curve represents a different case using the following key: —+— Case 0, - × - Case 1, ··◇·· Case 2, —⊟— Case 3, - ⊖ - Case 4, ··△·· Case 5, and —▽—· Case 6.