ACCELERATED ANALYSIS OF QUAD-TREES IN THE EZW ALGORITHM

Deividas Kancelkis, Jonas Valantinas

Kaunas University of Technology, Department of Applied Mathematics, Studentu St 50-325c, LT-51368 Kaunas, Lithuania, e-mail, deividas.kancelkis@stud.ktu.lt, jonas.valantinas@ktu.lt

Abstract. In this paper, a novel scheme for the analysis of quad-trees in the discrete wavelet spectrum of a digital image is proposed. The scheme can be successfully applied to any zero-tree based image encoder, such as the embedded zero-tree wavelet (EZW) algorithm of Shapiro and set partitioning in hierarchical trees (SPIHT) by Said and Pearlman. Exceptionally high performance of the proposed quad-tree analysis scheme, in the sense of image encoding times, is demonstrated using the EZW algorithm and the discrete Le Gall wavelet transform.

Keywords: discrete wavelets, wavelet transforms, quad-trees, zero-tree based image coding.

1 Introduction

Nowadays most modern digital image compression techniques employ a discrete wavelet transform, usually followed by quantization and entropy coding, [1-3]. Especially useful are image coders that allow progressive encoding with an embedded bit stream, such as the embedded zero-tree wavelet (EZW) image coder, suggested by Shapiro, [4]. With embedded bit streams, the wavelet coefficients are encoded in bit planes, with the most significant bit planes being transmitted first. In that way, the decoder can stop decoding at any point in the bit stream, and it will reconstruct an image with required level of accuracy.

There have been many variants of zero-tree-based progressive image coders since Shapiro introduced his algorithm in 1993. The SPIHT (Set Partitioning in Hierarchical Trees) algorithm, proposed by Said and Pearlman, shows excellent results in this class of coders, [5]. Some other interesting ideas and developments in the area are presented in [6-11].

Bit plane encoding is more efficient if one reorders the wavelet coefficient data in such a way that coefficients with small absolute values tend to get clustered together, increasing the lengths of the zero run in the bit planes. Data structures such as significance quad-trees (zero-trees) are very efficient in achieving such clustering of zeros. They are used in EZW, SPIHT, and other wavelet-based image coders.

Though there is a number of wavelet-based image coding schemes available, the need for improved performance and wide commercial usage demand newer and better techniques to be developed.

In this paper, we propose an original idea (scheme) for accelerated analysis of quad-trees in the discrete wavelet spectrum of the image under compression. The proposed scheme, being applied to EZW encoder, impressively improves image encoding times (45–55%), for lossless compression, and (40–90%), for lossy compression; Section 3.2) and, naturally, the overall performance of the encoder.

2 The embedded zero-tree wavelet (EZW) algorithm

The embedded zero-tree wavelet algorithm (EZW) is one of the most efficient coding schemes, which are developed for wavelets. The EZW transmits the large (significant) wavelet coefficients before transmitting the smaller coefficients. This is done by realizing multiple passes (iterations, scanning) over the discrete wavelet spectrum of the image, lowering the threshold by a factor of two each time, [4].

On each iteration the EZW codec refines the value of each coefficient determined to be significant during previous passes and then searches through the coefficients previously considered to be insignificant. A wavelet coefficient is said to be insignificant with respect to a given threshold $T$ if it is less (by absolute value) than $T$. The zero-tree is based on the hypothesis that if a particular wavelet coefficient (the parent) at a high scale is insignificant with respect to a given threshold $T$, then all wavelet coefficients (the descendants) of the same orientation in the same spatial location at lower scales are probably to be insignificant with respect to $T$ (Figure 1, a). If this is true, we say that this coefficient is the root of the zero-tree, [4].

Below, we briefly present both the encoding and decoding phases of the current EZW algorithm, [4, 6]. Let RL (Refinement list) be the list of wavelet coefficients which have been previously found to be significant and let IL (Index list) be the list of indices of all wavelet coefficients which have not yet been found to be significant, ordered so that parents are listed before their children (say, in accordance with Morton’s trajectory; Figure 1, b).
The image encoding phase is as follows.

1. Compute \( r = r_{\text{max}} = \left\lfloor \log_2 \max \{ |Y(k_1, k_2)| \} \right\rfloor \), where \( Y(k_1, k_2) \) is the \((k_1, k_2)\)-th wavelet coefficient of the digital image \([X(m_1, m_2)]\) under processing; \( k_1, k_2, m_1, m_2 \in \{0, 1, \ldots, N-1\}, \ N = 2^n \), \( n \in \mathbb{N} \). Set \( RL = \emptyset \).

2. For each \( Y(k_1, k_2) \in RL \), output the \( r \)-th most significant bit of \( Y(k_1, k_2) \).

3. Set the list \( IL \) to contain indices of all the wavelet coefficients, except of those in \( RL \), ordered so that parents are listed before their children. For each index \((k_1, k_2) \in IL\), do: if \( Y(k_1, k_2) \geq T_r \) \((T_r = 2^r)\) then output \( \text{poz} \) and add \( Y(k_1, k_2) \) to \( RL \), else if \( Y(k_1, k_2) \leq -T_r \) then output \( \text{neg} \) and add \( Y(k_1, k_2) \) to \( RL \), else if no descendant (not contained in \( RL \)) of \( Y(k_1, k_2) \) is significant with respect to \( T_r \) then output \( \text{zrt} \) and remove all the indices, corresponding to the descendants of \( Y(k_1, k_2) \), from \( IL \), else output \( \text{iz} \).

4. If \( r > 0 \) then decrease the value of \( r \) by one and go to the step 2.

5. The end.

Since symbols \( \text{poz} \), \( \text{neg} \), \( \text{zrt} \), \( \text{iz} \) do not occur with the same probability Huffman coding is usually applied to the EZW output to reduce data amount (bit stream).

The image decoding phase requires some data to start with, namely: the initial threshold \( T_r = T_{\text{max}} = 2^{r_{\text{max}}} \), the original image size, the sub-band decomposition scale and the encoded bit stream. The EZW decodes encoded files (bit stream) into a symbol sequence, creates all the proper size subbands needed and proceeds image decompression process as follows.

1. Input \( r = r_{\text{max}} \). Set \( RL = \emptyset \) and set all the wavelet coefficients to zeros.

2. For each \( Y(k_1, k_2) \in RL \), input the \( r \)-th most significant bit of \( |Y(k_1, k_2)| \).

3. Set the list \( IL \) to contain indices of all the coefficients, except of those in \( RL \), ordered so that parents are listed before their children. For each index \((k_1, k_2) \in IL\), do: input a current symbol \( \sigma \); if \( \sigma = \text{poz} \) then set \( Y(k_1, k_2) = T_r \) \((T_r = 2^r)\) and add \( Y(k_1, k_2) \) to \( RL \), else if \( \sigma = \text{neg} \) then set \( Y(k_1, k_2) = -T_r \) and add \( Y(k_1, k_2) \) to \( RL \), else if \( \sigma = \text{zer} \) then remove all indices, corresponding to the descendants of \( Y(k_1, k_2) \) from \( IL \), else if \( \sigma = \text{iz} \) then take no action.

4. If \( r > 0 \) then decrease the value of \( r \) by one and go to the step 2.

5. The end.

The other wavelet-based image coding algorithm, called SPIHT, is an improved version of EZW which achieves better compression and performance than EZW. For more detailed description of the algorithm, we refer the reader to [5].
3 A novel approach to quad-tree analysis in the EZW algorithm

The most time-consuming part of the EZW algorithm is the iterative image encoding phase (Section 2), because hereupon all quad-trees in the discrete wavelet spectrum \([Y(k_1, k_2)]\) of the image \([X(m_1, m_2)]\) are checked repeatedly for significant wavelet coefficients with respect to a certain (decreasing from iteration to iteration) threshold. When an insignificant wavelet coefficient (root of a quad-tree) \(Y(k_1, k_2)\) is found, and a scan of its children reveals that they are insignificant too, then it is possible to encode that wavelet coefficient \(Y(k_1, k_2)\) and its children, a zero-tree, by one symbol \(zrt\), thus achieving compression. Wavelet coefficients (roots of quad-trees) found to be insignificant in the encoding phase but with significant children are coded as isolated zeros \(iz\).

To avoid repeated scanning and verification of wavelet coefficients, comprising volumes of quad-trees in \([Y(k_1, k_2)]\), for significance with respect to changing threshold values, we have developed a novel accelerated quad-tree analysis scheme, which guarantees exceptionally high performance of the encoding phase in EZW.

3.1 An improved quad-tree analysis scheme

Let \([X(m_1, m_2)]\) \((m_1, m_2 \in [0,1,\ldots,N-1], N = 2^n, n \in N\) be a given digital image and \([Y(k_1, k_2)]\) \((k_1, k_2 \in [0,1,\ldots,N-1])\) be its discrete wavelet (Haar, Le Gall, Daubechies, etc.) spectrum. Also, let

\[ r_{\text{max}} = \left\lfloor \log_2 \max \{ |Y(k_1, k_2)| \mid k_1, k_2 \in [0,1,\ldots,N-1] \} \right\rfloor. \]

Consider a wavelet coefficient \(Y(k_1, k_2)\) \((k_1 = 2^{s-1} j_i, j_i \in \{0, 1, \ldots, 2^{n-1} - 1\}, s = 1, 2\) which is the root (parent) of the quad-tree (provided \(i_1 > 1\) and \(i_2 > 1\)), comprising the set of wavelet coefficients (descendants)

\[ \{Y(k_1', k_2') \mid (k_1', k_2') \in \mathcal{S}\}, \]

where \(\mathcal{S} = \bigcup_{i=1}^{\min\{i_1,i_2\}-1} \{\mathcal{S}_y(t)\times\mathcal{S}_y(t)\}, \mathcal{S}_y(t) = \{2^t \cdot k_1, 2^t \cdot k_1 + 1, \ldots, 2^{t}(k_1 + 1) - 1\} \text{ and } s = 1, 2.\)

Let us associate \(Y(k_1, k_2)\) with two binary codes (one for the offspring of \(Y(k_1, k_2)\), another for the descendants of \((k_1, k_2)\), except offspring), namely:

\[ \text{CodeOff} (k_1, k_2) = (u_{\text{max}}(k_1, k_2) \ldots u_1(k_1, k_2) u_0(k_1, k_2)) , \]

\[ \text{CodeDes} (k_1, k_2) = (v_{\text{max}}(k_1, k_2) \ldots v_1(k_1, k_2) v_0(k_1, k_2)) . \]

The above codes are generated by a single scanning of the discrete wavelet spectrum \([Y(k_1, k_2)]\) as follows:

1. \(u_r(k_1, k_2) = 1\), if at least one of coefficients (taken by absolute value) \(|Y_{2^{k_1}2^{k_2}}|, |Y_{2^{k_1}2^{k_2}+1}|\) falls into the half-open interval \([2^r, 2^{r+1})\), \(r \in \{0,1,\ldots,r_{\text{max}}\}\), and \(u_r(k_1, k_2) = 0\), otherwise;

\[
u_r(k_1, k_2) = \begin{cases} u_r(2k_1, 2k_2) \lor u_r(2k_1, 2k_2 + 1) \lor u_r(2k_1 + 1, 2k_2 + 1), & \text{for } N/8 \leq \max\{k_1, k_2\} \leq N/4 - 1, \\ u_r(2k_1, 2k_2) \lor u_r(2k_1, 2k_2 + 1) \lor u_r(2k_1 + 1, 2k_2 + 1), & \text{for } 1 \leq \max\{k_1, k_2\} \leq N/8 - 1, \end{cases} \]

for all \(r = 0, 1, \ldots, r_{\text{max}}\).

We here emphasize that in the first instance one must compute binary codes \(\text{CodeDes}(k_1, k_2)\) with index pairs \((k_1, k_2)\) satisfying condition \(N/8 \leq \max\{k_1, k_2\} \leq N/4 - 1\), then codes \(\text{CodeDes}(k_1, k_2)\) with index pairs \((k_1, k_2)\) satisfying condition \(N/16 \leq \max\{k_1, k_2\} \leq N/8 - 1\), and, finally, codes \(\text{CodeDes}(k_1, k_2)\) with index pairs \((k_1, k_2)\) satisfying condition \(\max\{k_1, k_2\} = 1\).

Thus, to state that the wavelet coefficient \(Y(k_1, k_2)\) \((k_1, k_2 \in [0,1,\ldots,N/2-1])\) is the root of the zero-tree, comprising the set of wavelet coefficients (descendants) \(\{Y(k_1', k_2') \mid (k_1', k_2') \in \mathcal{S}\}, \) with respect to the threshold \(T = T_r = 2^r\) \((r \in \{0,1,\ldots,r_{\text{max}}\}\), it suffices to ascertain that \(u_r(k_1, k_2) = 0\) and \(v_r(k_1, k_2) = 0\) (the key moment of the proposed idea).
We observe that generation of binary codes $\text{CodeOff}(k_1, k_2)$, for all possible values of $k_1$ and $k_2$, requires $(N^2 - 4)$ verifications of falling into one or another half-open interval, while that for $\text{CodeDes}(k_1, k_2)$ requires $(N^2/4 - 7)$ logical additions on binary codes of length $r_{\text{max}} + 1$.

Evidently, the described quad-tree analysis scheme can be used with other coding algorithms similar to the EZW, as the SPIHT algorithm (in processing lists of insignificant pixels and those of insignificant sets, [5]), and for other data as images.

3.2 Experimental results

To implement both the current and the modified (supplemented with the proposed quad-tree analysis scheme; Section 3.1) versions of the EZW algorithm, the discrete Le Gall (wavelet) transform (DLGT) was employed. The latter transform possesses a tolerable “energy compaction” property, has a fast performing technique and facilitates lossless compression of digital images. Incidentally, the default reversible transform in JPEG 2000 is implemented exactly by means of DLGT, [3].

With a view to estimate efficiency of the developed accelerated quad-tree analysis scheme, a number of digital images of size $256 \times 256$, characterized by different smoothness level, were processed, namely (Figure 2): Acura.bmp, Lena.bmp, Forest.bmp. Computer simulation was performed on a PC with CPU PENTIUM4 2.8 GHz, RAM 2 GB, OS Windows XP; Programming language Java.

![Figure 2. Test images: (a) image “Acura” 256x256; (b) image “Lena” 256x256; (c) image “Forest” 256x256](image)

As it can be seen from Table 1, application of the proposed accelerated quad-tree analysis scheme to lossless encoding (the threshold value $T = T_o = 1$), as well as to lossy encoding ($T = T_o > 1$), of test images leads to striking image encoding time gains $\tau_{\text{EZW}} - \tau'_{\text{EZW}}$ ($\tau_{\text{EZW}}$ signifies the time needed to encode the test image using the basic version of the EZW algorithm, whereas $\tau'_{\text{EZW}}$ - the time needed to encode the same image using the developed modified version of the EZW algorithm).

<table>
<thead>
<tr>
<th>Threshold Test image</th>
<th>$T = T_o = 1$</th>
<th>$T = T_1 = 2$</th>
<th>$T = T_2 = 4$</th>
<th>$T = T_3 = 8$</th>
<th>$T = T_o = 16$</th>
<th>$T = T_o = 32$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acura.bmp</td>
<td>$\tau_{\text{EZW}}$</td>
<td>25.435</td>
<td>7.705</td>
<td>4.23</td>
<td>3.118</td>
<td>2.574</td>
</tr>
<tr>
<td></td>
<td>$\tau'_{\text{EZW}}$</td>
<td>11.95</td>
<td>3.18</td>
<td>1.175</td>
<td>0.526</td>
<td>0.199</td>
</tr>
<tr>
<td>Lena.bmp</td>
<td>$\tau_{\text{EZW}}$</td>
<td>57.159</td>
<td>37.567</td>
<td>20.274</td>
<td>7.425</td>
<td>2.747</td>
</tr>
<tr>
<td></td>
<td>$\tau'_{\text{EZW}}$</td>
<td>29.919</td>
<td>19.928</td>
<td>10.461</td>
<td>4.02</td>
<td>1.287</td>
</tr>
<tr>
<td>Forest.bmp</td>
<td>$\tau_{\text{EZW}}$</td>
<td>159.911</td>
<td>118.142</td>
<td>85.508</td>
<td>57.358</td>
<td>42.308</td>
</tr>
<tr>
<td></td>
<td>$\tau'_{\text{EZW}}$</td>
<td>87.86</td>
<td>67.107</td>
<td>47.25</td>
<td>32.94</td>
<td>24.036</td>
</tr>
</tbody>
</table>

For instance, in the case of lossless image encoding ($T = T_o = 1$), the modified version of the EZW algorithm runs (on the average) twice faster than the current EZW algorithm. Furthermore, with increasing values of the threshold $T$, the obtainable fulfilling image encoding speed gains $\omega = (\tau_{\text{EZW}} - \tau'_{\text{EZW}})/\tau_{\text{EZW}} \cdot 100$ (%) have tendency to increase (Figure 3).

- 305 -
Similar promising results were obtained for images of other sizes. For instance, the encoding time speed gains for the image `peppers.bmp` of size 512x512 equal \( \omega = \frac{(3476.655 - 761.8)}{3476.655 - 100} = 78.088 \) (%), and that for the image `brain.bmp` of size 128x128 equal \( \omega = \frac{(5.472 - 3.441)}{5.472 - 100} = 37.116 \) (%). Thus, in all cases, the modified version of the EZW algorithm performs better than the basic EZW algorithm.

Worth emphasizing, the overall performance of both the current and the modified EZW coders depends on the smoothness class (level) of the image under processing, [12]. The higher smoothness of the image, the lower image encoding times and noticeably better performance of the modified EZW coder in comparison with the current EZW coder (Figure 4). We here notice that the smoothness level \( \alpha \) of test images was determined by computing the rate of “decay” of Le Gall wavelet coefficients.

**Conclusion**

In the paper, a novel scheme for the accelerated analysis of quad-trees in the discrete wavelet spectrum of a digital image is proposed. The scheme can be applied to any zero-tree based image encoder, such as the embedded zero-tree wavelet (EZW) algorithm of Shapiro, set partitioning in hierarchical trees (SPIHT) by Said and Pearlman, and others.

Numerous experimental results show that implementation of the proposed quad-tree analysis scheme in the EZW algorithm increases the overall performance of the EZW encoder somewhere\( (40–90) \)%. Worth emphasizing, the image encoding speed gains directly depend on the smoothness class of the image under processing. The lower smoothness of the image, the better image encoding speed gains are obtained.

In the future, efficiency analysis of the application of the developed quad-tree analysis scheme to other wavelet-based image encoders (SPIHT, EBCOT, etc.) is planned.
References


