Quantifying the Coding Performance of Zerotrees of Wavelet Coefficients: *Degree-k Zerotree*

Yushin Cho and William A. Pearlman

***Abstract*— Locating zerotrees in a wavelet transform allows encoding of sets of coefficients with a single symbol. It is an efficient means of coding if the overhead to identify the locations is small compared to the size of the zerotree sets on the average. It is advantageous in this regard to define classes of zerotrees according to the levels from the root until the remainder of the tree contains all zeroes. We call a tree with all zeroes except for the top k levels a degree-k zerotree. A degree-k zerotree coder is one that can encode degree-0 through degree-k zerotrees. We quantify the bit savings of a degree-k_2 over a degree-k_1, $k_2 > k_1$, coder. Because SPIHT is a degree-2 zerotree coder and EZW a degree-0 zerotree coder, we are able to explain the superior efficiency of SPIHT. Finally, we gather statistics of degree-k zerotrees for different values of k in the bitplanes of several image wavelet transforms to support our analysis of the coding performance of degree-k zerotree coders.**

***Index Terms*— zerotree, zerotree coder, SPIHT, EZW, wavelet image compression.**

I. Introduction

Two popular wavelet image coders, EZW [1] and SPIHT [2], use discovery of zerotrees to encode sets of wavelet coefficients efficiently. While the reason for different zerotree coding performance of the two schemes in terms of zerotree was not clearly stated in the literature, we establish a framework to explain it formally.

Both EZW and SPIHT use the idea of decaying spectral power density and successive quantization approximation. For each coding pass of these algorithms, a significance map is constructed, which contains the significance information of every coefficient with respect to a given threshold. When the coefficient's magnitude exceeds or equals the threshold, it is said to be *significant* and a '1' is posted; otherwise, it is said to be *insignificant* and a '0' is posted. The threshold decreases successively by a factor of 2 for each new pass, enabling the more important coefficients to be coded first.

Generally, the wavelet based image coding in EZW and SPIHT consists of two passes: *sorting* and *refinement*. The *sorting* pass encodes the location of the first significant bit (highest '1' in binary expansion of magnitude) of each coefficient and the locations of the insignificant (magnitude below current threshold or '0' in current bit plane) coefficients and tree-structured sets of insignificant coefficients, while the *refinement* pass simply encodes (records) the lower order bits of significant coefficients. In EZW, these passes are named differently as *dominant* and *subordinate* passes, respectively. Note that the coding of every zerotree is achieved in the *sorting* pass. The concept of a zerotree is based on the property that if a coefficient in a wavelet transform is insignificant, it is very likely that its descendant coefficients in higher frequency subbands are also insignificant. If a coefficient and all of its descendant coefficients are insignificant (i.e. zero in a bitplane), a zerotree is found in the EZW algorithm. The zerotree defined in EZW is simply a tree consisting of all zero values. We denote this zerotree as *degree-0 zerotree*.

On the other hand, the zerotrees in SPIHT are defined in a wider sense. SPIHT can represent two more classes of zerotrees. It treats a root coefficient and its corresponding descendants separately. So, if a tree has a significant root coefficient and remaining coefficients in the tree are all insignificant, all these insignificant coefficients are coded by one zerotree symbol, while the significant root is coded by another symbol. We denote this class of zerotree to be coded as *degree-1 zerotree* since every coefficient except at the top level is all zeros.

In addition, SPIHT can treat indirect descendant coefficients separately from a root and children coefficients. (Indirect descendant coefficients mean all coefficients except the root and its direct children.) Thus, if any coefficients at the children level are significant and all coefficients below them are insignificant, all the zeros below children level are coded by one zerotree symbol. The root and children coefficients are coded separately. We denote this class of zerotree to be coded as *degree-2 zerotree* since every coefficient except at the top two levels is zero.

Our models of zerotrees are based on their degree. At present, no image coder that can code zerotrees with more than degree-2 has been reported. The degree-2 zerotree is the maximum complexity of zerotree discovered so far and is used by the SPIHT image compression algorithm. In the viewpoint of block entropy coding, the entropy coding performance of a zerotree is defined and explained simply. Based on the suggested framework, the possibility of further improvement of SPIHT or any other zerotree-based algorithm is discussed [3].

Ramaswamy et al. [4] presented a criterion of 'cumulative zerotree count' to analyze the performance of the SPIHT coder. However, its purpose was to evaluate the wavelet filters in the SPIHT algorithm. Moreover, only the number of zerotrees was measured and the height of the zerotree was not considered.

Y. Cho is with Sony Electronics, San Jose, CA 95112 USA; E-mail: cho.yushin@gmail.com

W. A. Pearlman is with Rensselaer Polytechnic Institute, Electrical, Computer and Systems Engineering Dept., Troy, NY 12180-3590; E-mail: pearlw@ecse.rpi.edu; Tel/Fax: (518) 276-6082/8715

This work was performed in the Center for Image Processing Research at Rensselaer Polytechnic Institute. The equipment and facilities were augmented by CISE Infrastructure Award No. EIA 0224433 of the National Science Foundation, for which we are grateful. The government has certain rights in this material.

II. Testing for Zerotrees

Every coefficient of a wavelet transform is stored by its sign and binary expansion of its magnitude. Thresholds for testing significance are positive integer powers of 2, so these powers correspond to numbers or levels of bitplanes. The search for significant coefficients starts from the highest bitplane with most significant bits (MSB's) and proceeds successively to the lower bitplanes until it reaches the lowest (0) bitplane with the least significant bits (LSB's). When a coefficient *first* shows '1' in a bitplane, it is significant; otherwise it remains insignificant.

The bitplanes of a 4-level wavelet transform are shown in Figure 1. In the figure, there are 13 bitplanes shown, where bitplane 0 is the least significant bitplane (LSB) and bitplane 12 is the most significant bitplane (MSB). The two least significant bitplanes, 0 and 1, are assumed zeros here since they are not coded. The white areas indicate zeros, and the grey ones indicate not-all-zeros. Trees are formed within bitplanes by four-fold successive branching from a coefficient to its offspring in the next higher resolution (lower resolution number) at the same spatial orientation. A zerotree in a bitplane is a tree of all '0's at locations where there are no '1's in any higher bitplane. Therefore, because of the order of the search, once a coefficient is found to be significant, it can no longer belong to a zerotree.

Since most of the energy is concentrated in the lower frequency subbands, the large magnitude wavelet coefficients are found in the lower frequency subbands with very few exceptions. Thus, the top bitplanes contain significant bits (i,e. '1's) only in lower frequency subbands, as shown in the grey areas in bitplane 12 of Figure 1. Therefore, the zerotrees in these bitplanes tend to be long, extending from the low frequency subband at the top left to a high frquency subband at the bottom. In the lower bitplanes, we tend to find blocks of zeros only at the higher frequencies, since coefficients in the lower frequency subbands tend to have '1's in higher bitplanes, as illustrated also in Figure 1. Thus, the zerotrees in the lower bitplanes are often shorter than those in the higher bitplanes.

In EZW, once a wavelet coefficient is tested as significant for a given threshold, each of the four subtrees branching from this coefficient should be tested for its significance, i.e. if the subtree has any significant bit for a given threshold. Thus, four symbols (one for each subtree) necessarily follow, regardless of the significance of the four branched descendants.

Meanwhile, SPIHT handles this situation differently. It codes parent coefficients and children coefficients separately. It has a special syntax representing whether any of its children coefficients is significant. If there is no significant child coefficient, code '0'; otherwise, code '1'. Then code together these four bits, each of which specifies the significance of a child coefficient. Here, the syntax requires only one symbol or one bit (output '0' for the value of $D(i,j)$ in the original SPIHT article [2]) if there is no significant child coefficient.

There is also more advanced and sophisticated syntax defined in SPIHT, to represent the tree subset having zeros for all descendant coefficients of the four children. This again takes only one bit (output '0' for the value of $L(i,j)$ in the original article) if all those descendant coefficients are zeros.

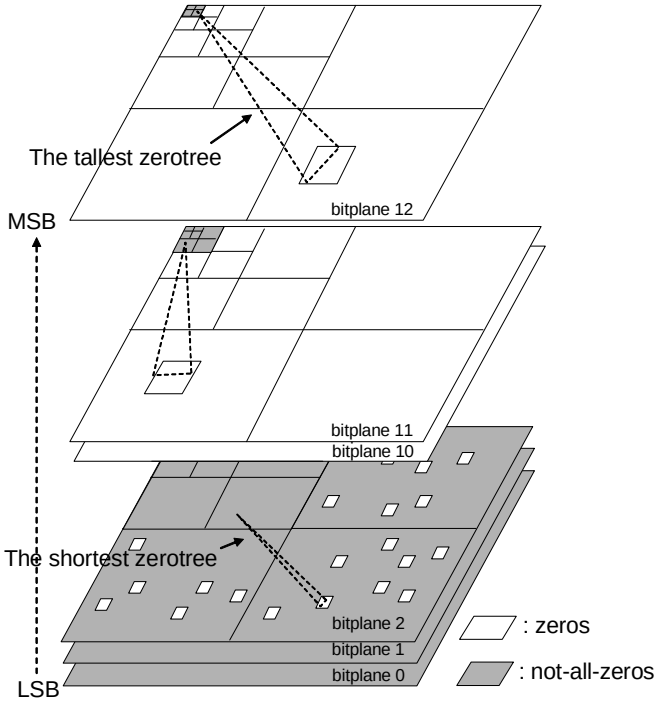


Fig. 1. Zerotrees on bitplanes

These are the most important reasons why the SPIHT algorithm improves the EZW algorithm in compression efficiency.

III. Degree-k Zerotree

We establish definitions and theorems regarding the efficacy of zerotrees, which can formally explain the differences of zerotree coding performance between popular EZW, SPIHT and possibly other zerotree-based algorithms. Viewing a zerotree as an entropy coding scheme, we classify it into different classes depending on the fullness of zeros at each level.

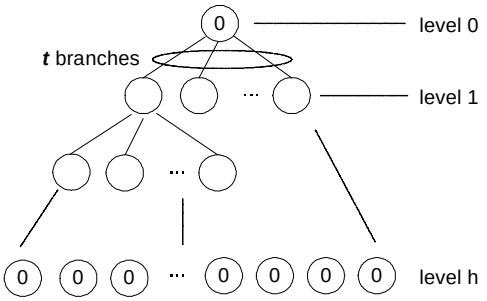


Fig. 2. A height h, t-ary tree

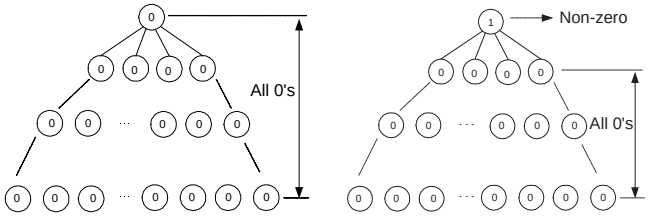
Through all following definitions, theorems, and proofs, we assume that each zerotree is of height h, t-ary, and complete (i.e. full leaves at bottom level), as shown in Fig. 2. And let us call this tree a *source tree* or a *source zerotree*. Note that the bottom level of a zerotree always indicates the highest resolution of subband.

The level 0 of a tree indicates the root node (i.e. the top) and the level h indicates the leaf nodes (i.e. the bottom). Note

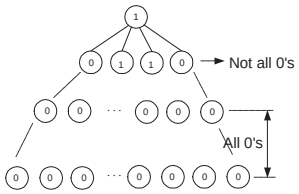
that the level is numbered starting from the top (root) level. Thus, a height h tree has $h+1$ levels, i.e. level 0 to level h. At level i, there are t^i nodes and the total number of nodes in the tree is simply $T = \sum_{i=0}^{h} t^i = \frac{t^{h+1}-1}{t-1}$.

Each node of the tree is associated with a binary number (i.e. 0 or 1), corresponding to a two symbol alphabet, $\{0, 1\}$. Each node has its value, which is the significance of a wavelet coefficient for a given threshold. Representing each node as a random variable X with a zero-order statistic, i.e., 0 and 1 are equally probable, we need N bits to encode a sequence of N nodes, $X_0, X_1, \cdots, X_{N-1}$. Thus, for a height α, t-ary subtree, the required number of bits to code it is equal to the total number of nodes in the subtree, i.e. $\frac{t^{\alpha+1}-1}{t-1}$ (bits).

Definition 1 (Degree-k zerotree): For any complete t-ary tree of height h, if all nodes from the bottom level (i.e. the level h) to the level k have zero values, we call the tree a 'degree-k zerotree'. In other words, all nodes except the top k levels have zero values in a degree-k zerotree.



(a) degree-0 zerotree (b) degree-1 zerotree



(c) degree-2 zerotree

Fig. 3. Degree-0, degree-1 and degree-2 zerotrees

So, the degree-0 zerotree is the tree having all zeros (See Fig. 3 (a)). The degree-1 zerotree (See Fig. 3 (b)) is the tree having all zeros except the root node. And, the degree-2 zerotree (See Fig. 3 (c)) is the tree having all zeros except the root node and the children nodes of root node.

Table I shows how EZW and SPIHT differently code the degree-0, -1, and -2 zerotrees in Fig. 3. As discussed in [1], the EZW coder has four symbols: POS (Positive Significant), NEG (Negative Significant), ZTR (Zerotree root), and IZ (Isolated Zero). IZ means a coefficient is insignificant but has some significant descendant.

The second symbol '0' (bold faced in Table I) alone in SPIHT's code for both degree-0 and degree-1 examples informs that there exists a degree-1 zerotree. The second symbol '1' (bold faced in Table I) in SPIHT's code for the degree-2 example informs that there does not exist a degree-1 zerotree. The last symbol '0' (bold faced) in SPIHT's code for degree-2 example informs that there exists a degree-2 zerotree. Note that 2 bits are required to code each symbol of EZW without entropy coding.

TABLE I

EXAMPLE OF CODED SYMBOLS GENERATED BY EZW AND SPIHT FOR DEGREE-1 AND DEGREE-2 ZEROTREE

	EZW	SPIHT
degree-0 zerotree in Fig. 3 (a)	*ZTR*	*0,**0***
degree-1 zerotree in Fig. 3 (b)	*POS,ZTR,ZTR,ZTR,ZTR*	*1,**0***
degree-2 zerotree in Fig. 3 (c)	*POS,ZTR,POS,POS,ZTR, ZTR,ZTR,ZTR,ZTR,ZTR,ZTR,ZTR,ZTR*	*1,**1**,0,1,1,0,**0***

Now we derive the rule which can be generally applied to an image coding algorithm using zerotrees of wavelet coefficients. Basically, without using a zerotree symbol, a source degree-k zerotree of a height h, t-ary is coded by two parts:

1) Non-zero part : Code all symbols from top (root) level (i.e. level 0) to level $k-1$, which are not all zeros (For degree-0 zerotree, there is no non-zero part). The number of the symbols is $N_k = \sum_{i=0}^{k-1} t^i$. By the definition of degree-k zerotree, at least one node from level $k-1$ is non-zero (i.e. one). These symbols can be modeled as a sequence of random variables, i.e. $X_0, X_1, \cdots, X_{N-1}$. Since 0 and 1 are assumed to be equally probable, N_k bits are required to represent this sequence.
2) Zero part : Code all symbols from level k to bottom level (i.e. level h), which are all zeros. The number of the symbols is $\sum_{i=k}^{h} t^i$. Since we assumed 0 and 1 are equally probable, $\sum_{i=k}^{h} t^i$ bits are required to represent this sequence of zeros.

However, if we use a zerotree symbol, the zero part can be coded by only one symbol.

The number of bits saved by the use of a degree-k zerotree symbol is the number of nodes from level k to bottom level in the zerotree minus one for the zerotree root symbol.

Definition 2 (Bit savings of a degree-k zerotree symbol): In representing a degree-k zerotree of height h and t-ary, the bit savings S_k by using a degree-k zerotree symbol is simply:

$$S_k = \sum_{i=k}^{h} t^i - 1 \quad \text{(bits)}$$

For example, for a height 3, 4-ary, degree-1 zerotree, the total number of nodes T is 10. We use one bit to represent the root node at level 0, and use another bit to represent the degree-1 zerotree. Then, the bit savings S_1 is 9-1 = 8, since without degree-1 zerotree symbol it needs 9 bits to represent the 9 nodes from level 1 to level 3 (bottom). The bit savings S_k is larger for the taller zerotree, i.e. the larger height h.

Definition 3 (Coding fraction of degree-k zerotree): Following above definition, the coding fraction F of degree-k zerotree in height h and k-ary tree is:

$$\frac{T - S_k}{T} = 1 - \frac{S_k}{T}.$$

For example, the bit savings S_1 of 8 (bits) for the above example, the coding fraction is calculated as $\frac{10-8}{10} = 0.2$, which means that only 20% of the original nodes in the source tree is coded by exploiting the degree-1 zerotree symbol, to represent the tree.

Therefore, the coding fraction F decreases as the bit savings obtained by zerotree increases. The range of coding fraction F is: $0 < F \leq 1$.

Theorem 1: For $k_1 < k_2$, the bit savings of degree-k_2 zerotree over degree-k_1 zerotree, D_{k_1,k_2}, is:

$$D_{k_1,k_2} = \sum_{i=k_1}^{k_2-1} t^i \,(\text{bits})$$

and the difference of coding fraction is:

$$\frac{D_{k_1,k_2}}{T}.$$

Proof: The difference of bit savings is:

$$D_{k_1,k_2} = \left(\sum_{i=k_1}^{h} t^i - 1\right) - \left(\sum_{i=k_2}^{h} t^i - 1\right) = \sum_{i=k_1}^{k_2-1} t^i \,(\text{bits}).$$

■

Corollary 1: The difference of bit savings between degree-0 zerotree and degree-1 zerotree is only one bit.

The proof is straightforward since a degree-0 zerotree will represent one more symbol than a degree-1 zerotree, i.e. root node. This implies that the difference of coding fraction is $1/T$, which is very small for large T and thus the coding performance of degree-0 and degree-1 zerotree is very close.

Definition 4 (Degree-k zerotree coder): A degree-k zerotree coder is a zerotree coder which can represent all zerotrees with degree-i, $0 \leq i \leq k$.

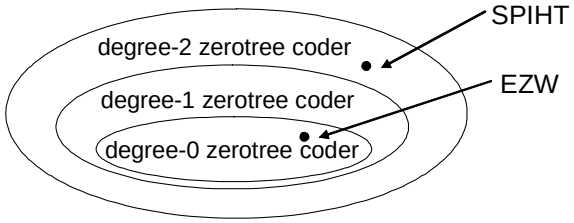


Fig. 4. Relationship of coding performances among degree-0, 1, 2 zerotree coders

By the definition of '*degree-k zerotree coder*' above, the degree-2 zerotree coder, as an example, can code all degree-0, -1, and -2 zerotrees. Hence, it is a more powerful coder than both degree-0 and -1 zerotree coders and has lower coding fraction (See Fig. 4). Note that a degree-0 zerotree source is represented with two symbols 00 by SPIHT algorithm.

Examples of coded bitstream for degree-0,1,2 zerotree sources by degree-0, 1, 2 zerotree coders are demonstrated in Table II. Assume that we use only two symbols 0 and 1 to code each binary decision of the zerotree coder. In the table, the d_i means degree-i. The 0_i or 1_i indicates the existence of degree-i zerotree, 0_i for existence and 1_i for non-existence. The d_0, d_1, and d_2 source zerotrees correspond to Figs. 3 (a),(b), and (c), respectively.

Note that a higher degree zerotree coder generates a shorter symbol stream among the three kinds of zerotree sources.

Theorem 2: For coding a degree-k_2 zerotree source, the maximum bit savings of degree-k_2 zerotree coder over degree-k_1 zerotree coder with $k_1 < k_2$ is:

$$t^{k_2-k_1} - 1 \,(\text{bits}).$$

TABLE II

EXAMPLE OF CODED SYMBOLS BY DEGREE-0, 1, 2 ZEROTREE CODERS

	source zerotree		
zerotree coder	d_0 zerotree	d_1 zerotree	d_2 zerotree
d_0 coder	0_0	$1_0 1_0 0_0 0_0 0_0 0_0$	$1_0 1_0 0_0 1_0 1_0 0_0 0_0 0_0$ $1_0 1_0 0_0 0_0 0_0 0_0$
d_1 coder	0_0	$1_0 1_0 1_0$	$1_0 1_0 0_1 1_0 1_0 1_0 1_0 0_0$
d_2 coder	0_0	$1_0 1_0 1_0$	$1_0 1_1 1_0 2_0 1_1 0$

Proof: A degree-k_2 zerotree coder can simply represent a degree-k_2 zerotree source with one symbol. However, since $k_1 < k_2$, a degree-k_1 zerotree coder rooted at top level 0 of a degree-k_2 source zerotree cannot represent the source zerotree. Instead, by having the roots of degree-k_1 zerotree coders at level $(k_2 - k_1)$, a degree-k_2 zerotree can be represented by a multiple of degree-k_1 zerotree coders. The number of these degree-k_1 zerotree coders minus one equals the bit savings.

If $k_1 = 0$, degree-0 zerotrees rooted at level k_2 are coded by degree-0 zerotree coders rooted at level k_2. Similarly, if $k_1 = 1$, degree-1 zerotrees rooted at level $k_2 + 1$ are coded by degree-1 zerotree coders rooted at level $k_2 + 1$. In this way, degree-k_1 zerotrees rooted at level $k_2 - k_1$ are coded by degree-k_1 zerotree coders rooted at level $k_2 - k_1$. The numbers of these additional zerotree coders rooted at level $k_2 - i$, $i = 0, 1, \cdots, k_1$ are : t^{k_2}, t^{k_2-1}, $\cdots$, $t^{k_2-k_1}$, respectively. Fig. 5 shows that a degree-k_2 zerotree is coded by $t^{k_2-k_1}$ degree-k_1 zerotree coders (shaded part). ■

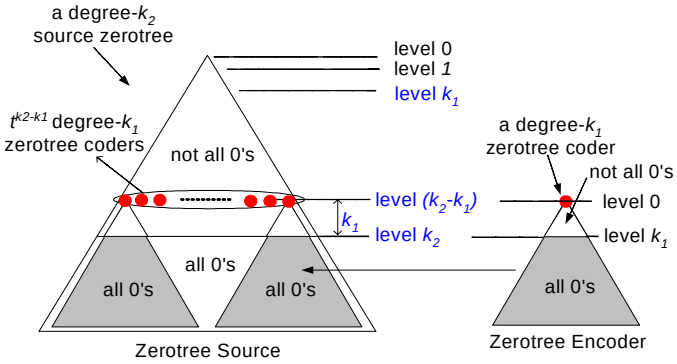


Fig. 5. A degree-k_2 zerotree coded by $t^{k_2-k_1}$ degree-k_1 zerotree coders

From the above definitions and theorems, our analysis on the performance difference between EZW and SPIHT is: EZW algorithm is only using degree-0 zerotrees, while SPIHT is using degree-1 and -2 zerotrees as well. The $D(i,j)$ of types A and B in SPIHT correspond to degree-1 and -2 zerotrees, respectively.

Ideally, if we were to use a higher degree zerotree coder, i.e. degree-m, $m > 2$, better coding performance would be expected. The hurdle for this, however, is the increased complexity in the implementation of the set partitioning engine. Also, because the number of wavelet decompositions, or equivalently the height of a spatial orientation tree, is usually not more than 5, 6, or 7, the zerotrees of degree greater than 2 do not occur frequently. An experimental analysis of the frequency of degree-1,2,3 zerotrees is presented in the next

section. In our experiments with degree-3 zerotree SPIHT coder, no coding improvement is achieved. In each source tree, if it is not a degree-2 zerotree, then we test if it is a degree-3 zerotree, and finally we code a degree-3 zerotree symbol to represent the test result. If the tree is a degree-3 zerotree, there is apparent coding gain. However, if it is not, then the coded degree-3 zerotree symbol works as overhead. In most wavelet transformed images the frequency of occurrence of the degree-3 zerotree is very low. And this means there is no coding improvement with degree-3 zerotree coders. The same is expected to be true for higher degree zerotree coders.

IV. Experimental Analysis

We will show the effectiveness of a higher degree zerotree coder by showing the actual occurrences of higher degree zerotrees in experiments. Also, we try to show that the characteristics of zerotrees depend on the significance of the bitplane in which they are located. Tables III - VI show the distributions of degree-0, -1, -2, and -3 zerotrees coded by SPIHT, for the 512×512 Lena at 1.0 bpp with 8 level decomposition, and Tables VII - X show the corresponding distributions for 5 levels of decomposition. Each entry indicates the number of zerotrees for the specific bitplane and zerotree height. Note that the zerotree height is equivalent to the resolution level of the location of a zerotree root. We label resolution level 0 for the highest resolution subbands. The bottom level of a zerotree corresponds to resolution level 0. In the table, resolution level 0 is not shown since a zerotree root can not located there. The higher number bitplane represents the significance map corresponding to the higher threshold. The information on bitplanes 0 and 1 is not coded at all at a rate of 1.0 bpp. For lower bitrates (i.e. < 1.0 bpp), the trends of zerotree characteristics can be expected simply by discarding the lowest bitplanes first because of the property of embedded coding.

A. The Effectiveness of a Higher Degree Zerotree Coder: Existence of Higher Degree Zerotrees

One important observation is that the occurrence of degree-2 zerotrees is as frequent as that of degree-0 and -1 zerotrees, as seen in Tables III, V, VII, and IX.

This proves the idea that a degree-2 zerotree coder is superior to a degree-1 zerotree coder, in representing a degree-2 zerotree for a 4-ary source tree (i.e. quadtree), since a degree-2 zerotree coder can directly encode a degree-2 zerotree with just one symbol, whereas a degree-1 zerotree coder would need three more symbols. From this example, where higher degree zerotree sources are found frequently, it is more effective in coding efficiency to use a higher degree zerotree coder. Meanwhile, the occurrences of degree-3 zerotrees are much less than those of degree-0, or -1 or -2 zerotrees, as shown in in Tables IV, VI, VIII, and X.

B. Location of Zerotree Root vs. Significance of Bitplane

The next observation is that the number of zerotree roots in a given subband depends on the significance level of the bitplane. The resolution level of a zerotree root is equivalent to the height of the zerotree as well. Generally, it is seen that the tall zerotrees are very few, while there are many short zerotrees. Tall and short zerotrees are depicted in Fig. 1. The zerotree roots found at higher bitplanes (such as bitplanes 12, 11, 10) are located in the lowest frequency subbands (such as resolution level 8,7,6), i.e. tall zerotrees. Meanwhile, the zerotree roots found at lower bitplanes (such as bitplanes 3,4,5) are located in the highest frequency subbands (such as resolution level 1,2,3), i.e. short zerotrees.

In summary, we make two observations. First, at higher bitplanes, all zerotrees found are tall and short zerotrees are never found. Second, at lower bitplanes, all zerotrees found are short; in other words, no zerotree root is found at lower resolution levels, such as 8, 7, and 6.

Because of the energy gain factor of two ($\sqrt{2}$ by horizontal filtering $\times$ $\sqrt{2}$ by vertical filtering) obtained after each level of wavelet decomposition (by 9/7 biorthogonal filter in these experiments), the higher bitplanes actually do not have any values defined for higher resolution subbands areas simply because those areas of the bitplanes are outside of the dynamic range of those subbands. This explains why there are no short zerotrees at higher bitplanes, such as bitplanes 12 down to 9 in Table III. There are no tall zerotrees at lower bitplanes, because the zerotrees located at lower resolution subbands (such as 8, 7, and 6) at former (higher) bitplane coding passes have been already partitioned into shorter zerotrees and finally into separate coefficients. In this case, no zerotree candidates are

TABLE III
Distribution of degree-0, -1, and -2 zerotrees in Lena coded by SPIHT, decomposition level = 8

	Height of *degree-0 or -1* zerotree (or the resolution level of zerotree root)							
bitplane	8	7	6	5	4	3	2	1
12	3	2	0	0	0	0	0	0
11	0	3	8	0	0	0	0	0
10	0	0	20	19	0	0	0	0
9	0	0	14	44	35	0	0	0
8	0	0	7	49	152	62	3	0
7	0	0	4	31	218	374	98	0
6	0	0	2	19	175	599	689	12
5	0	0	0	9	141	609	1380	678
4	0	0	0	2	89	605	1800	2442
3	0	0	0	0	16	482	2465	6885
2	0	0	0	0	0	10	78	299
1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

	Height of *degree-2* zerotree (or the resolution level of zerotree root)							
bitplane	8	7	6	5	4	3	2	1
12	0	1	0	0	0	0	0	0
11	1	3	4	0	0	0	0	0
10	0	3	13	5	0	0	0	0
9	0	0	19	34	13	0	0	0
8	0	0	7	43	86	25	1	0
7	0	0	2	18	146	232	38	0
6	0	0	2	15	119	403	451	0
5	0	0	3	7	103	480	1180	0
4	0	0	0	4	82	500	1879	0
3	0	0	0	0	36	377	2229	0
2	0	0	0	0	0	4	40	0
1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

TABLE IV
DISTRIBUTION OF DEGREE-3 ZEROTREES IN LENA CODED BY SPIHT, DECOMPOSITION LEVEL = 8

	Height of *degree-3* zerotree (or the resolution level of zerotree root)							
bitplane	8	7	6	5	4	3	2	1
12	1	0	0	0	0	0	0	0
11	0	3	0	0	0	0	0	0
10	0	7	5	0	0	0	0	0
9	0	7	17	4	0	0	0	0
8	0	4	32	38	2	0	0	0
7	0	1	20	143	39	0	0	0
6	0	2	15	142	294	7	0	0
5	0	0	14	98	514	22	0	0
4	0	0	9	99	607	37	0	0
3	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

left for further zerotree testing. Instead, the coefficients are being kept in the list of significant coefficients and only the refinement bits corresponding to current threshold are coded. Note that when a coefficient is tested as significant for a given threshold, it is moved to the list of significant coefficients, which is called SL (Subordinate List) in EZW and LSP (List of Significant Pixels) in SPIHT.

V. CONCLUSION

We have tried to establish a model of zerotrees and quantify the coding performance of zerotrees of wavelet coefficients. A *degree-k zerotree* means a tree with all zero node values except in the top k levels. And the *degree-k zerotree coder* means the source tree coder which can encode degree-i zerotrees, $0 \leq i \leq k$. Thus, the higher degree zerotree coder will have higher coding performance.

Based on this model, we classify the popular image coders EZW and SPIHT as examples, and this leads to an answer for the question as to why SPIHT is better than EZW. It is because EZW is a degree-0 zerotree coder and SPIHT is a degree-2 zerotree coder. SPIHT can encode a degree-1 or -2 zerotree with one symbol for each, while EZW will need three more symbols for each than does SPIHT. The SPIHT symbols are binary, while the EZW ones are quaternary. EZW compensates somewhat for this redundancy through adaptive arithmetic coding of the significance symbols. Even without subsequent entropy coding, SPIHT still beats EZW in overall efficiency, owing to its capability to code higher degree zerotrees. We also remark that, although experiments were conducted only on two-dimensional wavelet transforms of images, the theorems and coding formulas are valid for general t-ary branching zerotrees. So we can apply this work to one-dimensional wavelet transforms of biomedical signals ($t = 2$) and three-dimensional wavelet transforms of volume images ($t = 8$).

TABLE V
DISTRIBUTION OF DEGREE-0, -1, AND -2 ZEROTREES IN IN GOLDHILL CODED BY SPIHT, DECOMPOSITION LEVEL = 8

	Height of *degree-0 or -1* zerotree (or the resolution level of zerotree root)							
bitplane	8	7	6	5	4	3	2	1
13	4	0	0	0	0	0	0	0
12	1	0	0	0	0	0	0	0
11	0	5	0	0	0	0	0	0
10	0	11	11	0	0	0	0	0
9	0	2	40	28	0	0	0	0
8	0	2	44	142	24	0	0	0
7	0	0	16	165	345	67	0	0
6	0	0	13	109	712	843	130	0
5	0	0	3	54	592	2518	1971	21
4	0	0	0	38	332	2966	4559	67
3	0	0	0	12	31	634	3540	97
2	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

	Height of *degree-2* zerotree (or the resolution level of zerotree root)							
bitplane	8	7	6	5	4	3	2	1
13	0	0	0	0	0	0	0	0
12	3	0	0	0	0	0	0	0
11	1	4	0	0	0	0	0	0
10	0	6	8	0	0	0	0	0
9	0	7	26	11	0	0	0	0
8	0	1	32	82	12	0	0	0
7	0	1	21	154	166	25	0	0
6	0	0	2	131	448	483	9	0
5	0	0	0	25	587	1866	77	0
4	0	0	1	10	412	2765	129	0
3	0	0	0	3	29	806	35	0
2	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

REFERENCES

[1] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on Signal Processing*, vol. 41, pp. 3445–3462, 1993.

[2] A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 243–250, June 1996.

[3] Y. Cho and W. A. Pearlman, "Quantifying the coding power of zerotrees of wavelet coefficients: a degree-k zerotree model," in *2005 IEEE International Conference on Image Processing (ICIP '05)*, Sep 2005.

[4] V. N. Ramaswamy, N. Ranganathan, and K. R. Namuduri, "Peformance analysis of wavelets in embedded zerotree-based lossless image coding schemes," *IEEE Transactions on Signal Processing*, vol. 47, no. 3, pp. 884–889, March 1999.

TABLE VI

DISTRIBUTION OF DEGREE-3 ZEROTREES IN GOLDHILL CODED BY SPIHT, DECOMPOSITION LEVEL = 8

	Height of *degree-3* zerotree (or the resolution level of zerotree root)							
bitplane	8	7	6	5	4	3	2	1
13	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
11	3	0	0	0	0	0	0	0
10	1	5	1	0	0	0	0	0
9	0	9	12	1	0	0	0	0
8	0	3	38	20	0	0	0	0
7	0	0	35	111	25	0	0	0
6	0	0	10	170	305	9	0	0
5	0	0	3	93	786	74	0	0
4	0	0	0	31	587	70	0	0
3	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

TABLE VII

DISTRIBUTION OF DEGREE-0, -1, AND -2 ZEROTREES IN LENA CODED BY SPIHT, DECOMPOSITION LEVEL = 5

	Height of *degree-0 or -1* zerotree (or the resolution level of zerotree root)				
bitplane	5	4	3	2	1
11	**256**	0	0	0	0
10	247	0	0	0	0
9	174	23	0	0	0
8	99	195	29	0	0
7	50	381	304	18	0
6	18	415	953	133	0
5	6	378	1440	985	12
4	2	290	1670	2848	131
3	0	187	1942	6951	765
2	0	0	49	281	14
1	0	0	0	0	0
0	0	0	0	0	0

	Height of *degree-2* zerotree (or the resolution level of zerotree root)				
bitplane	5	4	3	2	1
11	0	0	0	0	0
10	9	0	0	0	0
9	70	13	0	0	0
8	68	101	11	0	0
7	27	255	155	6	0
6	24	262	648	63	0
5	17	274	1276	213	0
4	5	241	1781	439	0
3	1	167	1745	730	0
2	0	0	22	15	0
1	0	0	0	0	0
0	0	0	0	0	0

TABLE VIII

DISTRIBUTION OF DEGREE-3 ZEROTREES IN LENA CODED BY SPIHT, DECOMPOSITION LEVEL = 5

	Height of *degree-3* zerotree (or the resolution level of zerotree root)				
bitplane	5	4	3	2	1
11	0	0	0	0	0
10	0	0	0	0	0
9	12	0	0	0	0
8	64	22	0	0	0
7	59	148	13	0	0
6	38	301	129	0	0
5	24	340	290	0	0
4	17	392	347	0	0
3	0	143	251	0	0
2	0	0	0	0	0
1	0	0	0	0	0
0	0	0	0	0	0

TABLE IX

DISTRIBUTION OF DEGREE-0, -1, AND -2 ZEROTREES IN IN GOLDHILL CODED BY SPIHT, DECOMPOSITION LEVEL = 5

	Height of *degree-0 or -1* zerotree (or the resolution level of zerotree root)				
bitplane	5	4	3	2	1
11	256	0	0	0	0
10	254	0	0	0	0
9	218	14	0	0	0
8	125	156	3	0	0
7	40	436	209	9	0
6	11	471	1068	300	0
5	5	262	2244	2584	57
4	3	148	2341	5184	266
3	0	28	410	3320	423
2	0	0	0	0	0
1	0	0	0	0	0
0	0	0	0	0	0

	Height of zerotree (i.e. resolution level of zerotree root)				
bitplane	5	4	3	2	1
11	0	0	0	0	0
10	2	0	0	0	0
9	31	7	0	0	0
8	54	86	1	0	0
7	30	250	80	3	0
6	7	388	584	96	0
5	4	279	1749	527	0
4	6	167	2244	905	0
3	0	20	524	302	0
2	0	0	0	0	0
1	0	0	0	0	0
0	0	0	0	0	0

TABLE X
DISTRIBUTION OF DEGREE-3 ZEROTREES IN GOLDHILL CODED BY SPIHT, DECOMPOSITION LEVEL = 5

	Height of zerotree (i.e. resolution level of zerotree root)				
bitplane	5	4	3	2	1
11	0	0	0	0	0
10	0	0	0	0	0
9	7	0	0	0	0
8	70	5	0	0	0
7	73	111	11	0	0
6	32	336	140	0	0
5	3	504	445	0	0
4	0	400	527	0	0
3	0	0	0	0	0
2	0	0	0	0	0
1	0	0	0	0	0
0	0	0	0	0	0