ELHWT: Efficient Lowest to Highest Wavelet Tree for Image Processing and Compression

Mohammad Hossein Eslami, Mohsen Yakhshi Tafti

Abstract—**a new wavelet coding with ELHWT is presented. It is much more computationally efficient and uses much less internal memory than SPIHT. Image compression works on the problem of reducing the amount of data that is required to represent a digital image. This process removes redundant data from an image representation. The ELHWT algorithm is extended to the 3-D domain and a new 3-D wavelet tree based algorithm, which is much faster and uses less memory than previous 3-D wavelet tree based algorithms, is presented.**

Index Terms—**Wavelet tree, image, compression, algorithm**

I. Basic Idea of the ELHWT Algorithm

Many algorithms which are based on wavelet tree (e.g. EZW, SPIHT, and their variations) [3], [4] start encoding wavelet trees from the highest wavelet level to the lowest wavelet level. In such algorithms, a wavelet tree must be scanned multiple times to complete the coding process. And the problem is inefficiency of these multiple scans, computationally. Unlike such inefficient algorithms, Our Efficient-Lowest-to-Highest-Wavelet-Tree (ELHWT) starts encoding from the lowest wavelet level and moves through the higher wavelet levels. Coding processes finish during a single backward coding pass. This approach makes our ELHWT efficient [1], [2].

The ELHWT algorithm is very close to the algorithm in [4]. Assume $c_{i,j}$ is a wavelet coefficient at coordinate *(i, j)*. The wavelet tree structure used in the ELHWT algorithm is shown in Figure 1. It is similar to the wavelet tree structure in the algorithm in [4]. A coefficient with a coordinate of *(i, j)* in the level N subband has four offspring in the level N-1 subband. The coordinates of these four offspring are (*2i, 2j*), (*2i, 2j+1*), (*2i+1,2j*), and (*2i+1, 2j+1*).

The definitions used in the ELHWT algorithm are given as follows:

- $c_{i,j}$: The wavelet coefficient at coordinate *(i, j)*.
- $O_{i,j}$: A set of coordinates of all the offspring of *(i, j)*. (relate to the offspring of $c_{i,j}$)
- $L_{i,j}$: A set of coordinates of all the leaves of *(i, j)*. (belong to the leaves of $c_{i,j}$)
- $q_{i,j} = \begin{cases} \lfloor Ln\,|c_{i,j}|\rfloor & ,if\ |c_{i,j}| \geq 1 \\ -1 & ,otherwise \end{cases}$: The quantization level (maximum quantization threshold) of the coefficient $c_{i,j}$. ($c_{i,j}$ +1) is the number of bits required to represents its integer part of the absolute value.
- $q_{O(i,j)} = \max_{(k,l)\in O(i,j)} \{q_{k,l}\}$: The maximum quantization level of the offspring of (*i,j*).
- $q_{L(i,j)} = \max_{(k,l)\in L(i,j)} \{q_{k,l}\}$: The maximum quantization level of the leaves of (*i,j*).
- $q_{\min}$: The minimum quantization threshold. Any bits below $q_{\min}$ are not present at the output of the encoder.
- $q_{\max}$: The maximum quantization level of the entire wavelet coefficients.

Our most important step is building a map of maximum quantization levels of descendants (MQD map). Our approach utilizes the map as we want to store the maximum quantization level of WT. If an image has a size of $w \times h$, the size of the MQD map will be $w/2 \times h/2$. In the MQD map, each node $m_{i,j}$ represents the maximum quantization level of all the descendants of the wavelet coefficient $c_{i,j}$. If the wavelet transform has N levels, The MQD map will have N-1 levels. Therefore, *(i, j)* should lie in level 1 or higher levels. If *(i, j)* is in level 1, it will be the maximum quantization level of its offspring. If *(i, j)* is in level 2 or higher levels, it will be the maximum quantization level of its offspring and leaves.

This work is submitted by **Mohammad Hossein Eslami** and **Mohsen Yakhshi Tafti** to The International MultiConference of Engineers and Computer Scientists 2008. This paper is part of a research on signal processing which took place in Tehran, Iran from 2007 to 2008.

The research team was under the support of Mr. M.H. Hasanvand as advisor. Other aspects of this research will be presented in future. There are some research papers in preparation from this team.

Mr. **Mohammad Hossein Eslami** is undergraduate student of Electrical Engineering at ECE faculty, Shahid Beheshti University, Tehran, Iran. It's possible to contact him using this email address: mh_esl7@yahoo.com

Mr. **Mohsen Yakhshi Tafti**, is undergraduate student of Electrical Engineering at department of electrical engineering, Mazandaran University, Babol, Mazandaran Province, Tehran, Iran. At the moment, He is studying at Shahid Beheshti University. Please don't hesitate to cantact him if there is any question: mohsen.yt@gmail.com

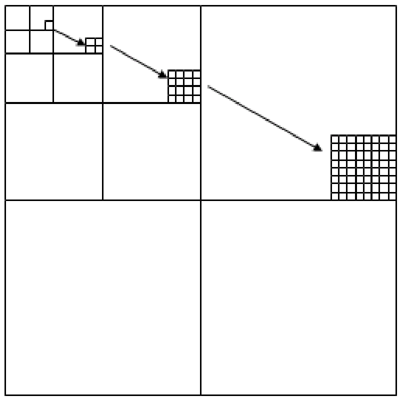


Fig. 1 Wavelet tree structure in the ELHWT algorithm.

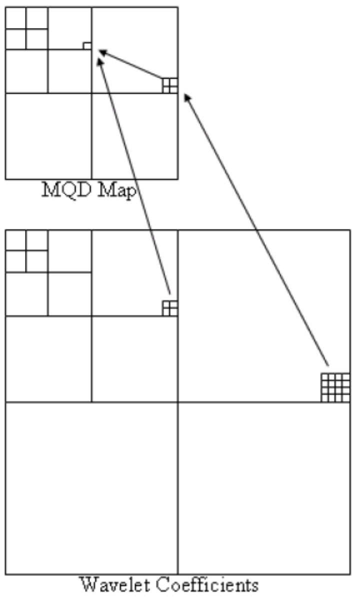


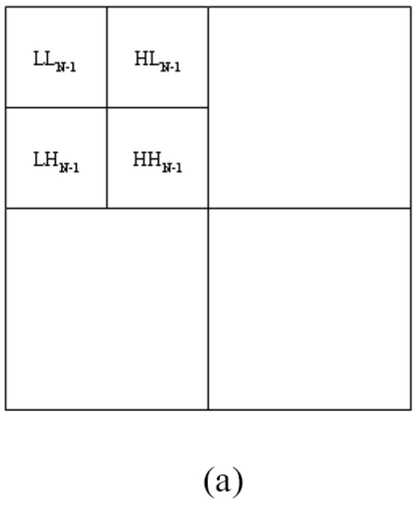
Fig. 1 Coding process of the ELHWT algorithm.

For the ELHWT algorithm, coding is achieved through the coding unit, which lies in three consecutive wavelet levels. A coding unit includes 20 wavelet coefficients and 5 MQD map nodes. Those 20 wavelet coefficients belong to the same wavelet tree. Figure 2 shows how to achieve ELHWT encoding based on the coding unit. The 25 small blocks in Figure 2 constitute a coding unit. In the wavelet coefficient domain, each small block represents a wavelet coefficient. Each small block in the MQD map represents a MQD node. The ELHWT algorithm utilizes the 2×2 block wavelet coefficients in level N and the 2×2 block MQD nodes in level N to encode the 4×4 block wavelet coefficients in level N-1 and generate a MQD node in level N+1. After encoding this coding unit, the encoding results of the 4×4 block wavelet coefficients in level N-1 are output, and the newly-generated MQD node is written to the MQD map. The lowest level MQD nodes are generated based on the level 0 wavelet coefficients. The higher level MQD nodes are generated during the encoding process. We can discard the lowest level MQD nodes after using them. Therefore the size of MQD map kept in the memory is ${}^{w}\!/_{4}\times{}^{h}\!/_{4}$.

II. ELHWT ALGORITHM

Notations of the complete ELHWT algorithm:

- $B(x)$: The binary code of $\lfloor |x| \rfloor$, e.g., $B(5.5) = 101$.
- $T(n)$: A binary code with a single one at the nth right-most-bit ($n \geq 0$), e.g., $T(0)$=00000001; $T(4)$=00010000.
- $B\Big|_{n}^{m}$: A section of binary code b, starting from the nth and ending at the mth right-most-bit $(m \geq n \geq 0)$ e.g., $00000100\Big|_{1}^{4} = 0010$; $00110110\Big|_{2}^{6} = 01101$



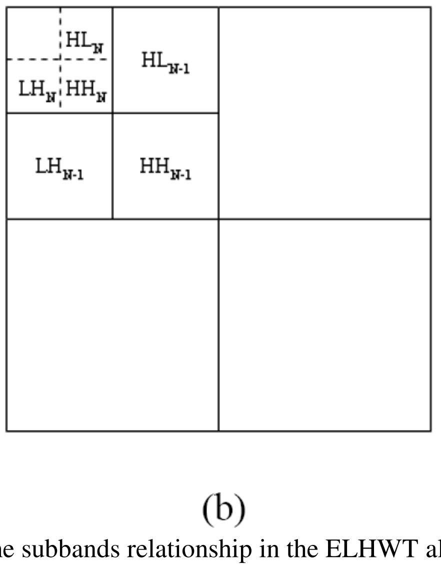


Fig. 2 The subbands relationship in the ELHWT algorithm. (a) N level wavelet decomposition. (b) Further partition the LLN-1 subband.

In order to effectively describe the coding steps, we give some other notations. As shown in [5], [6], [7], [8], for a N-level 2-D wavelet decomposition, there should be 3N+1 subbands. They are HL_0 , LH_0 , HH_0 , HL_1 , LH_1 , HH_1 , …, HL_{N-1} , LH_{N-1} , HH_{N-1} , and

LL_{N-1} subbands. Figure 3(a) marks the HL_{N-1}, LH_{N-1}, HH_{N-1}, and LL_{N-1} subbands in wavelet decomposition. In the ELHWT algorithm, we further divide LL_{N-1} subbands into four small subbands, as shown in Figure 3(b). The HLN, LHN and HHN subbands in the LL_{N-1} subband will be utilized to encode the HL_{N-1}, LH_{N-1} and HH_{N-1} subbbands. We use the notation S_N to represents the entirety of the three high frequency subbands at level N. Therefore, S_0 represents HL_0, LH_0 and HH_0 subbands. S_N represents HL_N, LH_N and HH_N subbands. The complete ELHWT encoding steps are given as follows:

A. Encode level 0 to level N-2 high frequency subbands:

```
**A.1.** $\forall(i,j)\in S_n\,(n=2,3,4,\ldots,N)$;
If n=2 , $\forall(k,l)\in O(i,j): m_{k,l} = \max_{(u,v)\in O(k,l)}\{q_{u,v}\}$
$q_{L(i,j)} = \max_{(k,l)\in O(i,j)}\{m_{k,l}\}$
If $q_{L(i,j)} \ge q_{\min}$, $\forall(k,l)\in O(i,j)$:
    If $m_{k,l} \ge q_{\min}$, $\forall(k,v)\in O(k,l)$:
        If $q_{u,v} \ge q_{\min}$, output $sign(c_{u,v})$
        Output $B(|c_{u,v}|)\Big|_{q_{\min}}^{m_{k,l}}$
    Output $T(m_{k,l})\Big|_{\max(m_{k,l},q_{\min})}^{q_{L(i,j)}}$
$m_{i,j} = \max\left\{\max_{(k,l)\in O(i,j)}\{q_{k,l}\},q_{L(i,j)}\right\}$
If $m_{i,j} \ge q_{\min}$, output $T(q_{L(i,j)})\Big|_{\max(q_{L(i,j)},q_{\min})}^{m_{i,j}}$
**A.2** n=n+1. If $n \le N$, go to step A.1
```

Step A is used to encode the coefficients in S_0 to S_{N-2}. Step A.1 describes the process of encoding a coding unit. For each coding unit, we encode the sixteen (4×4 block) wavelet coefficients. After that, we move to the next coding units. After all the coefficients in S_x are encoded, we move to S_{x+1} and encode the coefficients in S_{x+1}.

B. Encode level N-1 high frequency subbands:

```
**B.1** $q_{\max} = \max\left\{\max_{(i,j)\in LL_{N-1}}(q_{i,j}),\max_{(i,j)\in S_N}(m_{i,j})\right\}$
**B.2** $\forall(i,j)\in S_N$:
If $m_{i,j} \ge q_{\min}$, $\forall(u,v)\in O(i,j)$
    If $q_{u,v} \ge q_{\min}$, output $sign(c_{u,v})$
    Output $B(|c_{i,j}|)\Big|_{q_{\min}}^{m_{i,j}}$
Output $T(m_{i,j})\Big|_{\max(m_{i,j},q_{\min})}^{q_{\max}}$
```

It shows how to encode the coefficients in S_{N-1}. Here, we utilize S_N to encode the coefficients in S_{N-1}.

C. Encode the low frequency subband :

```
$\forall(i,j)\in LL_{N-1}$:
If $q_{i,j} \ge q_{\min}$, output $sign(c_{i,j})$
Output $B(|c_{i,j}|)\Big|_{q_{\min}}^{q_{\max}}$
```

After all the coefficients in S_{N-1} are encoded, we move to the LL_{N-1} subband. We use uniform quantization to encode the coefficients in the LL_{N-1} subband, which is shown in this step.

III. Advantages of the ELHWT Algorithm

From above coding steps, it can be found that the ELHWT algorithm is much more efficient than the algorithm in [4]. No tree-scanning, bitplane coding, or management of dynamical lists occurs in the ELHWT algorithm. All of the encoded bits for a coefficient are output during a one-pass coding. As a result, ELHWT is much faster than the SPIHT algorithm.

The traditional algorithm in [4] requires very large internal memory usage. During that coding, three separate lists must be maintained. Often, the memory required to keep the lists is larger than memory to store the wavelet coefficients. Also, the lengths of these lists are image-dependent. Here, we choose a moderate case to calculate the memory usage of the algorithm in [4]. The memory requirement for that can be calculated in terms of bit-per-coefficient with the following equation:

$$\frac{L_{\max}(LIS)+L_{\max}(LIP)+L_{\max}(LSP)}{RC}\big(Ln(R)+Ln(C)\big) \quad \textbf{(1)}$$

As you can see, in Equation 1, number of rows and columns are represented by *R* and *C*. $L_{\max}(.)$ represents the maximum length of a list. Let's consider a moderate case. If we store a floating-point wavelet coefficient with 16 bits (2 bytes), *R* and *C* are 512, and each list has the maximum length of RC/3, so that 18 bits/coefficient are required to store the lists. In this case, the memory usage for the lists is 112.5% of the memory usage for wavelet coefficients.

However, in the ELHWT algorithm, we do not need to store any list. The only memory consumed is the MQD map. The MQD node in the MQD map is a maximum quantization level. Because each wavelet coefficient is stored in 16 bits, for each MQD node, we just need $Ln(16)=4$ bits to represent it. Considering that the MQD map kept in memory is 1/16 the size of the image, we can get that the memory usage for the MQD map is 1/4 bit per coefficient. Compared with the 18 bits/coefficient in the SPIHT algorithm, the ELHWT algorithm uses 72 times less internal memory than the SPIHT algorithm.

In this paper, the working mechanism and the complete coding steps of the ELHWT algorithm are given.

IV. CONCLUSION

In short our paper is based on comparison of our ELHWT and other algorithms (e.g. SPIHT)

Our algorithm, presented in this paper, extended the recently-developed algorithms. The ELHWT algorithm begins to encode the wavelet trees from the lowest wavelet level and moves back through the higher wavelet levels. Experimental results show that the ELHWT is much faster than the well-known SPIHT. The internal memory usage of ELHWT is much less than the internal memory usage of SPIHT. The BCWT and SPIHT algorithm have almost identical PSNR performance, but the overall system memory usage of the BCWT algorithm is significantly less than the SPIHT.

REFERENCES

[1] J. Guo, S. Mitra, B. Nutter, and T. Karp, "A Fast and Low Complexity Image Codec based on Backward Coding of Wavelet Trees," Proceedings of Data Compression Conference, Snowbird, Utah, March, 2006.

[2] J. Guo, "A Hybrid Vector Scalar Quantization Based on Backward Coding of Wavelet Trees," Ph.D. Dissertation, Texas Technology University, December 2005.

[3] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," IEEE Transaction on Signal Processing, volume 41, pages 3445-3459, December 1993.

[4] A. Said and W. A. Pearlman, "A New Fast and Efficient Image Codec based on Set Partitioning in Hierarchical Trees," IEEE Transaction on Circuits and Systems for Video Technology, volume 6, pages 243-250, 1996.

[5] E. Moyano, F. J. Quiles, A. Garrido, L. Orozco-Barbosa, and J. Duato, "Efficient 3-D Wavelet Transform Decomposition for Video Compression," International Workshop on Digital and Computational Video, February, 2001.

[6] S. Mallat, "Multifrequency Channel Decompositions of Images and Wavelet Models," IEEE Transaction Acoustic Speech Signal Processing, volume 37, no. 12, pages 2091-2110, 1989.

[7] L. Ye, "Codebook Ordering for Vector Quantization," Master's Thesis, Texas Tech University, December 2003.

[8] R. C. Gonzalez and R. E. Woods, Digital Image Processing, Pearson Education, Inc., 2002.