The Seamlessly Multiplexed Embedded Codec (SMEC) and Its Application in Image Coding

Jin Li
Microsoft Research, Communication, Collaboration and Signal Processing
One Microsoft Way, Bld. 113, Redmond, WA 98052.
Tel. (425) 703-8451 Email: jinl@microsoft.com

ABSTRACT

The landmark JPEG 2000 image compression standard offers not only superior compression performance, but also incredible flexibility. The compressed bitstream of JPEG 2000 can be flexibly reorganized to another bitstream of different bitrate, resolution, and spatial region of interest (ROI) or a combination of any of the above. Such flexibility is achieved by multiplexing the compressed bitstream pieces of multiple code-blocks together into a combined bitstream, with the length of the code-block bitstream piece (LOCB) embedded in the combined bitstream. The LOCB serves both to reorganize the bitstream, and to decode the bitstream. It represents a significant overhead, especially since there is no correlation between the neighbor LOCBs. In this work, we introduce seamless multiplexing, and separate the information needed for the reorganization, i.e., the LOCB, from the compressed bitstream itself by using the decoder pointer to multiplex the bitstream pieces. As a result, the compressed bitstream consists of code-block bitstream pieces seamlessly concatenated to each other. With seamless multiplexing, only the compressed bitstream (without LOCB) needs to be delivered to the receiving client. It results in better compression performance and higher granularity of access. Another benefit of seamless multiplexing is that the relative coding orders of the code-blocks are preserved in the bitstream reorganization. As a result, the seamlessly multiplexed embedded codec (SMEC) may utilize the dependencies among the code-blocks in the coding, thus further boost the compression performance.

1. INTRODUCTION

Pioneered by Shapiro [1], embedded coding has the attractive feature that a lower rate compressed bitstream is embedded in a higher rate compressed bitstream. The resultant compressed bitstream can thus be reorganized to any lower bitrate through simple truncation. Embedded coding is usually achieved by coding the transform coefficients bitplane-by-bitplane, first the most significant bitplane, then the second most significant bitplane, and all the way to the least significant bitplane. If the compressed bitstream is later truncated, at least the first several most significant bits of all coefficients have been encoded. The compressed bitstream can thus be truncated at any point with a graceful tradeoff between the distortion and the coding rate.

As a new state-of-the-art image compression standard, JPEG 2000[4] offers both good compression performance and incredible flexibility. The compressed bitstream of the JPEG 2000 can not only be scaled in different bitrate, as in prior embedded image codecs such as EZW[1], SPIHT[2] and RDE[5], but also be scaled in resolution and spatial region of interest (ROI). The scalability in multiple facets is the result of the embedded block coding with optimized truncation (EBCOT)[3]. In EBCOT/JPEG 2000, an image is first divided into multiple code-blocks, each of which consists of a rectangular block of coefficients. Every code-block is then independently encoded into an embedded code-block bitstream, which is broken down into a number of *code-block bitstream pieces*. The code-block bitstream pieces are then multiplexed together to form a combined bitstream of JPEG 2000.

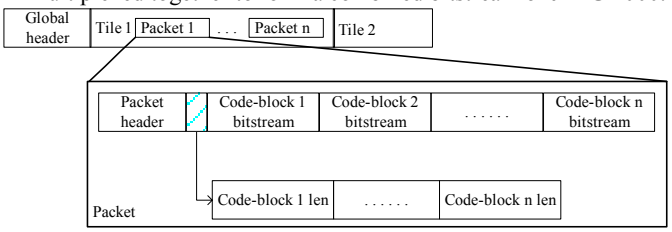

Figure 1. Syntax of the JPEG 2000 bitstream.

The syntax of the combined bitstream of EBCOT/JPEG 2000 can be shown in Figure 1. The bitstream is led by a global header, which contains information about the entire compressed bitstream, such as the image/tile size, the wavelet transform used, etc.. The body of the combined bitstream consists of a number of tiles, each of which further consists of a number of packets. A packet in turn consists of a number of code-block bitstream pieces. The bitstream piece of the code-block is the smallest access and reorganization unit in EBCOT/JPEG 2000. The EBCOT/JPEG 2000 packet is led by a packet header, and it is the length of the code-block bitstream piece (LOCB) in the packet header that enables the combined bitstream to be scaled and decoded. To scale the EBCOT/JPEG 2000 compressed bitstream by bitrate, the code-block bitstream pieces corresponding to the less significant bitplanes are dropped. To scale the compressed bitstream by resolution and/or ROI, the bitstream pieces corresponding to the code-blocks outside the desired resolution and/or ROI are dropped. Scalability in multiple facets is thus achieved. To decode the compressed bitstream, the bitstream is demultiplexed into the bitstream pieces with the aide of LOCB. After demultiplexing, the bitstream pieces of the same code-block are concatenated together to recover the bitstream of the code-block. The code-block bitstream is then fed into the code-block decoder to recover the transform coefficients. After all transform coefficients are recovered, they are inversely transformed to recover the image.

Apparently, the LOCB is an integral part of the compressed bitstream in EBCOT/JPEG 2000. It has to be delivered, even if the received bitstream is not further reorganized. It is an overhead paid for the scalability in multiple facets. The smaller the bitstream piece is, the more the overhead is in proportion to the compressed bitstream. To reduce the overhead, EBCOT/JPEG 2000 uses larger code-blocks and/or fewer layers of bitrate scalability. For example, the default code-block of JPEG 2000 is 64x64. Because the code-block is located in a wavelet subband, the corresponding spatial ROI is much larger, e.g., 2048x2048 for a 5-level wavelet transform. Though larger code-blocks and fewer bitrate layers reduce the overhead caused by the LOCB, it leads to poor granularity of access, which diminishes one of the key benefits of the scalability in multiple facets. Another shortcoming is

0-7803-8554-3/04/$20.00 ©2004 IEEE.

that the code-block has to be independently encoded and decoded. Thus, the dependencies between the cross-resolution code-blocks can not be explored, which also affects the compression performance.

Is the LOCB absolutely necessary to demultiplex the combined bitstream? Our answer is NO. The traditional embedded coder, such as EZW[1], SPIHT[2] or RDE[5], do not embed any bitstream length information in the compressed bitstream. In this work, we show that in multiple facet embedded coding, the information needed to reorganize the compressed bitstream, e.g., the LOCB, can be separated from the compressed bitstream itself. We call the technology as the seamlessly multiplexed embedded codec (SMEC), because the compressed bitstream now consists of code-block bitstream pieces seamlessly concatenated to each other, with no LOCB. In contrast to EBCOT/JPEG 2000 bitstream, which can not be decoded without the LOCB, SMEC combined bitstream can be decoded by demultiplexing on the fly, during the code-block entropy decoding stage. A separate companion file, which contains LOCB, is used to reorganize the SMEC compressed bitstream. Because the LOCB does not need to be sent to the receiving client, SMEC may allow much smaller granularity of access, and use much smaller bitstream pieces. By eliminating the LOCB from the compressed bitstream, SMEC boosts the effective compression performance of the embedded codec. Another benefit is that the relative coding orders of the code-block bitstream pieces are preserved in the SMEC reorganization. Thus, SMEC is able to utilize the dependencies among the code-blocks, which may further boost the compression performance.

The rest of the paper is organized as follows. The SMEC bitstream syntax is examined in Section 2. The principle of decoding the SMEC multiplexed bitstream is explained in Section 3. We explain the operation flow of the SMEC encoder and decoder in Section 4. The experimental results are shown in Section 5.

2. SEAMLESS MULTIPLEXING: BITSTREAM SYNTAX AND COMPANION FILE

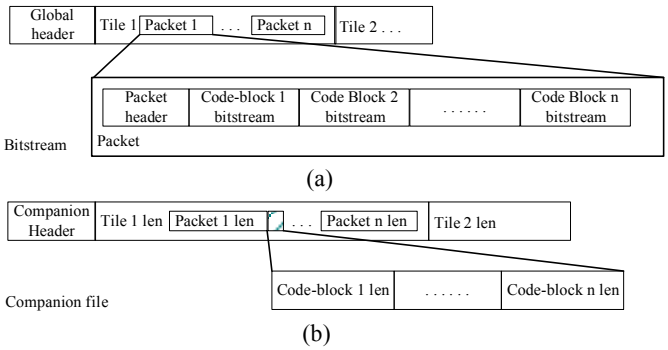

Figure 2 Syntax of the SMEC (a) bitstream, (b) companion file.

A major difference between the EBCOT/JPEG 2000 and the seamlessly multiplexed embedded codec (SMEC) is that the latter isolates the information needed for the bitstream reorganization, which includes the LOCB and the length of other components, and gathers them into a *companion file.* The syntax of the SMEC bitstream and the companion file can be shown in Figure 2. Compared with Figure 1, the difference is evident: all information needed to reorganize the compressed bitstream, i.e., the length of the tile, the packet and the LOCB, is moved to the companion file in SMEC. The companion file is still needed to reorganize the compressed bitstream, such as to scale the compressed image bitstream into a bitstream of desired bitrate, resolution, and ROI. However, it is not used in the decoding process.

At first sight, putting the reorganization information in the companion file does not seem to be much of a difference, as the encoder still generates about the same amount of information, and the parser still needs both the bitstream and the companion file. However, we point out the follows.

First, the SMEC compressed bitstream itself is a bitrate scalable (i.e. embedded) bitstream, scaling by bitrate can be achieved by simply truncating the SMEC bitstream without the use of the companion file.

Second, since the SMEC bitstream is decodable, the companion file can be recovered from the compressed bitstream. The only role of the companion file is to facilitate bitstream reorganization in multiple facets. Moreover, the main applications of scaling by multiple facets are the Internet server-client applications, such as the Vmedia image browser[6], where the server may tailor the embedded compressed bitstream according to the network condition and the capacity of the client device, and then deliver the reorganized bitstream of desired bitrate, resolution, and ROI to the client device. In such applications, the master bitstream and the companion file are stored on the server, whose storage capacity is large. The information delivered to the client device is only the reorganized compressed bitstream, which in SMEC contains no information for further multi-aspect reorganization, such as the LOCB, etc.. As a result, the network bandwidth is more efficiently utilized, results in an actual improvement of the compression performance.

Third, in SMEC, the companion file itself can be tailored to the desired level of reorganization, without changing the compressed bitstream. Similar to EBCOT/JPEG 2000, the SMEC bitstream may be organized with multiple levels of hierarchies, from the code-block level to the packet level and further up to the tile level. The LOCB provides the possibility to access and reorganize the bitstream on a code-block basis, whereas the length of the packet and the length of the tile provide the access of the bitstream on a packet and tile basis, respectively. If certain level of reorganization is not desired, the corresponding length information may be removed from the companion file. For example, if the code-block access is not required, we may remove the LOCBs in the companion file of Figure 2(b). The resultant companion file can still be used to reorganize the master bitstream on a packet and/or tile basis.

3 SEAMLESS MULTIPLEXING PRINCIPLE: DEMULTIPLEXING ON THE FLY

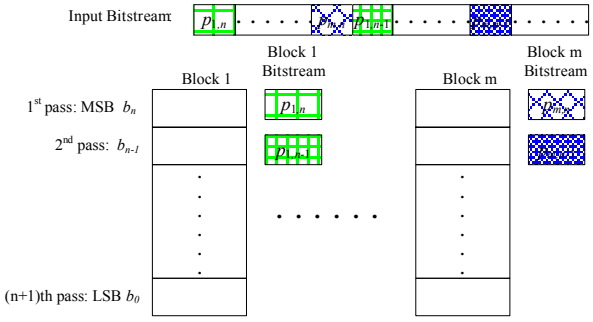

Figure 3 Decoding of the seamlessly multiplexed bitstream.

EBCOT/JPEG 2000 cannot separate the LOCB from the combined bitstream because the LOCB is essential in the decoding

operation. Without the LOCB, the code-block bitstream cannot be extracted and decoded. How is this solved in SMEC? In the following, we show that by using the decoder pointer to separate the bitstream pieces of the code-block, SMEC can decode the combined bitstream without first demultiplexing it.

We examine a simplified SMEC decoder, for which the combined bitstream is formed by concatenating the code-block bitstream pieces, with one bitstream piece corresponds to one bitplane of embedded coding. The combined bitstream is ordered first by bitplane, then by code-block index. The decoding of the combined bitstream can be shown in Figure 3. It is very similar to that of the embedded image decoder operating on the entire image. The coefficients, which are grouped into m code-blocks, are decoded bitplane-by-bitplane, from the most significant bitplane to the least significant bitplane. In the beginning of decoding, m code-block decoders are initialized in parallel. We then feed the combined input bitstream to the decoder of the code-block 1, as if the entire bitstream is destined for the code-block 1. The decoder decodes the code-block 1 to the end of the most significant bitplane. During the process, a certain portion of the input bitstream is consumed. Let the *decoder pointer* of the code-block 1 mark the position of the input bitstream that is to be read next. The decoder pointer separates the input bitstream into two parts: the first part has already been read into the code-block 1 entropy decoder, the second part is what is left. We now redirect the remaining of the input bitstream to the decoder of the code-block 2, which again decodes the code-block to the end of the most significant bitplane. The bitstream is split again by the decoder pointer, with the remaining part fed to the code-block 3. The process repeats code-block by code-block, and when the most significant bitplane of all code-blocks has been decoded, it moves to the second more significant bitplane. This repeats until the entire combined bitstream has been fed to the entropy decoders of the code-blocks. By demultiplexing during the code-block decoding operation, SMEC avoids the operation of explicitly demultiplexing the combined bitstream. It is thus able to decode the combined bitstream without the aide of LOCB.

4 SEAMLESS MULTIPLEXING: ENCODING AND DECODING OPERATION FLOW

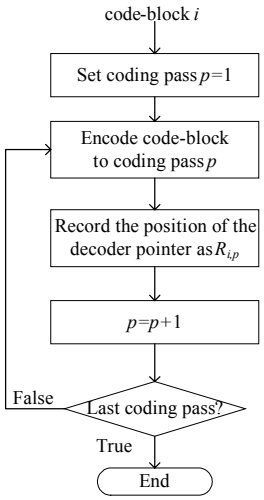

Figure 4 Operation flow of the SMEC code-block encoder.

The actual operation of SMEC is more complex than the simplified framework described above. First, rather than operating on a bitplane basis, the code-block embedded entropy coder may operate on a sub-bitplane basis, as in EBCOT[3] or RDE[5], with each coding pass just encoding the bits of a partial set of coefficients. The rational is that using the already coded information, it is possible to classify the coefficients into sets of various rate-distortion (R-D) characteristics. By encoding the sets of bits according to the descend order of the R-D slope [5], we may further improve the R-D performance of the code-block coding. SMEC follows EBCOT/JPEG 2000 in coding each bitplane of coefficients with three passes (sub-bitplanes): first the insignificant coefficients with at least one significant neighbor, then the refinement coefficients, and finally the insignificant coefficients with no significant neighbors.

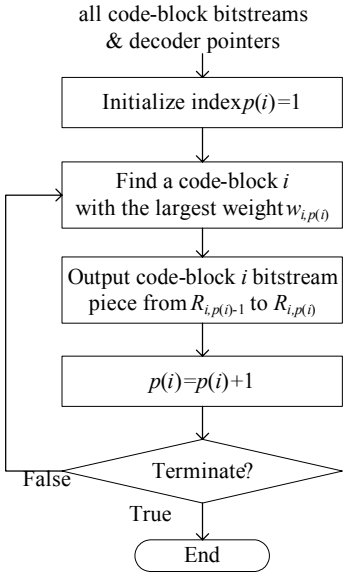

Figure 5 Operation flows of the SMEC multiplexer.

In general, the SMEC code-block entropy coder is a multi-pass embedded coder. The operation flow of the SMEC code-block encoder can be shown in Figure 4. It is very similar to an EBCOT/JPEG 2000 code-block encoder. The only extra operation is that at the end of coding pass p of the code-block i, the position of the decoder pointer is recorded as $R_{i,p}$. The decoder pointer is recorded in the SMEC companion file in the form of LOCB, and is used by the multiplexer of the encoder to form the seamlessly multiplexed bitstream.

The second issue in implementing an advanced SMEC codec is that the different code-blocks may not have the same number of coding passes. Moreover, the coding passes of different code-blocks may not be on the same R-D curve, or contribute the same to the psycho-visual distortion. For the bitstream piece of the code-block i at pass p, we introduce the weight of the bitstream piece as $w_{i,p}$, which is defined as the expected reduction of the psycho-visual distortion divided by the expected coding rate, i.e., the expected R-D slope of the bitstream piece.

Instead of the actual R-D slope calculated from the bitstream piece that is used in the R-D optimization of EBCOT/JPEG 2000, SMEC uses the expected R-D slope, as in RDE [5]. In the current SMEC implementation, the weight parameter $w_{i,p}$ is calculated as:

$$w_{i,p} = v_i \cdot 4^{B_i} \cdot a(p), \qquad (1)$$

where v_i is the visual weight of the subband, B_i is the number of bitplanes of the code-block i, and $a(p)$ is the calibrated weight of the coding passes, which is a constant look-up table with pre-calculated values of:

$$a(p) = \begin{cases} 0.75, & p = 1, \\ 0.25, & p = 2, \\ 0.24, & p = 3, \\ a(p-3)/4, & p > 3, \end{cases} \quad (2)$$

The bitstream pieces are then concatenated together in the descending order of their weight by the SMEC multiplexer. The operation flow of the multiplexer can be shown in Figure 5. The multiplexer continues until the satisfactory of a terminate condition, which can be either a desired bit rate, a desired visual distortion level, or the lossless condition that all code-block bitstream pieces have been outputted.

The SMEC parser reorganizes the compressed bitstream into another compressed bitstream of desired bitrate, resolution and ROI. Using the LOCB recorded in the companion file, the SMEC parser first splits the master bitstream into bitstream pieces, and then reassembles them through a multiplexer similar to the encoder multiplexer of Figure 5. The multiplexer of the parser controls the bitrate of the output application bitstream by terminating when the desired bitrate is reached. It reorganizes the application bitstream in resolution and ROI by dropping the bitstream pieces of undesired code-blocks. Because the parser just splits and reassembles the bitstream pieces, and does not perform entropy coding or transform operation, it can perform the bitstream reorganization very fast. In the SMEC parser, the bitstream pieces can be dropped, however, their relative orders never change because the weight parameter of equation (1) that controls the order of multiplexing is a fixed value for a particular code-block and coding pass. Therefore, the dependencies between the code-blocks can be used in the encoding and decoding. As a result, we may use the inter-scale dependency of the wavelet, and let a code-block be encoded with reference to another code-block in the same spatial location, but at a coarser resolution level. This further improves the compression performance of SMEC.

5. EXPERIMENTAL RESULTS

To evaluate the merit of the seamless multiplexing, we build a SMEC image coder. We compare the compression performance of SMEC with that of the JPEG 2000 VM. The test data set is the set of images used in the JPEG 2000 standard. The test bitrates are 0.125, 0.25, 0.5, 1.0 and 2.0 bit per pixel (bpp). Both SMEC and JPEG 2000 decompose the image with a 5-level bi-orthogonal 9-7 wavelet filter. A code-block size of 8x8 is used in both SMEC and JPEG 2000. Thus both SMEC and JPEG 2000 compressed bitstream can be scaled in term of bitrate, resolutions (6 level of scaling) and spatial ROI (with ROI being 256x256). Both SMEC and JPEG 2000 encode the image at the top bitrate of 2.0bpp. The bitstreams of the other bitrate are obtained by truncating the top rate bitstream. The PSNR of the decoded images for SMEC and JPEG 2000 are tabulated in Table 1 and 2, respectively.

It is observed that with the same granularity of access, SMEC with the seamless multiplexing outperforms JPEG 2000 by an average of 0.8dB. By reducing the overhead of multiplexing and utilizing cross-resolution dependencies, the seamless multiplexing is an effective technology to improve the compression performance of the embedded codecs with multiple facet scalability, and to reduce the granularity of access of such codecs.

Table 1 PSNR performance (dB) of the SMEC codec.

Bitrate	2.0bpp	1.0	0.5	0.25	0.125
aerial2	37.89	33.17	30.52	28.46	26.41
bike	43.24	37.68	33.19	29.32	25.77
cafe	38.67	31.87	26.70	22.87	20.51
cats	52.77	43.91	37.23	32.72	29.94
cmpnd1	50.27	41.29	32.30	26.40	21.67
cmpnd2	49.82	43.04	35.73	30.42	26.30
finger	37.09	31.20	27.74	24.36	21.89
gold	42.49	37.33	34.00	31.50	29.49
hotel	42.66	38.12	33.67	29.92	27.19
mat	49.77	44.78	40.27	36.59	33.57
seismic	50.40	46.66	43.70	40.70	36.73
target	51.35	43.13	33.84	26.87	22.61
tools	37.92	32.21	27.47	23.63	21.30
txtur1	30.42	25.05	22.04	20.33	18.89
txtur2	37.15	31.50	28.59	26.64	24.94
us	48.33	39.86	34.02	29.30	25.29
woman	43.33	38.08	33.34	29.57	27.17
Average	43.74	37.58	32.61	28.80	25.86

Table 2 PSNR performance (dB) of JPEG 2000.

Bitrate	2.0bpp	1.0	0.5	0.25	0.125
aerial2	36.54	32.44	30.02	27.96	25.96
bike	42.34	36.68	32.16	28.45	25.42
cafe	37.22	30.30	25.47	22.30	20.16
cats	51.89	42.10	35.96	31.91	29.45
cmpnd1	51.36	38.86	30.30	24.43	20.91
cmpnd2	50.12	41.24	34.04	29.10	25.64
finger	35.70	30.61	26.91	23.71	21.51
gold	41.39	36.64	33.46	30.95	28.91
hotel	41.91	37.20	32.86	29.35	26.37
mat	50.05	44.30	39.70	36.05	32.97
seismic	49.29	46.74	43.58	39.57	35.58
target	54.68	40.77	32.10	26.16	22.07
tools	36.79	31.13	26.49	23.17	20.88
txtur1	28.84	24.29	21.61	19.94	18.74
txtur2	35.53	30.80	28.01	26.17	24.83
us	47.58	38.73	32.67	28.17	24.22
woman	42.54	37.12	32.49	29.15	26.82
Average	43.16	36.47	31.64	28.03	25.32

6. REFERENCES

[1] J. Shapiro, "Embedded image coding using zerotree of wavelet coefficients", *IEEE Trans. On Signal Processing*, vol. 41, pp.3445-3462, Dec. 1993.

[2] A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE. Trans. Circ. Syst. Video Tech*, vol. 6, pp. 243-250, June 1996

[3] D. Taubman, "High performance scalable image compression with EBCOT", *IEEE Trans. On Image Processing*, Vol. 9, No. 7, pp. 1158-1170, Jul. 2000.

[4] M. W. Marcellin and D. S. Taubman, *Jpeg2000: Image compression fundamentals, standards, and practice*, Kluwer Academic Publishers.

[5] J. Li and S. Lei, "An embedded still image coder with rate-distortion optimization", *IEEE Trans. On Image Processing*, Vol. 8, No. 7, pp. 913-924, Jul. 1999.

[6] J. Li and H. Sun, "A virtual media (Vmedia) interactive image browser", Dec. 2003, *IEEE Trans. On Multimedia*.