Reduced Memory Multi-Layer Multi-Component Rate Allocation for JPEG2000

Prajit Kulkarni[*1], Ali Bilgin[1], Michael W. Marcellin[1], Joseph C. Dagher[1], Thomas Flohr[2] and Janet Rountree[2]

[1]Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ USA 85721
[2]Science Applications International Corporation (SAIC), Tucson, AZ USA 85711

ABSTRACT

Remote sensing images are often multispectral in nature and are acquired by on-board sensors in a "push-broom" fashion. These images are compressed and transmitted to ground stations for further analysis. Since they are extremely large, buffering all acquired data before encoding requires huge amounts of memory and introduces latency. Incremental compression schemes work on small chunks of raw data as soon as they are acquired and help reduce buffer memory requirements. However, incremental processing leads to large variations in quality across the reconstructed image. We propose two "leaky bucket" rate control algorithms that can be employed for incrementally compressing hyperspectral images using JPEG2000. Both schemes perform rate control using the fine granularity afforded by JPEG2000. The proposed algorithms have low memory requirements and enable SNR scalability through the use of quality layers. Experiments show that the proposed schemes provide significant reduction in quality variation with no loss in mean overall PSNR performance.

Keywords: image compression, JPEG2000, incremental processing, rate allocation, hyperspectral imagery

1. INTRODUCTION

With the increased use of remote sensing data, the need for efficient transmission of such data has gathered greater importance in recent times. Remote sensing images are often multispectral in nature and are acquired by on-board sensors in a "push-broom" fashion. One way to handle such acquisition is to buffer the entire image and then perform the compression operation on the data. This, however, is highly undesirable. Since remote sensing data are often quite large, buffering of the entire image requires large amounts of on-board storage. Secondly, buffering introduces a delay into the processing system. Thus, it is of interest to consider algorithms that can process and transmit data incrementally, thus reducing latency and memory requirements. However, incremental processing can lead to large variations in quality across the reconstructed image. This problem is addressed here by careful design of rate control algorithms.

Some research has gone towards developing rate control algorithms designed specifically for on-board operations on space remote-sensing missions ([1]–[6]). The DCT-based rate control scheme of [1] divides the acquired data into units called segments. Rate control is performed on a segment-by-segment basis. The algorithm is a variation of the JPEG standard and its performance is slightly inferior to that of JPEG due to the push-broom approach. The method uses a quantization table which is updated to control the data rate (compression ratio).

Discrete Wavelet Transform (DWT) techniques have become popular because of their advantages over other techniques such as DPCM and DCT [2]. A low complexity rate constrained wavelet coder is presented in [3]. This system supports hierarchical processing of wavelet subband coefficients. Such processing can be used to design a rate control loop for multispectral images ([4], [5] and [6]). In [4], the spectral components are decorrelated to remove

[*] prajit@ece.arizona.edu

Image and Video Communications and Processing 2005, edited by Amir Said, John G. Apostolopoulos, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 5685 © 2005 SPIE and IS&T · 0277-786X/05/$15

spectral redundancy. Then, the wavelet coefficients are encoded in a hierarchical manner. Different quantization step sizes are used for different levels of the multiresolution pyramid. The schemes described in [5] and [6] use feedback from a regulator to allocate bits to the various wavelet subbands.

The use of JPEG2000 in compression applications is desirable because of its state-of-the-art performance and the inherent scalable nature of its compressed codestream. Research into rate control methods that can be used in conjunction with JPEG2000 is described in [7]–[10]. In [7], the efficiency of the Post Compression Rate-Distortion Optimization (PCRD-opt [11], [12]) algorithm used in JPEG2000 is enhanced using a byte-slope table that stores the number of bytes that have been contributed to each possible R-D slope value. Based on entries in this table, it is decided if a coding pass will be included in the final codestream. If not, it is not coded at all. The table is updated each time a coding pass is encoded. This removes the wasted computation and memory usage in encoding and storing data that will be thrown away when the final codestream is produced.

Work in [8] and [9] deal with scan-based approaches to rate control in JPEG2000. The PCRD-opt algorithm is used to form embedded bitstreams for "stripes" in the image and these are combined to form a composite final codestream. Such incremental processing is made possible without "hard seams" at stripe boundaries because of reduced memory wavelet decomposition algorithms ([12], [13]) that may be used in JPEG2000 compression. Such transforms compute the transform in an incremental fashion. That is the wavelet coefficients for the "top" of the image are computed after only a few rows of image data are available. Such processing is possible because the filters employed in the wavelet transform have finite regions of support.

In [8], a quality control loop is used with a scan-based wavelet transform algorithm to meet a final MSE target. Based on the MSE of previously released image portions, the quantization step size for each subband of each stripe is varied to minimize the total bit rate for a target MSE. In [9], a sliding-window approach for rate control is used. Here, stripes of the input image, called scan elements, are processed and stored in a rate control buffer whose size depends on the target rate. Each time a new scan element is added to the buffer, an R-D slope threshold is computed for the buffer and coding passes having slope values lower than this threshold are deleted. Data are removed from the buffer and transmitted at a fixed rate corresponding to the desired amount of compression. Similar algorithms are described in [14] for compressing gray scale video.

The work presented in this paper extends that described in [9] and [14]. In particular, we present the development of a novel rate control method for JPEG2000 for multi-component images and for multi-layered codestreams. We present two rate control algorithms that have been successfully tested on remote sensing data.

Section 2 provides an overview of JPEG2000. In Section 3, we present our work, the Multi-layer Sliding Window Rate Control (M-SWRC) and Extended Sliding Window Rate Control (M-EWRC) algorithms, along with results. Section 4 is a note on our conclusions and future work.

2. OVERVIEW OF JPEG2000

This section provides a brief overview of JPEG2000. For more details, the reader is referred to [10].

When the image to be compressed is input to the JPEG2000 encoder, the image is first divided into tiles. It is allowable to include the entire image in a single tile, (thus essentially eliminating the use of tiles). Each tile is compressed independently, thus enabling one method of low-memory operation. The disadvantage of using tiles is that, at low rates, images may suffer from artifacts because tile boundaries may become visible.

An (optional) component transform is performed on each tile. The wavelet transform is then performed on each resulting tile component. Each resolution of a wavelet-transformed tile-component is partitioned into *precincts*. Each precinct corresponds to a resolution level of one spatial region. The precinct size can be chosen independently by resolution, but each dimension of the precinct size must be a power of 2. Each wavelet transform subband is also

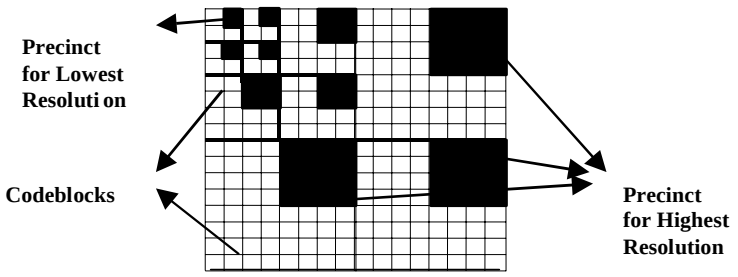

Fig. 1. Relationship between Precincts and Codeblocks.

divided into rectangular regions, referred to as *codeblocks*. Fig. 1 shows the relationship between precincts and codeblocks for three levels of wavelet transform (4 resolutions).

The wavelet coefficients in each codeblock are quantized and independently coded using a bitplane coder. This coding involves context-dependent, binary, arithmetic coding of bitplanes of the quantized coefficient values. A collection of bits, one from the binary representation (of the magnitude) of each wavelet coefficient in a codeblock (all having the same significance), forms a bitplane. For example, the most significant bits from each coefficient comprise the most significant bitplane. Each bitplane is encoded using three *coding passes*, starting from the most significant bitplane. Each bit in a bitplane is encoded in exactly one of the three coding passes – significance propagation, magnitude refinement or clean-up.

Each coding pass has a *distortion-length slope* associated with it. The distortion-length slope of a coding pass is defined as

$$S = \frac{\Delta D}{\Delta R},$$

where, ΔD is the reduction in image distortion obtained by including the coding pass, and ΔR is the number of bits used to code the coding pass.

The encoder computes the distortion-length slope of every coding pass and uses it to guide codestream formation as described below. Each codeblock produces an elementary embedded bitstream. The bitstreams from different codeblocks can be independently truncated to different lengths to satisfy certain constraints.

In a typical JPEG2000 encoder, selection of truncation points is done after all coding passes of all codeblocks in a tile (or image) have been generated, at which point the distortion-length slopes of all the coding passes are known. The higher the slope value of a coding pass, the more "important" it is, since its inclusion leads to a greater reduction in distortion (per bit) than that obtained by including a coding pass with a smaller slope value. Thus, the coding passes with the largest distortion-length slopes (subject to a constraint on the desired file size) are included in the file. This optimization strategy is known as *Post-Compression Rate-Distortion Optimization* and minimizes the distortion (typically MSE) of the compressed/decompressed image. Conceptually, this optimization is performed by sorting the coding passes in descending order of distortion-rate slope, and including coding passes starting from the top of the list, until the byte budget is exhausted.

Once the encoder decides on which coding passes will be included in the codestream, the final task is to form the codestream. The fundamental unit used in the construction of a JPEG2000 codestream is called a *packet*. A packet contains the compressed data from a varying number of coding passes from the codeblocks in one precinct of one tile-component. Any number of coding passes (including zero) can be included from each codeblock. The packets are then organized into *layers*. A layer is defined as a collection of packets, one from every precinct, of every resolution of every tile-component of a tile. A packet provides one quality increment for one spatial region (precinct), at one resolution of one tile-component. Since a layer consists of one packet from every precinct of each resolution of each tile-component, it provides one quality increment for an entire tile. The introduction of layers into a codestream facilitates progression by quality (SNR scalability) decoding. The layered codestream can be used to enable many applications. For example, a layered codestream can be used to transmit data over a time-varying communication channel. Depending on the available bandwidth, varying numbers of layers can be transmitted and decoded. Similarly, a layered codestream can be used in a multicast environment where different channels have different bandwidths. In this case, a single multi-layered codestream would meet the bandwidth requirements of all the channels, with each layer meeting the target rate of a different channel.

Besides compressed coding passes, the JPEG2000 codestream includes information necessary for decompression of the codestream. The codestream starts with the main header. The main header contains global information that is necessary for decompression. This includes the sizes of the geometric structures (image, tile, precinct, codeblock), as well as coding parameters. The main header is followed by a sequence of tile-streams. Each tile-stream starts with a tile header that contains information related to that particular tile. Tile headers can be used to overwrite some of the coding parameters specified in the main header. The body of a tile-stream contains the packets belonging to the tile. A two-byte marker called the EOC (End of Codestream) terminates the codestream.

3. MULTI-LAYER MULTI-COMPONENT RATE ALLOCATION

In the absence of tiles, a typical JPEG2000 encoder compresses and buffers all coding passes of all codeblocks prior to forming any packets. This requires storage corresponding to somewhat more than the *compressed* image. This is done to ensure uniform quality from top to bottom of the image and works well in most cases. However, in some applications (remote-sensing in particular), buffering all data, even compressed, is not feasible. One approach to circumventing this problem is to employ tiles (with the risk of artifacts as discussed above). Fortunately, JPEG2000 supports another approach to low memory compression. This approach is enabled by the use of precincts. As described above, a precinct corresponds to one spatial region at one resolution. Precincts can be thought of loosely as tiles, but in the wavelet domain. Since there are no seams (breaks) in the wavelet transform itself, precincts do not cause artifacts like tiles do. Packets from one precinct can be encoded independently of those from other precincts. Thus, when an incremental wavelet transform is employed, packets can be encoded and transmitted in a top-to-bottom fashion without buffering the entire (compressed) image.

While the JPEG2000 standard envisions employing precincts in this fashion, it does not require any specific rate control technique. Specifically, there is no required method that attempts to achieve uniform quality from top to bottom within an image. This is the main subject, and contribution, of this paper.

Before a detailed treatment of rate control, we introduce the idea of a *scan element*. As discussed above, precinct sizes can be chosen independently by resolution level. In this work, we chose precinct sizes in a specific fashion so that there is a correspondence between precincts from different resolutions. Specifically, corresponding precincts from different resolutions correspond to the same spatial region. We call each set of corresponding precincts a scan element. To achieve the goal of low memory processing, the spatial region of a scan element should be "short," corresponding to a small number of image lines. However, the width can generally as wide as the image. With this choice, the spatial region of a scan element corresponds to a "stripe" of image pixels. This situation is depicted in Fig. 2. This figure shows a number of scan elements for one component with 2 levels of wavelet transform. Here, regions in different subbands shaded with the same color correspond to the same region of the image in the spatial domain. As mentioned before, the formation of scan elements using small amounts of data from different subbands is possible because of the use of an incremental wavelet transform. This concept of scan elements can be extended to any number of resolution levels, with contributions from subbands getting smaller (by powers of 2) at the lower resolutions.

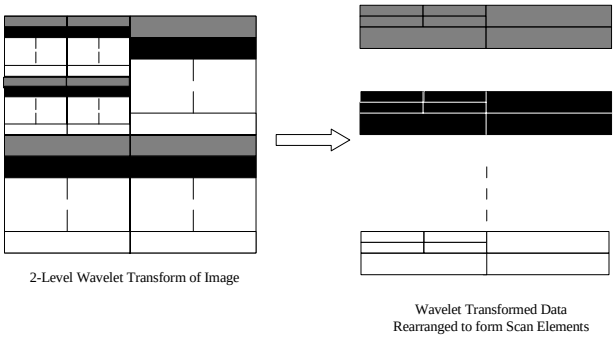

Fig. 2. Relationship between Scan Elements and Resolution Subbands for a Single Component.

All results presented in this paper were obtained using the hyperspectral AVIRIS [15] data set ("Low Altitude"). This data set has 224 components and the dimensions of the image are 3584 pixels (high) by 614 pixels (wide). In all cases, we employ scan elements corresponding to 32 rows of image data. Therefore, 112 scan elements are required to completely cover the image. Two levels of wavelet transform are applied with a codeblock size of 16x256. Four

quality (SNR scalable) layers are used, corresponding to 0.5 bpp/component (112 bpp), 1.0 bpp/component (224 bpp), 1.5 bpp/component (336 bpp) and 2.0 bpp/component (448 bpp), respectively. In all figures, results are reported for a rate of 448 bpp (2 bpp/component), unless mentioned otherwise. This is the highest quality layer among the four layers in the codestream. In this paper, compression performance is measured using Peak Signal-to-Noise Ratio (PSNR) numbers that quantify the amount of distortion that the compression process introduces into the image.

Before we introduce our sliding window based rate control, we thoroughly explore the "obvious" approach. In this approach, each scan element is encoded at the same (constant) rate (in bits per pixel). Equivalently, each scan element experiences the same compression ratio. While simple, this method has some problems. Scan elements that are easy to compress (e.g. have low detail) will end up being compressed with higher quality than harder-to-compress scan elements.

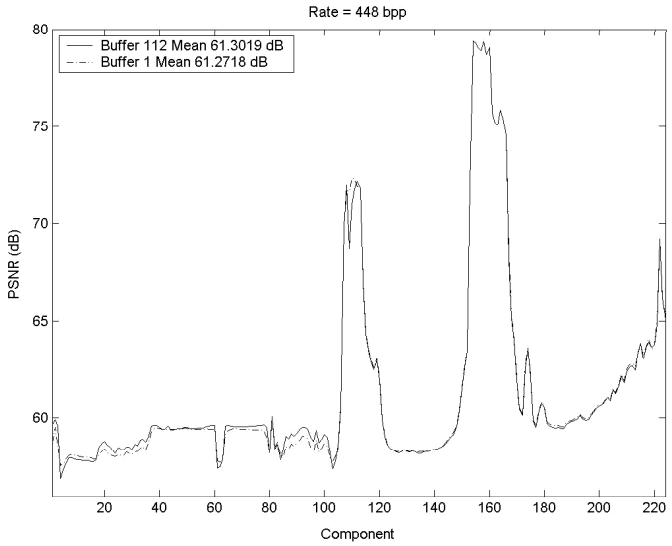

Fig. 3. Comparison of Component PSNR values when Buffering all 112 Elements vs. Fixed-rate Encoding (Buffering 1 Scan Element).

Using the parameters outlined above, Fig. 3 compares the PSNR of individual image components via two different treatments. While both employ scan elements, one employs fixed rate encoding for each scan element, while the other buffers all scan elements before creation of the final codestream. It is seen that the mean PSNR over all components differs by only about 0.03 dB between the two cases. From these results, it is obvious that there is almost no loss in PSNR performance even when individual scan elements are compressed at a fixed rate. Hence, we will not consider overall PSNR performance in the remainder of this paper. However, this does not take into account possible quality differences from top to bottom within an image.

This is illustrated by Fig. 4, which shows the PSNR for each individual scan element of a particular component. The spread in PSNR across scan elements is nearly 3.5 dB. The easily compressible scan elements have high PSNR and the ones with large image information have lower PSNR. For comparison, the case when all data are buffered is also included. It can be seen that the unbuffered case has a large variation in the quality across each component, even though the mean overall PSNR is close to when all data are buffered. This is the drawback of fixed-rate encoding. Though plots have been shown for only one rate (2 bpp/component), they are similar for other rates.

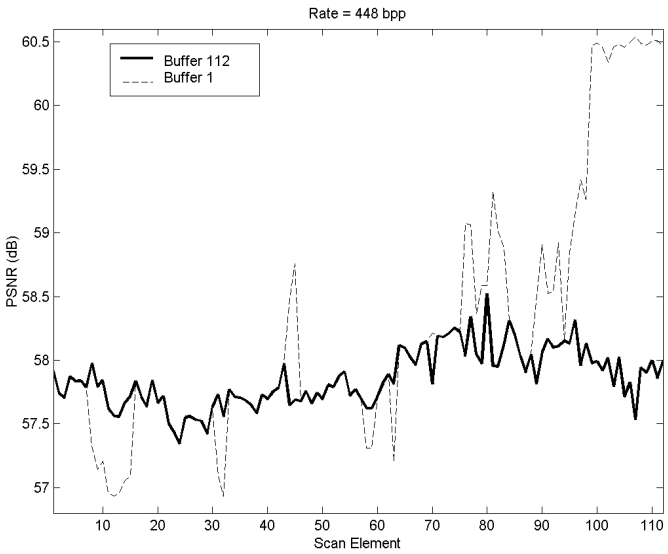

Fig. 4. Scan Element PSNRs for Component 84.

Our goal in what follows is to maintain the quality fairly constant across the image without buffering the entire image. To this end, we employ a "leaky bucket" approach to rate control. Fig. 5 shows the block diagram of a scheme for low-memory incremental processing of large multi-component images. The buffer

control mechanism performs rate control on the incremental compressed data at the output of the bitplane coder and fills up the rate control buffer. The result of this rate allocation process is that each scan element is encoded at a different rate depending on how "difficult" it is to compress. Even so, data are taken out of the buffer for transmission at a constant rate. Layers are formed each time a new compressed scan element is added to the rate control buffer. This is in contrast to the "normal" JPEG2000 encoder where layers are formed after all compressed data are buffered for the entire image. The salient feature of leaky bucket rate control schemes is that they have low memory requirements since only a moderate number of compressed scan elements are buffered.

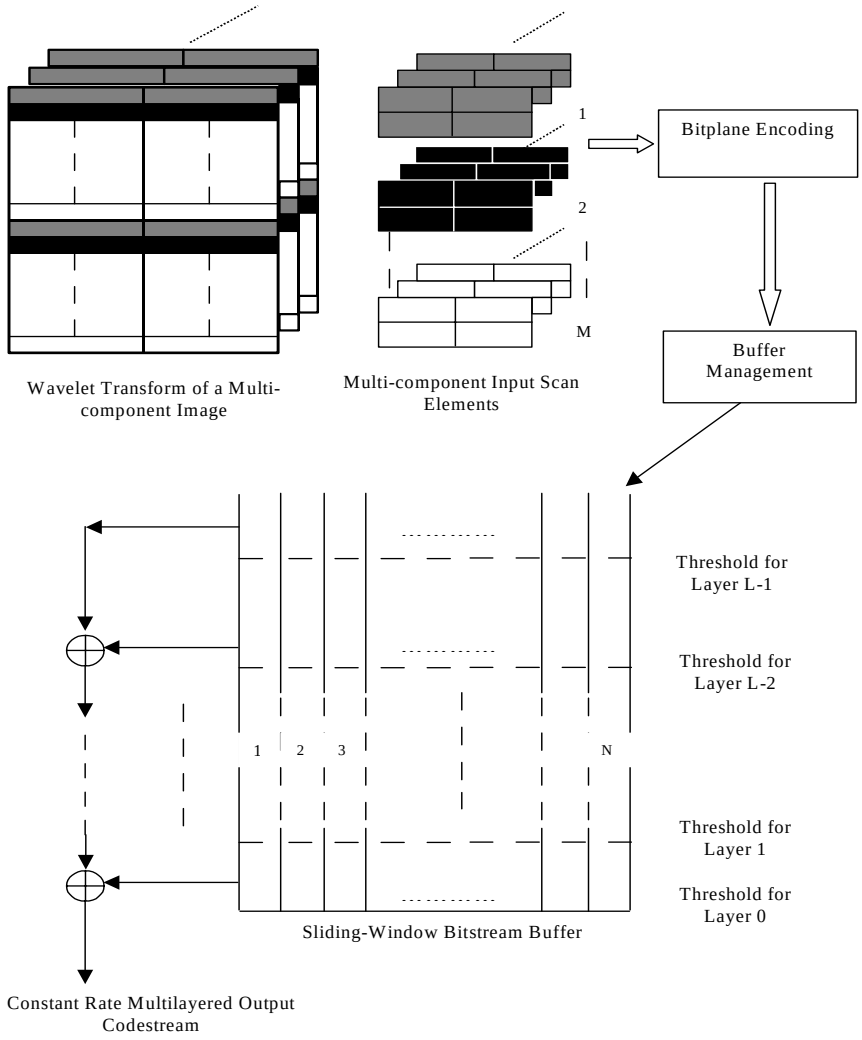

Fig. 5. Block Diagram of Leaky-Bucket Rate Control Algorithm for Incremental Processing of Large Multi-Component Images with Multiple Quality Layers.

Two leaky-bucket algorithms for multi-layer multi-component rate control have been implemented. Consider a fixed-rate downlink that is designed to carry *M* bits (corresponding to an entire compressed image) in *T* seconds. The fixed rate of transmission on the link is *M/T* bits/sec. If the number of scan elements is *N*, we (remove from the rate control buffer and) transmit *M/N* bits of data every *T/N* seconds. For fixed rate encoding, the *M/N* bits transmitted each time correspond (exactly) to one complete scan element. In the leaky bucket case, this correspondence is only nominal (or average). The actual compressed size varies from one scan element to another. The algorithms are described in more detail in the following sections.

3.1. Multi-Layer Sliding Window Rate Controller (M-SWRC)

The algorithm was first described in [9] and was adapted to video coding in [14]. Each scan element is quantized, encoded using a bitplane coder and added to the rate control buffer. The size of the buffer is determined in terms of the number of scan elements (nominally held) in the buffer. Each time a new scan element is to be added to the buffer, a distortion-rate slope threshold is calculated for each desired codestream layer such that all the compressed scan elements fit in the buffer. All coding passes belonging to the scan elements in the buffer and to the new scan element, with slopes lower than the *lowest layer threshold*, are deleted. When enough scan elements have been processed to fill

up the buffer, the multi-layered data are pulled out of the buffer at a constant rate and new data are added using the scalability property of the bitstreams. Thus, one scan element is processed every time in a "sliding window" fashion. The algorithm is described in more detail in Table I. In this table, a quantity called *excess bytes* is used. As mentioned, the data being removed from the buffer correspond (nominally) to one compressed scan element. However, since each scan element is encoded at a different rate, the data may correspond to only a portion of a scan element. In that case, only a part of the scan element is removed from the buffer. The remaining portion of the scan element is "frozen" in the buffer and no more rate-distortion optimization decisions are made on this data when the next scan element is added to the buffer. This portion of data is flushed along with the next scan element to be flushed out. The portion of the buffer occupied by this frozen chunk of data is not available when the next PCRD-optimization is performed on the buffer. The excess bytes value for each layer stores the size of the contribution from that layer to this frozen portion of the buffer and is used when layer thresholds are calculated.

Similarly, the data rate may be such that the scan element to be released may not be large enough to utilize the entire available bandwidth. In this case, portions of an additional scan element (or elements, depending on their sizes) have to be released in order to maintain a constant rate output. Again, if a portion of a scan element is left behind after the previous flush operation, the part of the buffer occupied by this data is frozen and the excess bytes value for each of the layers is updated to reflect this change. This process is repeated until all scan elements are processed.

TABLE I
M-SWRC ALGORITHM FOR MULTIPLE LAYERS

Let the total number of scan elements for an image be N, each corresponding to P pixels. Let the number of layers to be generated be L, with corresponding rates (in bpp) $R_0, R_1,..., R_{(L-1)}$. Let the nominal number of scan elements in the buffer be S. Let the target sizes corresponding to the buffered portion of the L layers be $B_0, B_1,... B_{(L-1)}$ bytes. We have $B_l = PSR_l/8$, $l=0, 1, 2,..., L-1$.
The nominal number of bytes contributed by each layer of a single scan element to the final composite codestream (flush bytes) is given by $F_l = PR_l/8$, $l=0, 1, 2,..., L-1$.
The fixed transmission rate is such that $PR_{(L-1)}/8$ bytes are removed from the buffer before each new scan element is added.

Then, the Sliding Window Rate Control algorithm is described as follows:

1. Start with the buffer empty and reset the value of excess bytes for each layer, i.e., $E_l = 0$, $l=0, 1, 2,..., L-1$.

2. When the first scan element has been processed and is ready to be buffered, compute the distortion-rate slope threshold for each layer, $T_{DR}(l)$, such that the total size of the coding passes from the scan element that contribute to each layer, and have slopes $\geq T_{DR}(l)$ does not exceed the target size for that layer, B_l. Delete all coding passes of the scan element, with slope values $< T_{DR}(L-1)$, the lowest threshold among all the layer thresholds. If $S = 1$, go to Step (6).

3. When the next scan element has been processed and is ready to be buffered, for each layer l, $l \in \{0,1,..., L-1\}$, perform the following operations:
   a. Calculate the new layer target size as $b_l = B_l - E_l$.
   b. Compute the distortion-rate threshold, $T_{DR}(l)$, such that the total size of all qualifying coding passes, from the new scan element and from all the unfrozen scan elements already in the buffer, that contribute to this layer, does not exceed the target size for this layer, b_l.

4. Delete all coding passes of the new scan element and those of the unfrozen scan elements in the buffer, with slope values $<T_{DR}(L-1)$.

5. If buffer has not been initialized, repeat Steps 3 and 4 until S scan elements have been processed and buffered. Now, the buffer is initialized and data transmission can begin.

6. For each layer $l, l \in \{0,1,..., L-1\}$, perform the following operations:

a. Flush out E_l bytes of data (if any), which form the remaining portion of a previous (partially flushed) scan element. Update the value of number of bytes to be flushed, given by, $f_l = F_l - E_l$.
b. Compute the contribution, O_l, of the qualifying coding passes from the next scan element to be flushed, to this layer.

7. If $O_{(L-1)} = f_{(L-1)}$, the size of scan element being released fits the output byte requirement exactly. For each layer $l, l \in \{0,1,..., L-1\}$, update value $E_l = 0$. Go to step (10). Else, go to step (8).

8. If $O_{(L-1)} > f_{(L-1)}$, the total number of bytes being released is greater than the allotted number and some data from the current scan element being flushed will remain in the buffer. For each layer $l, l \in \{0,1,..., L-1\}$, update value $E_l = O_l - f_l$. Go to step (10). Else, go to step (9).

9. If $O_{(L-1)} < f_{(L-1)}$, releasing one scan element does not meet the output byte requirement. We need to release an additional scan element from the buffer. For each layer l, $l \in \{0,1,..., L-1\}$, compute the total contribution, O_l, from this layer. Go to step (7)

10. Release scan element (or elements, as required) from the buffer. Repeat steps (3) through (9) until all scan elements have been processed and buffered. No further calculation is then required. Flush out $F_{(L-1)}$ bytes at a time, until all data have been flushed.

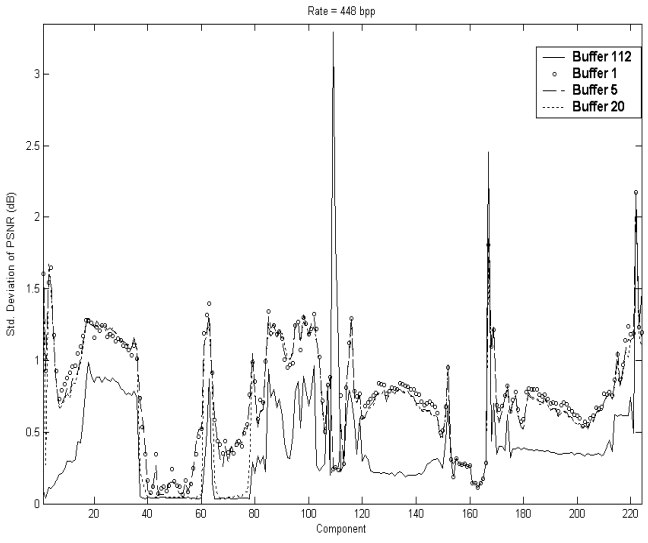

Fig. 6. Standard Deviation of PSNR by Component for M-SWRC.

Fig. 6 shows the variation in PSNR across scan elements for each component obtained with the M-SWRC for various buffer sizes. It is seen that the variation in PSNR decreases as the buffer size is increased. This is because a bigger buffer means that quality fluctuation among a larger number of scan elements can be considered by the rate control scheme during bit allocation, thus providing greater uniformity in quality from top to bottom within the image.

3.2. Multi-Layer Extended Sliding Window Rate Controller (M-EWRC)

With the Multi-Layer SWRC, all coding passes with slopes lower than the lowest threshold are deleted. However, under certain conditions, we may prefer to have a few of the coding passes that were deleted in previous optimization steps. This situation may arise when most of the coding passes of the new scan element being added to the buffer have slopes lower than the current lowest threshold, i.e., the new scan element is "easy" to compress. Since we have already deleted the required coding passes, we are left to include "extra" coding passes from the "easy" scan element. To avoid this, we use the M-EWRC algorithm where an extended buffer is maintained. This algorithm can be understood as maintaining an extra layer in an enlarged buffer with all the

processing being identical to that in M-SWRC using $L+1$ layers. But data from only L layers are transmitted. This extra layer in the buffer is used to hold some of the coding passes, from the scan elements in the buffer, which normally would have been discarded if the M-SWRC were used. When an easily compressible scan element is added to the buffer, these coding passes can be allocated to the first L layers. The use of the extended buffer improves the performance of the rate control algorithm. Compared to M-SWRC, tighter quality control is performed and the variation in quality between scan elements if further reduced.

The algorithm is described in more detail in Table II. In this table, a quantity called the *Effective Buffer Ratio* (*EBR*) has been used. It is defined as follows:

$EBR = \frac{B_2}{B_1}$, where, B_2 = size of buffer used in M-EWRC,

B_1 = size of buffer used in M-SWRC.

EBR has a value greater than or equal to 1. When $EBR = 1$, the M-EWRC algorithm is identical to the M-SWRC algorithm.

TABLE II
M-EWRC ALGORITHM FOR MULTIPLE LAYERS

Let the total number of scan elements for an image be N, each corresponding to P pixels. Let the number of layers to be generated be L, with corresponding rates (in bpp) $R_0, R_1,..., R_{(L-1)}$. Let the nominal number of scan elements in the buffer be S. Let the target sizes corresponding to the buffered portion of the L layers be $B_0, B_1,..., B_{(L-1)}$ bytes. We have $B_l = PSR_l/8$, $l=0, 1, 2,..., L-1$. Let the ***effective buffer ratio*** be *EBR*.
The nominal number of bytes contributed by each layer of a single scan element to the final composite codestream (flush bytes) is given by $F_l = PR_l/8$, $l=0, 1, 2,..., L-1$.

The fixed transmission rate is such that $PR_{(L-1)}/8$ bytes are removed from the buffer before each new scan element is added.

Then, the Extended Sliding Window Rate Control algorithm is described as follows:

1. Start with the buffer empty and reset the value of excess bytes for each layer, i.e., $E_l = 0$, $l=0, 1, 2,..., L-1$.

2. When the first scan element has been processed and is ready to be buffered, compute the distortion-rate slope threshold for each layer, $T_{DR}(l)$, such that the total size of the coding passes from the scan element that contribute to each layer, and have slopes $\geq T_{DR}(l)$ does not exceed the target size for that layer, B_l. Also compute the threshold $T_{EBR}(L-1)$ for the extended buffer of size $EBR^* B_{L-1}$. Since $EBR > 1$, $T_{EBR}(L-1) < T_{DR}(L-1)$.Delete all coding passes belonging to the scan element, which have slope values smaller than $T_{EBR}(L-1)$. If $S = 1$, go to Step (6).

3. When the next scan element has been processed and is ready to be buffered, for each layer l, $l \in \{0,1,..., L-1\}$, perform the following operations:
   a. Calculate the new buffer size as $b_l = B_l - E_l$.
   b. Compute the distortion-rate threshold, $T_{DR}(l)$, such that the total size of all qualifying coding passes, from the new scan element and from all the unfrozen scan elements already in the buffer, that contribute to this layer, does not exceed the target size for this layer, b_l.

4. Compute the threshold $T_{EBR}(L-1)$ and delete all coding passes belonging to the new scan element and to the unfrozen scan elements in the primary and secondary buffers, with slope values $< T_{EBR}(L-1)$.

5. If buffer has not been initialized, repeat Steps 3 and 4 until S scan elements have been processed and buffered. Now, the buffers are initialized and data transmission can begin.

6. For each layer $l, l \in \{0,1,..., L-1\}$, perform the following operations:

a. Flush out E_l bytes of data (if any), which form the remaining portion of a previous (partially flushed) scan element. Update the value of the number of bytes to be flushed, given by, $f_l = F_l - E_l$.

b. Compute the contribution, O_l, of the next scan element to be flushed, to this layer. Do not count the bytes that will be deleted in Step 7.

7. From the rate control buffer, delete all the coding passes belonging to the scan element that is being flushed, but with slope values that lie between the threshold for layer *L-1* ($T_{DR}(L-1)$) and the threshold for the extended buffer ($T_{EBR}(L-1)$).

8. If $O_{(L-1)} = f_{(L-1)}$, the size of scan element being released fits the output byte requirement exactly. For each layer l, $l \in \{0,1,..., L-1\}$, update value $E_l = 0$. Go to step (11).

9. If $O_{(L-1)} \geq f_{(L-1)}$, the total number of bytes being released is greater than the allotted number and some data from the current scan element being flushed will remain in the buffer. For each layer l, $l \in \{0,1,..., L-1\}$, update value $E_l = O_l - f_l$. Go to step (11). Else go to step (10).

10. If $O_{(L-1)} < f_{(L-1)}$, releasing one scan element does not meet the output byte requirement. We need to release an additional scan element from the buffer. For each layer l, $l \in \{0,1,..., L-1\}$, compute the total contribution, O_l, from this layer. Go to step (7).

11. Release scan element (or scan elements, as required) from the buffer. Repeat steps (3) through (10) until all scan elements have been processed and buffered. No further calculation is then required. Flush out $F_{(L-1)}$ bytes at a time, until all data have been flushed.

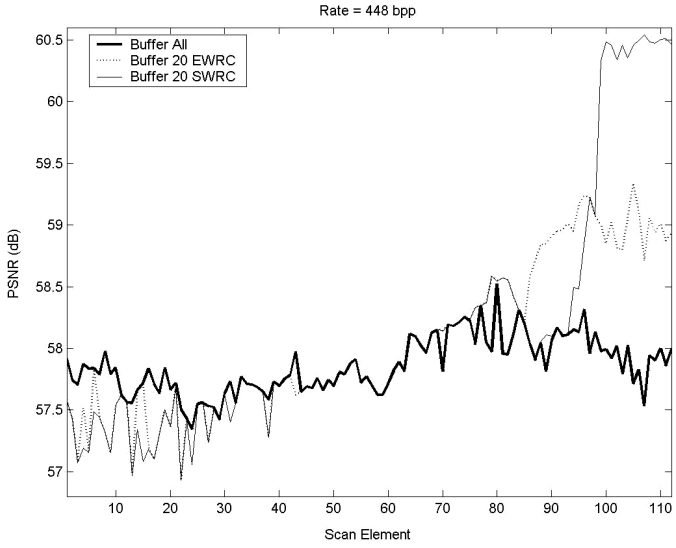

Fig. 7. Variation in PSNR across Scan Elements for Component 84 with Buffer 20 for M-EWRC versus M-SWRC.

The main advantage of M-EWRC over M-SWRC is that it reduces the variation in PSNR across scan elements with similar overall PSNR performance. By maintaining a few more coding passes than necessary, the M-EWRC algorithm is able to take bits away from easily compressible scan elements and give them to scan elements that are harder to compress. Therefore, there is a "fairer" distribution of the bit budget between scan elements compared to M-SWRC. Fig. 7 illustrates this for a particular component when a buffer size of 20 is used. In this paper, a value of *EBR=1.5* is used for all experiments employing the M-EWRC algorithm.

Fig. 8 shows the PSNR variation across scan elements for the 224 components. As expected, the standard deviation is the least when all the data are buffered. Comparing Fig. 8 to Fig. 6, we

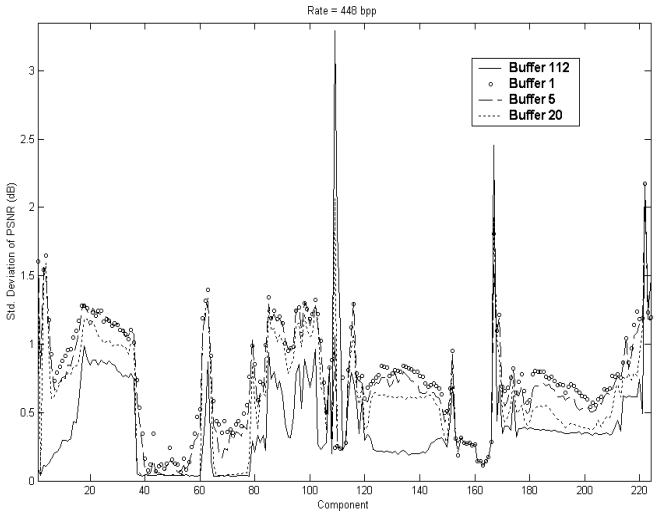

Fig. 8. Standard Deviation of PSNR by Component for M-EWRC.

see that for the same number of buffered scan elements, there is smaller variation in image quality across the image when M-EWRC is used.

Ideally, the M-EWRC algorithm reduces the standard deviation of PSNR across scan elements. In a few components though, the standard deviation for M-EWRC may actually be higher than that for M-SWRC. This occurs only when the standard deviation obtained by buffering all the data is higher than when incremental processing is performed. This happens when a component has very few features and is mostly uniform (probably due to water absorption). Such a component is very easy to compress. Since M-EWRC always tends towards the case when all the data are buffered, the standard deviation in this case tends to be higher than that for M-SWRC. The disadvantage with the M-EWRC scheme is that it requires more memory compared to the M-SWRC scheme. Thus, there is a trade-off between the quality fluctuation in the transmitted scan elements and the amount of memory needed for processing.

For the intermediate layers (corresponding to 112 bpp, 224 bpp and 336 bpp respectively), the values for M-SWRC and M-EWRC are identical. This is because these layers, by virtue of having layers above them, always operate in the M-EWRC "mode." Therefore, there is nothing to be gained by using a larger buffer under the M-EWRC scheme for these layers.

4. Conclusion

We have developed two novel rate-control schemes within the JPEG2000 framework. Our work has resulted in the development of rate control schemes for multi-component images. We have also introduced quality scalability by incorporating multiple layers in the codestream. We have tested the two multi-layer multi-component rate control algorithms on remote-sensing data. We have compared the performance of the M-SWRC and M-EWRC schemes for various buffer sizes. We observe that the overall PSNR of the components is reduced by only a small amount (less than 0.03 dB on average) when incremental processing is performed.

We also observe that the M-EWRC algorithm improves the overall PSNR slightly compared to M-SWRC, but its main feature is that it reduces the variation in PSNR across scan elements. This is a very desirable feature in maintaining the quality constant from top to bottom within an image.

These results demonstrate that our rate control schemes provide very good performance while still using modest amounts of memory. Moreover, since these algorithms work for any number of layers, they can be used in multi-cast environments to cater to differing quality requirements. These algorithms have widespread applications, especially in the remote-sensing field, where multispectral data sets (e.g., LANDSAT) are of interest.

The coding efficiency could be improved even further by removing spectral redundancy using the wavelet (or other) transform in the third dimension.

REFERENCES

1. S. Gonnelli, A. Henrichs, F. Thimler, "A data compression algorithm for the LRPT direct broadcast link on future polar orbit missions," *Proceedings of the IEEE Geoscience and Remote Sensing Symposium*, Vol. 4, pp. 2026-2028, 1999.

2. C. Lambert-Nebout, G. Moury, “A survey of on-board image compression for CNES space missions,” *Proceedings of IEEE International Geoscience and Remote Sensing Symposium*, Vol. 4, pp. 2032-2034, 1999.
3. C. Parisot, M. Antonini, M. Barlaud, “EBWIC: A low complexity and efficient rate constrained wavelet image coder,” *Proceedings of the International Conference on Image Processing*, Vol. 1, pp. 653-656, 2000.
4. T. Markas, J. Reif, “Multispectral image compression algorithms,” *Data Compression Conference*, pp. 391-400, 1993.
5. C. Lambert-Nebout, C. Latry, G. Moury, “On-board optical image compression for future high resolution remote sensing systems,” *Proceedings of SPIE International Symposium on Optical Science and Technology*, Vol. 4115, No. 39, pp. 332-346, August 2000.
6. C. Parisot, M. Antonini, M. Barlaud, C. Lambert-Nebout, C. Latry, G. Moury, “On Board Stripe-based Wavelet Image Coding for Future Space Remote Sensing Missions,” *Proceedings of IEEE International Geoscience and Remote Sensing Symposium*, Vol. 6, pp. 2651-2653, 2000.
7. T.-H. Chang, C.-Jr. Lian, H.-H. Chen, J.-Y. Chang, L.-G. Chen, “Effective hardware-oriented technique for the rate control of JPEG2000 encoding,” *Proceedings of the IEEE International Symposium on Circuits and Systems*, Vol. 2, pp. 684-687, May 2003.
8. C. Parisot, M. Antonini, M. Barlaud, “Scan-based quality control for JPEG2000 using R-D Models,” *Proceedings of XI European Signal Processing Conference*, Tolouse, France, September 2002.
9. T. J. Flohr, M. W. Marcellin, J. C. Rountree, “Scan-based processing with JPEG2000,” *Applications of Digital Image Processing XXIII, Proceedings of SPIE*, Vol. 4115, pp. 347-355, July 2000.
10. D. S. Taubman and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals, Practice and Standards*, Kluwer Academic Publishers, Massachusetts, 2002.
11. D. Taubman, “High performance scalable image compression with EBCOT,” *IEEE Transactions on Image Processing*, **Vol. 9**, pp. 1158-1170, July 2000.
12. C. Chrysafis and A. Ortega, “Line-based reduced memory wavelet image compression,” Proceedings *of IEEE Data Compression Conference*, Snowbird, UT, pp. 308-407, 1998.
13. E. Ordentlich, D. Taubman, M.J. Weinberger, G. Seroussi, and M. W. Marcellin, "Memory Efficient Scalable Line-based Image Coding," *Proceedings of Data Compression Conference*, pp. 218-227, Snowbird, Utah, March 1999.
14. J. C. Dagher, A. Bilgin, M. W. Marcellin, “Resource Constrained Rate Control for Motion JPEG2000,” *IEEE Transactions on Image Processing*, **Vol. 12**, No. 12, pp. 1522- 1529, December 2003.
15. Website: *http://aviris.jpl.nasa.gov*.