The Predictive Embedded Zerotree Wavelet (PEZW) Coder: Low Complexity Image Coding with Versatile Functionality

Jie Liang

Media Technology Lab, DSPS R&D Center, Texas Instruments, MS 8374
8330 LBJ Freeway, Dallas, TX 75243
email: liang@ti.com

ABSTRACT

In this paper, we introduced the Predictive Embedded Zerotree Wavelet (PEZW) codec, an image coder that achieves good coding efficiency and versatile functionality with limited complexity requirement. Our complexity analysis showed that the memory requirement of this coder is less than 15k bytes regardless of image sizes. Our simulation results also showed that the coding efficiency of this low complexity coder is competitive with the state of the art of wavelet coders that use whole image buffers. The PEZW coder described in this paper has been adopted in MPEG4 as its still texture coding tool and is currently a proposal to the evolving JPEG2000 standard.

1. INTRODUCTION

Image compression is becoming ubiquitous in many application areas as diverse as web browsing, multimedia database, digital still cameras, printers, and scanners. They are supported on many different hardware as well as software platforms such as PCs, Unix Workstations, embedded systems such as DSPs, and micro-controllers. All these different applications and appliances impose different constraints and requirements for the image coding algorithm. On the other hand, it is highly preferred that one single image coding algorithm can meet the requirements of the majority of these applications so that interoperability can be guaranteed. These conflicting requirements provide a big challenge for image coding research.

In the last a few years, a lot of progresses have been made in image coding algorithms, especially with the introduction of wavelet based methods [1,2,3,4]. However, the majority of the attention has been paid to improving coding efficiency of the image compression algorithms. While coding efficiency is an very important factor in judging an image coding algorithm, even more important for practical applications, is to evaluate the balance between coding efficiency, implementation complexity, and features such as spatial and quality scalability, progressive transmission, random accesses, and error resilience. This paper is an attempt in that direction.

In this paper, we describe the Predictive Embedded Zerotree (PEZW) wavelet coder that achieves many of the above mentioned objectives, namely good coding efficiency, low complexity requirement, and versatile functionality. This coder has been adopted in the ISO MPEG4 standard as part of the still texture coding tools. It is also currently a proposal from Texas Instruments to the ISO JPEG2000 committee, and has scored well in its evaluation tests. In PEZW, the wavelet coefficients are encoded tree by tree, while the coefficients that are located at different decomposition levels are encoded separately using different context models. Combined with a low complexity, localized wavelet transform such as the block based wavelet transform [5] or the line based wavelet transform [6], the PEZW algorithm eliminates the need for large image buffers. As a matter of fact, the memory requirement of the PEZW coder is independent of the image sizes, a very useful feature for system designs. In the meantime, we maintain the capability to have byte-accurate rate control, scalability, random accesses, etc. Due to its flexibility for inserting resynchronization marker and re-initialization at the boundary of each tree block, the error resilience ability of the PEZW coder is also significantly enhanced. Even though the complexity of the PEZW is much reduced and more features are supported, the coding efficiency of the PEZW method is as good as the state of the art wavelet coders that use whole image buffers such as the wavelet trellis coded quantization (WTCQ) algorithm and is much improved compared to Shapiro's zerotree method.

2. THE PREDICTIVE EMBEDDED ZEROTREE (PEZW) ALGORITHM

2.1. Overview

The Predictive Embedded Zerotree (PEZW) coder is a zerotree based codec. Zerotrees are the basic coding units for PEZW. Since all the coefficients in one zerotree correspond to the same spatial location, it makes it possible that we can do the transform and entropy coding all locally, just like how JPEG baseline can be implemented with efficient hardware implementations. Figure 1. describes the basic data flow of the coding algorithm. We can see that the PEZW can be efficiently implemented as a block based coder, with the advantages that are associated with block based coder such as low memory storage requirement and local memory access. What is interesting is that we can maintain the coding efficiency and scalability features that are usually associated with global wavelet coders that use whole image buffers.

For an input image block, we first classify it into natural image blocks or synthetic/bi-level image blocks. For natural image blocks, a block based localized wavelet transform was

performed. The wavelet coefficients produced by the block wavelet transform is exactly the same as the coefficients obtained from the wavelet transforms using a whole image buffer [5]. Then all the wavelet coefficients are quantized with scalar quantization with a dead-zone around zero. Then, the quantized coefficients within the block are grouped into zerotrees, and each tree was entropy encoded with PEZW entropy coder from the MSB bitplane to the LSB bitplane. The DC band was encoded separately from the AC band coefficients, and are coded with DPCM coding. For image blocks that are classified as bi-level or synthetic image blocks, we bypass the wavelet transform and code the image blocks with image domain coding methods, such as JBIG and JPEG-LS.

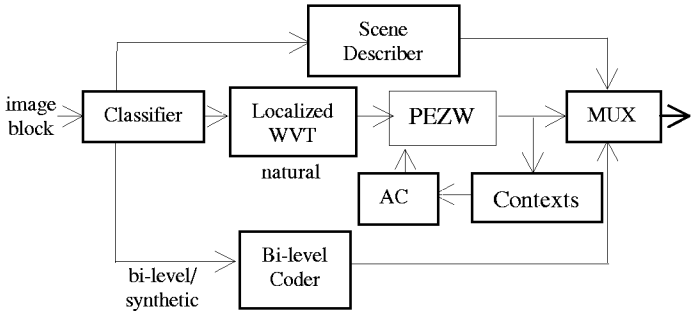

Figure 1. The EPZW Codec

The PEZW entropy coder introduced several modifications that significantly improves the original zerotree coder. The major improvements are as follows:

- Introduction of new zerotree symbols (such as ZTRV: non-zero-valued zerotree root) for forming zerotrees.
- Use of adaptive context models for encoding the zerotree symbols.
- The zerotrees are encoded depth first and all bitplanes of one zerotree are encoded before moving to the next zerotree. This significantly reduces the complexity requirement of the zerotree coder.

2.1. Forming Zerotrees

Zerotrees have been shown to be efficient in representing a large portion of zero coefficients with one zerotree root symbol, which property works well for wavelet transforms where most of the coefficients are zero after quantization. The zerotree symbols in PEZW are slightly different from the zerotree symbols used in Shapiro's EZW method. The zerotree nodes are represented by the following symbols:

- ZTRZ (zero-valued zerotree root): the wavelet coefficient itself is zero for a given threshold, and all its descendants are zero.
- ZTRV (non-zero-valued zerotree root): the wavelet coefficient itself is significant, but all its descendants are zero.
- IZER (Isolated Zero): the wavelet coefficient itself is zero, but not all its descendants are zero.
- IVAL (Isolated Significant): the wavelet coefficient itself is significant, and some of its descendants are also significant.

2.2. Context Models for Encoding Zerotree symbols

The zerotree symbols are encoded with context-based adaptive arithmetic coding. To reduce complexity, we did not use high order contexts. Instead, when encoding the zerotree symbol of a certain coefficient, we use the zerotree symbol of the same coefficient in the ***previous*** bitplane as the context for the arithmetic coding. For each coefficient, the total maxim number of context models is five, and only one memory access is needed to form the context. By using these simple contexts, we significantly reduced the memory requirement for storing the context models and the number of memory accesses needed to form the contexts. Our experience is that by using previous zerotree symbols as context, we are able to capture the majority of the redundancy and not much can be gained by adding more contexts.

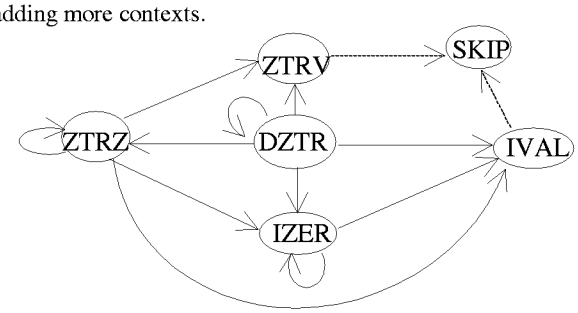

Figure 2. State transition map for contexts

If we consider the entropy coder as a state machine, PEZW uses a order-1 Markov model for encoding the zerotree symbols. Figure 3 describes the state transition of the zerotree symbols. Solid line describes the transition that emit bits to the decoder, and dashed line describes the transition that can be derived from other contexts and do not emit any bits to the decoder.

In addition to the zerotree symbols listed in section 2.1., we have two addition states in Figure 2. The state DZTR means that the coefficient is a descendant of ZTRZ or ZTRV in a previous bitplane zerotree scan. The symbol SKIP is a sink node, which means that the encoder/decoder will skip this coefficient from now on (because they are already significant, and no additional zerotree symbol need to be sent. Only refinement bits need to be sent). Table 1 shows the conditional statistics for each context. we can see that different previous state generates significantly different statistics, which means that these contexts are effective for entropy coding.

Previous	Probability (%)			
Symbols	IZER	IVAL	ZTRV	ZTRZ

IZER	54	46	0	0
ZTRZ	15	11	52	22
DZTR	13	4	60	23

Table 1 Conditional Probability for Different Contexts

Each scale/decomposition level and each bitplane have their context models. The arithmetic coder is initialized at the beginning of each bitplane and subband. The initialization eliminates dependencies of the context models and arithmetic coders across scales and bitplanes, a very important property for good error resilience performance. In addition, initialization can be done at any locations that are at the boundary of a zerotree and resynchronization marker can be inserted, so that additional protection can be injected for a selected area of images.

2.3. Tree-Based Localized Coding

Most wavelet based image coding methods encode the wavelet coefficients with global information such as global context models, rate allocation, etc. Our goal here is to demonstrate that it is possible to achieve good coding efficiency, versatile features and low complexity within one coding algorithm with only local information. Figure 4 shows the scanning order of the PEZW coder using the example of a 8x8 image block with 2 levels of decompositions.

1	2	3	4	7	8	11	12
23	44	5	6	9	10	13	14
24	25	45	46	15	16	19	20
26	27	47	48	17	18	21	22
28	29	32	33	49	50	53	54
30	31	34	35	51	52	55	56
36	37	40	41	57	58	61	62
38	39	42	43	59	60	63	64

Figure 4. Scanning order of PEZW coder

You can see that the coefficients are encoded tree-depth first. There are totally 3 zerotrees in this 8x8 block, with roots at coefficients labeled 2, 23, and 44. They are at the top level of the decomposition (coarsest scale). Let us call the tree rooted at #2 coefficient tree #0, the tree rooted at #23 coefficient tree #1, and the tree rooted at #44 coefficient tree #2. The encoding procedure goes as follows:

step 1. Read in all coefficients belonging to tree #m. (m was initialized to be zero, and T was initialized as a value larger than all possible wavelet coefficients).

step 2. For a given threshold T, label all the coefficients as ZTRZ, ZTRV, IZER, IVAL, and DZTR as defined in section 2.1. This is the bitplane *snrl* (initialized to be zero).

step 3. Let *spl* be the scale of the coefficient and encode the zerotree symbols (ZTRZ, ZTRV, IZER, IVAL) with context based arithmetic coding, using the context [*spl, snrl*], and output the bitstream to the bitstream buffer [*spl, snrl*].

step 4. Update the contexts with the current zerotree symbols. For coefficient with IVAL and ZTRV symbol, use SKIP as their current context.

step 5. Go to step 3, until all coefficients in tree #m are done.

step 6. T=T/2, *snrl*++, go to step 2 (encode the next bitplane).

step 7. Done with the current tree, m++, if not the last tree, go to step 1 (encode the next zerotree).

At any time, we are looking at only one zerotree, which are related to the same spatial locations. This property facilitates implementation of the image coder with very low memory requirement.

3. FEATURES

Features of an image coder are very important for practical applications. For instance spatial and quality scalability enables efficient memory management for digital still camera products. Progressive transmission effectively reduces the bandwidth requirement of web browsing and enables multicasting to many clients with different connection speeds. Random access and the ability to focus on selected areas with increased resolution are very useful for telemedicine and interactive media services. The PEZW coder provides support for most of the desirable features as list below:

- **spatial scalability:** the decoder can decode the bitstream at a chosen resolution.
- **quality scalability:** the decoder can decode the image at a certain bitrate or up to a certain bitplane. Actually, the bitstream supports scalability in both the spatial and SNR direction simultaneously.
- **progressive transmission:** the bitstream can be arranged so that the bitstream can be decoded progressively.
- **random access**: the PEZW encode the bitstream using tree as basic units. For a 4 level decomposition, the minimal block size that a wavelet tree corresponds to is 16x16. By inserting a start code at the beginning of a tree block, we can provide random access point to that block. The granularity of the random access is 16x16 blocks.
- **error resilience:** the bitstream is separated into segments by bitplanes and spatial levels, with each segment started with a resynchronization marker. In addition, start code can be inserted before any tree blocks, providing additional protection for selected areas.

4. COMPLEXITY ANALYSIS

Complexity requirement of an algorithm is one of the most important considerations in practical applications. In general, complexity can be categorized as memory requirement, speed (which is directly related to memory access since memory access is the bottleneck for most systems), and power

consumption. Power consumption is hard to quantify, but should be highly correlated to memory access and running time. In the following, we analyze the memory requirement and memory access of the PEZW codec.

4.1 Memory Size Requirement

In the tree-by-tree encoding/decoding fashion, each time we are dealing with a block of coefficients that are organized into a tree structure. The following describes the memory allocated in the encoder/decoder function in our current implementation. We assume a decomposition level of 4. The memory allocated are as follows:

- wavelet tree buffer: 341 bytes (not needed at decoder)
- model buffer: #decomposition level X #bitplanes X #number of models X #bytes per model = 4x10x9x2 = 720 bytes
- bitstream buffer pointers: #decomposition level x #bitplanes x 4 bytes = 4x10x4 = 160 bytes
- other programming buffer: 1193 bytes

The total is 3606 bytes. If we consider the memory requirement of the total system, using a localized block based overlapped wavelet transform as in [5] and a block size of about 64x64, with 7x9 bi-orthogonal wavelets, the total system requirement will be less than 15k bytes excluding bitstream buffer, regardless of the input image size.

4.2 Memory Accesses

Another important question is memory access. Due to big difference in access time of different memory types (on-chip/cache memory access takes one cycle, and off-chip low-cost DRAM takes a lot longer), it is important to distinguish between the different memory access types.

(1). off-chip memory access:
Since the PEZW codec requires a small size memory, which can be fit on-chip or in-cache, and only needs local information, off-chip or off-cache memory accesses are greatly reduced. If combined with a local wavelet transform, we only need to access off-chip memory for an image block (one read per pixel), and write to off-chip memory if the compressed bitstream can not be fit on-chip (this is in compressed domain, which in most case averages 0.1 write per pixel, assuming 10:1 compression). To make things better, since the off-chip memory access timing and size can be well defined, DMA or burst mode for DRAM access can be used to minimize read/write time.

(2). on-chip memory access:
On-chip memory access takes zero wait state. At the encoder, one read per pixel is needed to form the tree structure. After that, we access the pixels from tree-root down, therefore, most of the time, we only access a small portion of the wavelet coefficient. At the decoder, we always decodes from top down. Therefore, the speed of the decoder is a lot faster. The number of memory access is also dependent on how many bitplanes are coded. In general, we feel that due to the hierarchical structure of the zerotree, the number of memory access is greatly reduced.

5. CODING RESULTS

In this section, we show the coding results of the PEZW coder with comparisons to the wavelet trellis coded quantization (WTCQ) method. The WTCQ method represents the state of the art in image coding, and is currently the baseline entropy coder in the JPEG2000 verification model (VM) version 1.0. The image used is the *woman* image from the JPEG testing image suite. This image is 2048x2560 in size and monochrome. We used Daubechies 7x9 filters and 5 levels of decomposition. The results showed that PEZW outperformed WTCQ at all bitrates. In general, the coding efficiency of PEZW and WTCQ are similar, but PEZW has much lower complexity.

JPEG2000 VM0		PEZW			
bitrate	psnr	bitrate	diff	psnr	diff
1.9179	43.64	2.0444	+0.126	43.88	+0.24
1.0007	38.16	1.0151	+0.014	38.43	+0.27
0.4906	33.17	0.4973	+0.006	33.39	+0.22
0.2494	29.51	0.2469	+0.002	29.76	+0.25
0.1237	26.81	0.1257	+0.002	27.23	+0.42
0.0624	25.05	0.0629	+0.001	25.34	+0.29

Table 2. Coding results for *woman* image

6. REFERENCE

[1] Jerome M. Shapiro, "*Embedded Image Coding Using Zerotrees of Wavelet Coefficients*", IEEE Trans. Signal Processing, December 1993.

[2] P. Sriram and M. W. Marcellin, *"Image Coding Using Wavelet Transforms and Entropy-Constrained Trellis Coded Quantization*", IEEE Trans. Image Processing, June 1995.

[3] A. Said and W. A. Pearlman, *"A New, Fast, and Efficient Image Coder Based on Set Partitioning in Hierarchical Trees*", IEEE Trans. Circuit and System Video Technology, June 1996.

[4] Jie Liang, "*Highly Scalable Image Coding for Multimedia Applications*", ACM Multimedia Conference, October 1997, Seattle, WA.

[5] Motorola Australia Research Center, "*Embedded independent block-based coding of subband data*," ISO/IEC JTC/SC29/WG1 Document, June 1998.

[6] C. Chrysafis and A. Ortega, *"Line Based Reduced Memory Wavelet Image Compression.*," in Proc. IEEE Data Compression Conference, pp. 308-407, Snowbird, UT, 1998.