IAENG International Journal of Computer Science, 36:2, IJCS_36_2_09

An Optimum Approach for Image Compression: Tuned Degree-K Zerotree Wavelet Coding

Li Wern Chew*, Wai Chong Chia, Li-minn Ang and Kah Phooi Seng

Abstract - **This paper presents an image compression technique called the Tuned Degree-K Zerotree Wavelet (TDKZW) coding which is targeted at single-encoding, multiple-decoding image processing applications. In the proposed work, the degree of zerotree tested is tuned in each encoding pass to achieve an optimal compression performance. The TDKZW coder uses the set-partitioning approach which provides embedded coding and makes progressive transmission possible. Simulation results which were performed for both lossless and lossy compression show that the proposed TDKZW coder gives a better compression performance than the SPIHT coder.**

Index Terms - **Degree-k zerotree coding, embedded coding, image compression, set-partitioning coding.**

I. INTRODUCTION

Image compression has become increasingly important with the continuous development of the Internet, remote sensing, medical imaging, forensic analysis and satellite communication techniques. Due to the high cost of providing a large transmission bandwidth and a huge amount of storage space, many fast and efficient image compression engines have been introduced.

Among the many types of image compression techniques, the zerotree-based image coding scheme is generally preferred due to its fast and efficient image processing capability, high compression quality as well as its simplicity in coding procedures [1, 2]. In zerotree-based image compression, wavelet transformation is first applied on an image to decompose the image into multi-resolution wavelet subbands of different frequency content. Zerotree coding techniques such as the embedded zerotree wavelet (EZW) [1] and the set-partitioning in hierarchical trees (SPIHT) [2] are then used to encode the wavelet coefficients to achieve compression.

In this paper, a zerotree-based image compression algorithm called the Tuned Degree-K Zerotree Wavelet (TDKZW) coding is presented. In the proposed coding scheme, the degree of zerotree tested is tuned in each encoding pass to achieve optimal compression performance. To obtain a suitable tuning table, parallel coding of degree-1 to degree-k zerotree coding are first performed in the pre-processing stage. The degree-k zerotree coding that gives the best coding performance at each encoding pass is then selected for the TDKZW coding, i.e. the degree-k coding scheme which encodes the image with the fewest bits at each encoding plane is selected as the tuning parameter for the proposed TDKZW coder.

A major advantage of the TDKZW coder is that the compressed images are stored in a lossless form with optimal compression efficiency. Besides this, the proposed work which uses the set-partitioning approach provides embedded coding and enables progressive transmission to take place. Since it is applicable to both lossless and lossy image compression, the proposed TDKZW coding is mainly targeted at single-encoding, multiple-decoding image processing applications. These include online image databases for medical and satellite imaging as well as online photo hosting applications.

The remaining sections of this paper are organized as follows: An overview of the set-partitioning coding with degree-k zerotree is first presented in Section II. This includes the EZW and SPIHT coding schemes. Section III presents the proposed TDKZW coding scheme. In the proposed work, a pre-processing stage is first carried out to obtain a suitable tuning table followed by the TDKZW coding. Next, the performance of TDKZW coder for both lossless and lossy compression is evaluated and discussed in Section IV. Section V concludes this paper.

II. SET-PARTITIONING CODING WITH DEGREE-K ZEROTREE

Wavelet-based image coding techniques such as EZW and SPIHT are significant breakthroughs in still-image compression. These two coding schemes apply the concept of zerotree coding where the relationship of wavelet coefficients across the different frequency subbands at the same spatial location is exploited and encoded.

In zerotree-based coding, when a wavelet coefficient at a higher level is found to be insignificant with respect to a given threshold T, then all the wavelet coefficients of the same orientation in the same spatial location at the lower levels are likely to be insignificant with respect to T [1]. These insignificant wavelet coefficients which form a spatial orientation tree (SOT) are referred to as a zerotree and are encoded using a single symbol to achieve compression.

In general, a degree-k zerotree is a SOT where all its nodes are insignificant with respect to a threshold value except for the nodes in the top k levels and a degree-k zerotree coder performs significance tests on a degree-1 zerotree up to a degree-k zerotree for a wavelet coefficient (i, j) [3, 4].

The EZW coding scheme encodes the wavelet coefficients starting with a degree-0 zerotree, which is a SOT where its root node and all its descendants are insignificant

---

Manuscript received May 08, 2009. Manuscript submitted to the IAENG special issues of IMECS 2009.

The authors are with the School of Electrical and Electronics Engineering, The University of Nottingham, 43500 Semenyih, Selangor, Malaysia. Tel: +603 8924 8350, Fax: +603 8924 8017, Email: (eyx6clw, key7cwc, kezklma, kezkps)@nottingham.edu.my.

with respect to a threshold T. This degree-0 zerotree is then coded as zerotree root (ZTR) if it is not the descendant of a previously found ZTR for that threshold T. However, if a coefficient is insignificant with respect to T but has some significant descendants, it is then coded as isolated zero (IZ). For a coefficient that is found to be significant with respect to T, it is coded as positive significant (POS) or negative significant (NEG) depending on the sign of the coefficient.

In comparison, the SPIHT coder performs significance tests on the individual tree nodes, the degree-1 zerotree (Type A set) and the degree-2 zerotree (Type B set). During SPIHT coding, the degree-1 and degree-2 zerotrees are partitioned into four individual tree nodes and four sub trees respectively when the significance test results are positive. It uses the set-partitioning approach where the test results are binary and since SPIHT provides more levels of descendant information for each coefficient tested, it gives a better coding performance than EZW. This hypothesis has been justified in [3, 4].

In this paper, a new family of set-partitioning coding schemes with a degree-k zerotree is introduced. The proposed coding schemes perform significance tests on a degree-1 zerotree up to a degree-k zerotree for a wavelet coefficient (i, j) using the set-partitioning approach similar to SPIHT.

The notations used in the proposed coding scheme for a five-scale subband decomposition are summarized as follows:

- SIG(i, j) is the significance of node (i, j);
- DESC(i, j) is the significance of descendants of node (i, j);
- GDESC(i, j) is the significance of grand descendants of node (i, j);
- GX2DESC(i, j) is the significance of great grand descendants of node (i, j).
- GX3DESC(i, j) is the significance of the descendants of node (i, j) in the first two scales of wavelet subbands.
- GX4DESC(i, j) is the significance of all leaf nodes of node (i, j) in the first scale of wavelet subbands.

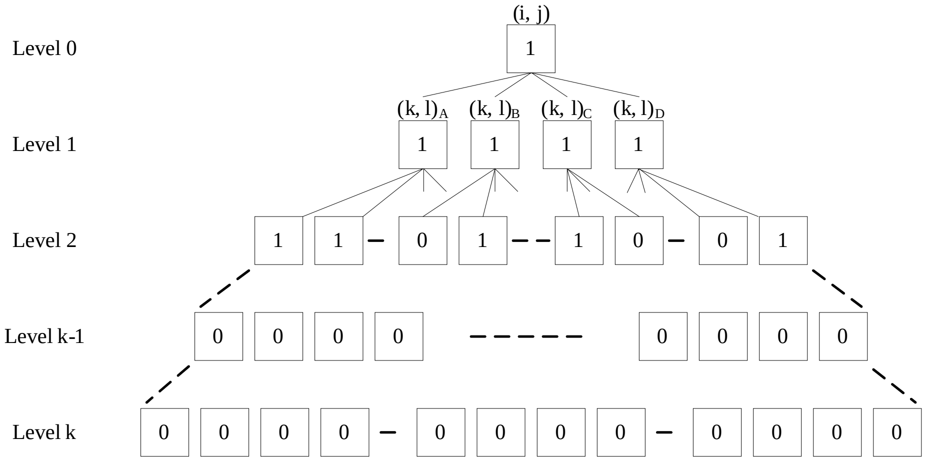
For a degree-1 zerotree coding, significance tests are performed on the individual coefficient (i, j) and on the descendants of the coefficient, i.e. SIG(i, j) and DESC(i, j) are determined in this coding.

A degree-2 zerotree coding tests the significance of a coefficient (i, j), the significance of the descendants of the coefficient and the significance of the grand descendants of the coefficient, i.e. SIG(i, j), DESC(i, j) and GDESC(i, j) are determined in this coding. This is equivalent to the traditional SPIHT coding.

An illustration of a degree-3 zerotree coding as compared to degree-2 zerotree coding is shown in Figure 1. In a degree-3 zerotree coding, a significance test is also carried out on the great grand descendants of the coefficient (i, j), i.e. GX2DESC(i, j) in addition to the significance tests on SIG(i, j), DESC(i, j) and GDESC(i, j). If GX2DESC(i, j) = 0, significance tests on the grand descendants for all offsprings of (i, j), i.e. GDESC(k, l) where (k, l) ϵ O(i, j), can be omitted. Therefore, when compared to a degree-2 zerotree coding, with just one extra GX2DESC(i, j) bit transmitted in a degree-3 zerotree coding, four GDESC(k, l) bits can be saved in the best case scenario if GX2DESC(i, j) is shown to be insignificant. This is illustrated in Figure 1.

Based on such deduction, it is anticipated that the more the levels of descendant information that can be obtained for each coefficient tested, the more there will be the number of encoding bits that can be saved. Therefore, a further improvement in performance can be expected for a degree-4 zerotree coding and an even better performance can be expected for a degree-5 zerotree coding.

Figure 2 shows the set-partitioning coding with a degree-5 zerotree which is a derivative of the SPIHT algorithm. The sets of coordinates used are: O(i, j) that holds the set of coordinates of all offsprings of node (i, j); H that holds the set of coordinates of all SOT roots (nodes in the highest pyramid level); L_k that holds the set of coordinates of all degree-k descendants.



Degree-2 Coding	Level 0:	DESC(i, j) = 1, GDESC(i, j) = 1.	Total bits sent for DESC and GDESC (degree-2 zerotree coding) = 10 bits.
	Level 1:	$\mathrm{DESC}(k, l)_A = 1$, $\mathrm{DESC}(k, l)_B = 1$, $\mathrm{DESC}(k, l)_C = 1$, $\mathrm{DESC}(k, l)_D = 1$, $\mathrm{GDESC}(k, l)_A = 0$, $\mathrm{GDESC}(k, l)_B = 0$, $\mathrm{GDESC}(k, l)_C = 0$, $\mathrm{GDESC}(k, l)_D = 0$.	
Degree-3 Coding	Level 0:	DESC(i, j) = 1, GDESC(i, j) = 1, **GX2DESC(i, j) = 0**.	Total bits sent for DESC and GDESC (degree-3 zerotree coding) = 7 bits.
	Level 1:	$\mathrm{DESC}(k, l)_A = 1$, $\mathrm{DESC}(k, l)_B = 1$, $\mathrm{DESC}(k, l)_C = 1$, $\mathrm{DESC}(k, l)_D = 1$.	

Figure 1. An illustration of a degree-2 and a degree-3 zerotree coding.

```
Set-Partitioning Coding with Degree-5 Zerotree

1. Initialization:
Output $n = [\log_2 (\max_{(i,j)} \{|c_{i,j}|\}]$; LSP is set as an empty list and the
coordinates (i, j) Є H is added to the LIP, those with descendants is
added to LIS as type A entries.

2. Sorting pass:
2.1  For each entry (i, j) in the LIP do:
  2.1.1  Output SIG(i, j).
  2.1.2  If SIG(i, j) = 1, move (i, j) to the LSP and output the sign of $c_{i,j}$.

2.2  For each entry (i, j) in the LIS do:
  2.2.1  If the entry is of type A, then
    ▪ If GDESC_PREV(parent of (i, j)) = 1 or (i, j) is a tree root
        ▪ Output DESC(i, j).
        ▪ If DESC(i, j) = 1, then
          • For each (k, l) Є O(i, j) do:
              o Output SIG(k, l).
              o If SIG (k, l) = 1, add (k, l) to the LSP and output the sign
                of $c_{k,l}$.
              o If SIG (k, l) = 0, add (k, l) to the end of the LIP.
          • If $L_2(i, j) \neq Ø$, move (i, j) to the end of the LIS as an entry of
            type B; else, remove entry (i, j) from the LIS.

  2.2.2  If the entry is of type B, then
    ▪ If GX2DESC_PREV(parent of (i, j)) = 1 or (i, j) is a tree root
        ▪ Output GDESC(i, j).
        ▪ If GDESC(i, j) = 1, then
          • GDESC_PREV(i, j) = 1.
          • For each (k, l) Є O(i, j) do:
              o Add (k, l) to the end of LIS as entry of Type A.
          • If $L_3(i, j) \neq Ø$, move (i, j) to the end of the LIS as an entry of
            type C; else, remove entry (i, j) from the LIS.

  2.2.3  If the entry is of type C, then
    ▪ If GX3DESC_PREV(parent of (i ,j)) = 1 or (i, j) is a tree root
        ▪ Output GX2DESC(i, j).
        ▪ If GX2DESC(i, j) = 1, then
          • GX2DESC_PREV(i, j) = 1.
          • If $L_4(i, j) \neq Ø$, move (i, j) to the end of the LIS  as an entry of
            type D; else, remove entry (i, j) from the LIS.

  2.2.4  If the entry is of type D, then
    ▪ If GX4DESC_PREV(parent of (i ,j)) = 1 or (i, j) is a tree root
        ▪ Output GX3DESC(i, j).
        ▪ If GX3DESC(i, j) = 1, then
          • GX3DESC_PREV(i, j) = 1.
          • If $L_5(i, j) \neq Ø$,  move (i, j) to the end of the LIS as an entry of
            type E; else, remove entry (i, j) from the LIS.

  2.2.5  If the entry is of type E, then
        ▪ Output GX4DESC(i, j).
        ▪ If GX4DESC(i, j) = 1, then
          • GX4DESC_PREV(i, j) = 1.
          • Remove entry (i, j) from the LIS.

3. Refinement pass:
For each entry (i, j) in the LSP, except those included in the last sorting
pass (i.e. with same n), output the n-th most significant bit of $|c_{i,j}|$.

4. Quantization-step update:
Decrement n by 1 and go to Step 2.
```

Figure 2. Set-partitioning coding with degree-5 zerotree.

Significance tests on the entries in the list of significant pixels (LSP) and entries in the list of insignificant pixels (LIP) in the set-partitioning coding with a degree-5 zerotree are similar to that of SPIHT. However, in addition to Type A which is the degree-1 descendant set and Type B which is the degree-2 descendant set, the list of insignificant sets (LIS) also contains new types of sets called Type C, Type D and Type E, which are the degree-3, degree-4 and degree-5 descendant sets respectively.

Our proposed degree-k zerotree coding has similar characteristics as that of the SPIHT algorithm where all the entries added to the end of the LIS are evaluated in the same sorting pass. Since they are partitioned, moved and added to the LIS based on the set-partitioning decision and are evaluated according to their order of importance, the bit streams generated have properties that are similar to that of embedded coding. Besides this, it also makes progressive transmission possible.

In terms of complexity, the proposed algorithm does not significantly increase the coder complexity as the fast technique used to identify zerotrees which is proposed in [5] can be incorporated into the proposed degree-k zerotree coding scheme. Here, zerotrees for all sorting passes are identified prior to encoding. This increases the processing speed of zerotree coding by eliminating the need for recursively checking the zerotrees in the sorting pass.

Due to the nature in which the order of wavelet decomposition is processed, coefficients in a lower level of wavelet sub trees are available before those coefficients in the higher levels. Thus, the zerotree information at the N level can be determined by performing a bitwise-OR operation on the zerotree information at the (N+1) level with the parent node at N level [5]. Since only bitwise-ORing operation is needed, the increase in complexity of the coder due to the significance tests on higher degree zerotrees becomes insignificant.

It is also shown in Figure 2 that significance maps that store the global significance test results GDESC_PREV(i, j) to GXnDESC_PREV(i, j) are used in the proposed set-partitioning algorithm with degree-k zerotree. These maps only require one bit per pixel and the size of GDESC_PREV(i, j) and GXnDESC_PREV(i, j) is only 1/16 and $1/4^{(n+1)}$ of the image size respectively. With the global significance storage, set-partitioning implementation of a higher degree zerotree coder can be performed easily without significantly increasing the coder complexity.

As aforementioned, it is anticipated that a better coding performance can be achieved with a higher degree zerotree coding. However, this only applies if the extra significance test that is carried out has an insignificant result of ‘0’. If the result of significance test is ‘1’, then one bit will be wasted since the decoder has to search the lower level of the tree.

From studies carried out on the degree-k zerotree coding, it has been found that at lower bit-rates where most of the coefficients are insignificant, a higher degree zerotree coding gives a better coding performance. On the other hand, at higher bit-rates, coding with a higher degree zerotree is less efficient since the wavelet coefficients are more likely to be significant as the number of planes encoded is increased. Hence, to obtain an optimal compression performance, the degree of zerotree tested is tuned in each encoding pass in the proposed TDKZW coding. Section III presents the coding methodology of the proposed work.

III. TUNED DEGREE-K ZEROTREE WAVELET CODING

The proposed TDKZW coding scheme is divided into three main stages as shown in Figure 3. Similar to other wavelet-based image coders, an image is first fed into the discrete wavelet transform (DWT) module for wavelet decomposition. Then, a pre-processing step is carried out on the wavelet coefficients to find the best tuning table for that particular image. Based on the result stored in the tuning table, the degree of zerotree tested is tuned in each encoding pass during the TDKZW coding. Finally, an embedded bit-stream is obtained and is either transmitted or stored for future use. The decoding process is just the reverse of the encoding without the pre-processing stage.

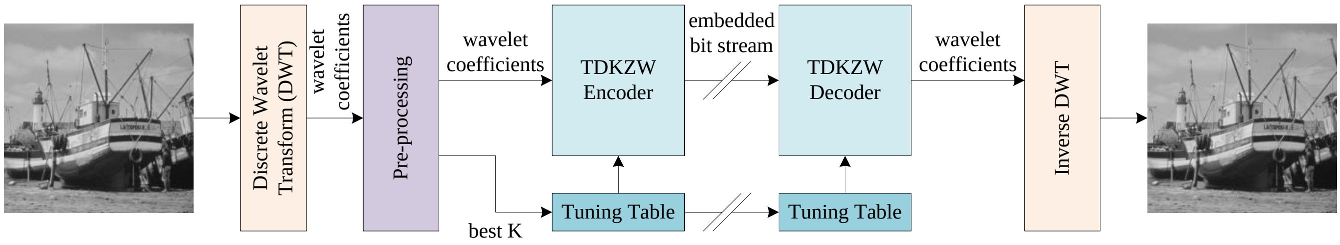


Figure 3. The proposed Tuned Degree-K Zerotree Wavelet (TDKZW) coding scheme.

A. *Discrete Wavelet Transform (DWT)*

DWT is applied to decompose an image into wavelet subbands at different resolution levels. In the proposed work, the reversible Le Gall 5/3 filter [6] is selected since it provides lossless transformation. The low-pass and high-pass filter coefficients of Le Gall 5/3 analysis and synthesis filters are given in (1) and (2) respectively and (3) gives the lifting implementation of the filter [6].

Le Gall 5/3 Analysis Filter: (1)

$$5\ tap\ low\ pass\ filter = [\,h_{-2},\ h_{-1},\ h_0,\ h_1,\ h_2\,]$$
$$h_2 = h_{-2} = -1/8$$
$$h_1 = h_{-1} = +1/4$$
$$h_0 = +3/4$$
$$3\ tap\ high\ pass\ filter = [\,g_{-1},\ g_0,\ g_1\,]$$
$$g_1 = g_{-1} = -1/2$$
$$g_0 = +1$$

Le Gall 5/3 Synthesis Filter: (2)

$$3\ tap\ low\ pass\ filter = [\,h'_{-1},\ h'_0,\ h'_1\,]$$
$$h'_1 = h'_{-1} = +1/2$$
$$h'_0 = +1$$
$$5\ tap\ high\ pass\ filter = [\,g'_{-2},\ g'_{-1},\ g'_0,\ g'_1,\ g'_2\,]$$
$$g'_2 = g'_{-2} = -1/8$$
$$g'_1 = g'_{-1} = -1/4$$
$$g'_0 = +3/4$$

Le Gall 5/3 Lifting Implementation: (3)

$$P(z) = \begin{bmatrix} 1 & \frac{1}{4}(1+z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{1}{2}(1+z^{-1}) & 1 \end{bmatrix}$$

B. *Pre-processing*

In the pre-processing stage, degree-1 to degree-k zerotree coding are first carried out on the image. These operations are performed in parallel to increase the processing speed of the encoder. The number of bits sent at each encoding pass for each degree-k zerotree coding is then recorded. Finally, the coding scheme with the lowest number of bits generated at each encoding pass is selected and is recorded in a tuning table.

For example, Table 1 and Table 2 show the number of bits sent for test images Lenna and Barbara at each bit-plane for a degree-1 to a degree-7 zerotree coding. From Table 1, it can be seen that at bit-plane 1 to 9, the least number of bits generated occurs at degree-7 zerotree coding whereas at bit-plane 10, bit-plane 11 and bit-plane 12 to 13, the least number of bits generated occurs at degree-3, degree-2 and degree-1 zerotree coding respectively. Thus, the best encoding path for Lenna is K = [7, 7, 7, 7, 7, 7, 7, 7, 7, 3, 2, 1, 1].

Table 1. Parallel degree-k zerotree coding (k = 1, 2, 3 … 7) were performed on test image Lenna to find the best encoding path for Tuned Degree-K Zerotree Wavelet (TDKZW) coder.

Number of bits generated at each encoding plane for degree-k zerotree coding on test image Lenna of size 512 x 512 pixels.								**Tuning Table**
Plane	k=1	k=2	k=3	k=4	k=5	k=6	k=7	
1	121	94	93	93	93	93	**93**	k=7
2	274	225	222	222	222	222	**222**	k=7
3	799	677	673	669	669	669	**669**	k=7
4	1840	1537	1525	1515	1503	1503	**1503**	k=7
5	4763	4040	4028	4020	4001	3988	**3988**	k=7
6	10505	8739	8634	8599	8571	8554	**8543**	k=7
7	20619	16949	16640	16483	16404	16361	**16346**	k=7
8	37637	31598	31068	30939	30904	30896	**30892**	k=7
9	66513	58627	57991	57879	57848	57841	**57841**	k=7
10	122825	114064	**113912**	113942	113954	113965	113966	k=3
11	227181	**225137**	225815	225990	226017	226017	226017	k=2
12	**311222**	313407	313636	313648	313648	313648	313648	k=1
13	**319090**	319447	319458	319458	319458	319458	319458	k=1

Table 2. Parallel degree-k zerotree coding (k = 1, 2, 3 … 7) were performed on test image Barbara to find the best encoding path for Tuned Degree-K Zerotree Wavelet (TDKZW) coder.

Number of bits generated at each encoding plane for degree-k zerotree coding on test image Barbara of size 512 x 512 pixels.								**Tuning Table**
Plane	k=1	k=2	k=3	k=4	k=5	k=6	k=7	
1	37	34	34	34	34	34	**34**	k=7
2	96	78	78	78	78	78	**78**	k=7
3	280	**241**	242	242	242	242	242	k=7
4	868	719	711	707	707	707	**707**	k=7
5	2235	1867	1866	1860	1847	1847	**1847**	k=7
6	5183	4317	4278	4264	4243	4231	**4230**	k=7
7	16108	13438	13219	13139	13104	13086	**13074**	k=7
8	43747	37775	37164	36927	36825	36766	**36750**	k=7
9	77015	68024	67117	66859	66757	66724	**66717**	k=7
10	119523	110995	110303	110203	110202	**110195**	110198	k=6
11	168897	161251	160903	160853	160839	**160836**	160837	k=6
12	234139	**230505**	230639	230714	230729	230732	230732	k=2
13	**300544**	302579	302854	302885	302887	302887	302887	k=1
14	**311418**	311772	311774	311774	311774	311774	311774	k=1

From Table 2, it should be noted that at bit-plane 3, although degree-2 zerotree coding gives the best performance, degree-7 zerotree coding is selected instead because of the nature of our proposed algorithm that encodes the wavelet coefficients starting from a higher degree to a lower degree zerotree coding. Thus, the best encoding path for Barbara is K = [7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 6, 2, 1, 1].

It can also be seen from Table 1 and Table 2 that at lower bit-rates where most of the coefficients are insignificant, a higher degree zerotree coding can code the image with a fewer number of bits. However, when the number of planes encoded is increased, coding with a higher degree zerotree becomes less efficient. This is because at higher bit-rates, wavelet coefficients are more likely to be significant and one bit will be wasted for every extra significance test performed.

C. TDKZW coding

After pre-processing, the TDKZW coding is carried out. The encoder is 'tuned' to the type of coding scheme according to the results stored in the tuning table. This will produce an encoded bit stream with the minimum number of bits possible. This means that with an individual tuning table for each image, an optimal coding performance can be obtained.

The overhead required by the proposed TDKZW coding is (S x P) bits where S is the number of binary bits needed to represent the highest degree of zerotree tested and P is the number of encoding planes needed. For example, the overhead needed for 13 encoding planes with coding up to a degree-7 zerotree is 39 (= 3 x 13) bits. As shown in Figure 4, the tuning table overhead is sent as a part of the encoding bit-stream. Once the TDKZW decoder is notified of the tuning scheme, the compressed image can be reconstructed accurately.

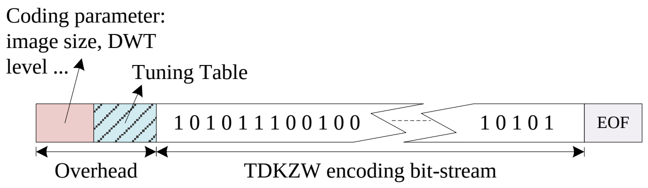


Figure 4. Tuned Degree-K Zerotree Wavelet (TDKZW) encoder bit-stream.

Although the pre-processing step which involves the parallel coding of degree-1 to degree-k zerotree coding does not significantly increase the processing time, it does increase the implementation cost of the TDKZW coder as multiple processors are needed (k numbers of processors for every degree-k zerotree coding where k = 1, 2, 3, etc.). Therefore, the proposed TDKZW coding scheme is primarily aimed at single-encoding, multiple-decoding image processing applications such as medical imaging, satellite imaging and online photo hosting applications. In such applications, the image is encoded only once and is stored in a database for future use.

It should also be noted that the proposed TDKZW coding is applicable to both lossy and lossless image compression. Since the bit-stream is generated in accordance to the order of importance, the end user can choose to stop decoding as soon as the targeted bit-rates are achieved and yet, will still be able to obtain a fully reconstructed image of sufficiently good quality. For a lossless reconstruction, the whole bit-stream is retrieved and a reconstructed image which is identical to the original image can be obtained. The advantage of this embedded coding is that both the lossy and lossless reconstructions can be performed using a single compression file.

IV. PERFORMANCE EVALUATION

Software simulations using MATLAB were carried out to evaluate the performance of both lossless and lossy compression using the proposed TDKZW coding. Natural grayscale images of size 512 x 512 pixels were used in all the simulations. The proposed work uses the reversible Le Gall 5/3 wavelet transform. In addition, a six-scale wavelet decomposition and a seven-scale spatial orientation tree decomposition were performed on the test images.

A. Pre-processing for Tuning Tables

Table 3 gives the tuning tables used in the simulations. These individual tuning tables are obtained after the pre-processing stage for each of the test images.

Table 3. Tuning tables for Tuned Degree-K Zerotree Wavelet (TDKZW) coding for standard test images used in our simulations. All test images are of size 512 x 512 pixels.

	Tuning Table for Degree-K Zerotree Coding, K													
Bit-Plane	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Airplane	7	7	7	7	7	7	7	7	7	5	5	2	1	n.a
Barbara	7	7	7	7	7	7	7	7	7	6	6	2	1	1
Boat	7	7	7	7	7	7	7	7	6	3	2	1	1	n.a
Fingerprint	7	7	7	7	7	7	7	7	7	5	2	1	1	n.a
Goldhill	7	7	7	7	7	7	7	7	7	6	3	1	1	1
Lenna	7	7	7	7	7	7	7	7	7	3	2	1	1	n.a
Zelda	7	7	7	7	7	7	7	7	7	6	2	1	1	n.a

Table 4. Performance comparison of Tuned Degree-K Zerotree Wavelet (TDKZW) coder versus existing coders on natural images in terms of lossless compression efficiency represented by bit-per-pixel (bpp).

	Embedded Coders					Non-Embedded Coder
Images	TDKZW	SPIHT	PPBWC	JPEG2000	PDF 1-Pass	CALIC
Airplane	**3.28**	3.74	3.92	-	-	3.83
Barbara	4.49	4.71	4.68	4.69	**4.21**	4.49
Boat	4.01	4.35	4.08	4.43	4.08	**3.78**
Fingerprint	**5.13**	5.23	-	5.69	5.40	-
Goldhill	4.51	4.78	4.54	4.87	4.69	**4.39**
Lenna	**3.91**	4.20	4.15	4.35	4.07	4.04
Zelda	**3.58**	3.94	-	4.02	3.79	-
Average:	4.13	4.42	4.27	4.68	4.37	**4.11**

B. Lossless Compression using TDKZW Coder

The lossless compression efficiency of TDKZW coding was compared to other existing popular lossless coders such as SPIHT [2, 7], PPBWC [8], JPEG2000 [9], progressive lossless/near-lossless image coding (PDF 1-Pass) [10, 11] and CALIC [12]. The coding methodology of each of these coders is documented in [2, 7-12]. It should be noted here that all of these are embedded coders except for CALIC.

For lossless compression, our proposed TDKZW coding applies a context-based arithmetic coding to encode the binary bit-stream generated by the TDKZW encoder module prior to transmission. Table 4 shows the lossless compression efficiency obtained from the simulations carried out in terms of bit-per-pixel (bpp). The results for lossless SPIHT are obtained from [7] which uses an integer multi-resolution S+P transformation presented in [13]. Results for PPBWC and CALIC are extracted from [8] and the results of JPEG2000 and PDF 1-Pass are extracted from [10]. The bold text shows the lowest bpp obtained and results that are not available are indicated by a dash line.

From the simulation results shown in Table 4, it can be seen that our proposed TDKZW coding gives a better lossless compression efficiency for four out of the seven images tested. Among the five embedded coders, TDKZW coding is found to require the fewest encoding bits except for image Barbara. It can also be seen that our TDKZW coding has the highest average lossless compression efficiency of 4.13 bpp compared to the other embedded coding techniques, with a maximum difference of 0.55 bpp.

Although CALIC gives a slightly better average lossless coding performance of 4.11 bpp compared to TDKZW coding with an average of 4.13 bpp, CALIC is a non-embedded coder. In comparison, our proposed TDKZW coder not only provides embedded coding, it also allows progressive transmission to be carried out. This makes our TDKZW coding applicable to both lossless and lossy image compression.

It should be noted that in [14], the proposed lossless TDKZW coder uses a predefined tuning table for all test images. However, in this paper, a pre-processing step is carried out to determine the individual tuning table for each test image. Simulation results presented in this paper show that there is a slight improvement in lossless compression performance for test images Barbara and Fingerprint compared to the work published in [14].

C. Lossy Compression using TDKZW Coder

The binary uncoded compression performance of the proposed TDKZW coder was also evaluated. Here, the performance of the lossy compression of our proposed work was compared to the binary uncoded SPIHT coder. For a meaningful comparison, both the TDKZW and SPIHT coders used the reversible Le Gall 5/3 wavelet transform.

Table 5 records the simulation results obtained at various bit-rates (bpp) in terms of the peak signal-to-noise ratio (PSNR) and Figure 5 shows the reconstructed images at 0.10 bpp for Fingerprint using both the TDKZW and SPIHT coder.

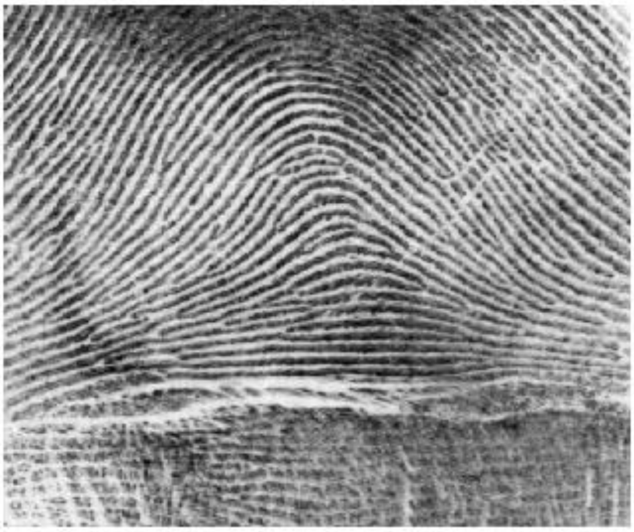
From Table 5, it can be seen that our proposed TDKZW coder outperforms the SPIHT coder at all bit-rates with the maximum improvement of 0.30 dB at 1.00 bpp for Fingerprint. Since the degree of zerotree tested is tuned in the proposed coding scheme, a better compression performance is expected since there are more significant coefficients that are encoded at a particular bit-rate compared to the SPIHT coder. This can be verified from the reconstructed images in Figure 5 where the proposed TDKZW coder can be seen to give a better reconstructed visual quality than the SPIHT coder.

V. CONCLUSION

Since the degree of zerotree tested is tuned in each encoding pass, the proposed TDKZW algorithm gives an optimal compression efficiency. The proposed coding scheme which uses a set-partitioning approach provides embedded coding and allows progressive transmission to take place which makes it applicable to both lossless and lossy image compression. Simulation results show that the proposed TDKZW coder gives a higher lossless compression efficiency than SPIHT and many other existing lossless image compression techniques. In addition, our proposed coder has also been shown to consistently outperform the SPIHT coder for lossy compression.

Table 5. Performance comparison of the proposed Tuned Degree-K Zerotree Wavelet (TDKZW) coder versus SPIHT coder on natural images in terms of peak signal-to-noise ratio (PSNR) at various bit-rates (bpp).

Airplane	PSNR (dB)	
Bit-Rate (Bpp)	TDKZW	SPIHT
0.25	33.03	32.89
0.50	36.83	36.75
1.00	41.44	41.37
2.00	47.38	47.29
Barbara	PSNR (dB)	
Bit-Rate (Bpp)	TDKZW	SPIHT
0.25	26.23	26.17
0.50	29.70	29.63
1.00	34.49	34.34
2.00	41.40	41.32
Boat	PSNR (dB)	
Bit-Rate (Bpp)	TDKZW	SPIHT
0.25	30.13	30.05
0.50	33.53	33.43
1.00	38.29	38.21
2.00	43.63	43.58
Fingerprint	PSNR (dB)	
Bit-Rate (Bpp)	TDKZW	SPIHT
0.25	24.05	23.88
0.50	27.55	27.27
1.00	31.25	30.95
2.00	36.94	36.74
Goldhill	PSNR (dB)	
Bit-Rate (Bpp)	TDKZW	SPIHT
0.25	29.99	29.98
0.50	32.42	32.37
1.00	35.78	35.71
2.00	40.85	40.84
Lenna	PSNR (dB)	
Bit-Rate (Bpp)	TDKZW	SPIHT
0.25	33.06	32.98
0.50	36.20	36.13
1.00	39.44	39.38
2.00	43.82	43.78
Zelda	PSNR (dB)	
Bit-Rate (Bpp)	TDKZW	SPIHT
0.25	36.82	36.76
0.50	39.08	39.05
1.00	41.42	41.37
2.00	45.31	45.25



(a) Original Image

(b) TDKZW (20.49 dB @ 0.10 bpp)

(c) SPIHT (20.26 dB @ 0.10 bpp)

Figure 5. Comparison of the original and reconstructed images at 0.10 bpp using the proposed Tuned Degree-K Zerotree Wavelet (TDKZW) coder and the SPIHT coder. Test image: Fingerprint (a) Original image, (b) Reconstructed image using TDKZW coder and (c) Reconstructed image using SPIHT coder.

REFERENCES

[1] J. M. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3445-3462, Dec. 1993.

[2] A. Said and W. A. Pearlman, “A new fast/efficient image codec based on set partitioning in hierarchical trees,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 12, pp. 243-250, Jun. 1996.

[3] Y. Cho and W. A. Pearlman, “Quantifying the coding performance of zerotrees of wavelet coefficients: degree-k zerotree,” *IEEE Trans. Signal Process.*, vol. 55, no. 6, pp. 2425-2431, Jun. 2007.

[4] L. Cicala and G. Poggi, "A generalization of zerotree coding algorithms," in *Picture Coding Symposium 2007*, Lisboa, Portugal, 7-9 Nov. 2007.

[5] J. M. Shapiro, "A fast technique for identifying zerotrees in the EZW algorithm," *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 1455-1458, May. 1996.

[6] T. Archarya and P.-S. Tsai, JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures, Wiley-Interscience, 2004.

[7] A. Said and W. A. Pearlman, "SPIHT image compression: Demo Programs/Downloads," [Online] http://www.cipr.rpi.edu/research/SPIHT/spiht3.html. Date Access: 14 Apr. 2009.

[8] H. Pan, W. -C. Siu, N. -F. Law, “Lossless image compression using binary wavelet transform,” *IET Image Process.*, vol. 1, no. 4, pp. 353-362, Dec. 2007.

[9] D. Taubman and M. W. Marcellin, JPEG 2000, Image Compression, Fundamentals, Standards and Practice, Kluwer Academic Publishers, 2002.

[10] I. Avcıbas, N. Memon, B. Sankur, K. Sayood, “A Successively Refinable Lossless Image-Coding Algorithm,” *IEEE Trans. Comm.*, vol. 53, no. 3, Mar. 2005.

[11] I. Avcıbas, N. Memon, B. Sankur, K. Sayood, “A Progressive Lossless/Near-Lossless Image-Compression Algorithm ,” *IEEE Signal Process. Lett.*, vol. 9, no. 10, Oct. 2002.

[12] X. Wu and N. Memon, “Context-based, adaptive, lossless image coding,” *IEEE Trans. Comm.*, vol. 45, no. 4, pp. 437-444, Apr. 1997.

[13] A. Said and W. A. Pearlman, "An image multiresolution representation for lossless and lossy compression," *IEEE Trans. Image Process.*, vol. 5, no. 9, pp. 1303-1310, Sep. 1996.

[14] L. W. Chew, Li-M. Ang, and K. P. Seng, “Lossless Image Compression using Tuned Degree-K Zerotree Wavelet Coding”, *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMESC 2009)*, Hong Kong, vol. 1, pp. 779-782, Mar. 2009.