International Journal of Engineering Inventions
e-ISSN: 2278-7461, p-ISSN: 2319-6491
Volume 3, Issue 5 (December 2013) PP: 11-15

Image compression using EZW encoding

Swati Ollalwar
Assistant Professor, MVJ College of Engineering, Near ITPB, Whitefield, Bangalore-560 067

Abstract: *With the increase in the demand for remote area applications the need for accurate and high-speed data transmission is increasing. The available resources are getting constrained for such a service requirements. One such advanced service which demands both accuracy with speed of operation is telemedicine applications. In such application the medical images could be forwarded through wired or wireless network for remote monitoring. To improve the performance of such system JPEG committee have come out with higher resolution compression architecture called JPEG2000.*
The JPEG2000 coding system uses wavelet transform which decomposes the image into different levels where the coefficient in each sub band are uncorrelated from coefficient other sub bands as a result the coefficient in each sub band can be quantized independently of coefficient in other sub band with no significant loss in performance, but the coefficient in each sub band requires different amount of bit resources to obtain best coding performance [8]. A hierarchical coding algorithm called Embedded Zero tree wavelet coding is proposed which exploits the multi-resolution properties of the wavelet transform to give a computationally simple algorithm with better performance compared to existing wavelet transform[6]. This coding finds the co relational properties of each band and eliminate the coefficients from each band as per their significance.
Keywords: *DCT, JPEG, EZW, Zerotree*

I. EZW encoding

An EZW encoder is an encoder specially designed to use with *wavelet transforms*, which explains why it has the word wavelet in its name. The EZW encoder was originally designed to operate on images (2D-signals) but it can also be used on other dimensional signals.The EZW encoder is based on *progressive encoding* to compress an image into a bitstream with increasing accuracy.

This means that when more bits are added to the stream, the decoded image will contain more detail, a property similar to JPEG encoded images. It is also similar to the representation of a number like π. Every digit we add increases the accuracy of the number, but we can stop at any accuracy we like. Progressive encoding is also known as *embedded encoding*, which explains the E in EZW.
This leaves us with the Z. This letter is a bit more complicated to explain, but I will give it a try in the next paragraph.

Coding an image using the EZW scheme, together with some optimizations results in a remarkably effective image compressor with the property that the compressed data stream can have *any* bit rate desired. *Any* bit rate is only possible if there is information loss somewhere so that the compressor is *lossy*. However, lossless compression is also possible with an EZW encoder, but of course with less spectacular results.

II. Introduction

In this tutorial I will try to explain the implementation of an Embedded Zerotree Wavelet encoder or EZW encoder. The reason for this tutorial is that I have never come across a good explanation of this technique, yet many researchers claim that they have implemented it. Of course there is Shapiro's original paper, but when reading it carefully many details are not immediately clear or even missing. Since I think that the approach of EZW encoding is a fruitful one. I have decided to present the details here. This might save others some work in the future.
This expects some understanding of wavelet transforms.

III. The Zero Tree

The EZW encoder is based on two important observations:
1. Natural images in general have a low pass spectrum. When an image is wavelet transformed the energy in the subbands decreases as the scale decreases (low scale means high resolution), so the wavelet coefficients will, on average, be smaller in the higher subbands than in the lower subbands. This shows that progressive encoding is a very natural choice for compressing wavelet transformed images, since the higher subbands only add detail.
2. Large wavelet coefficients are more important than smaller wavelet coefficients.

These two observations are exploited by the EZW encoding scheme by coding the coefficients in decreasing order, in several passes. For every pass a threshold is chosen against which all the coefficients are measured. If a wavelet coefficient is larger than the threshold it is encoded and removed from the image, if it is smaller it is left for the next pass. When all the wavelet coefficients have been visited the threshold is lowered and the image is scanned again to add more detail to the already encoded image. This process is repeated until all the wavelet coefficients have been encoded completely or another criterion has been satisfied (maximum bit rate for instance).

The trick is now to use the dependency between the wavelet coefficients across different scales to efficiently encode large parts of the image which are below the current threshold. It is here where the *zerotree* enters. So let me now add some detail to theforegoing. (As most explanations, this explanation is a progressive one.)

A wavelet transform transforms a signal from the time domain to the joint time-scale domain. This means that the wavelet coefficients are two-dimensional. If we want to compress the transformed signal we have to code not only the coefficient values, but also their position in time. When the signal is an image then the position in time is better expressed as the position in space. After wavelet transforming an image we can represent it using trees because of the subsampling that is performed in the transform. A coefficient in a low subband can be thought of as having four descendants in the next higher subband (see figure 1). The four descendants each also have four descendants in the next higher subband and we see a quad-tree emerge: every root has four leafs.

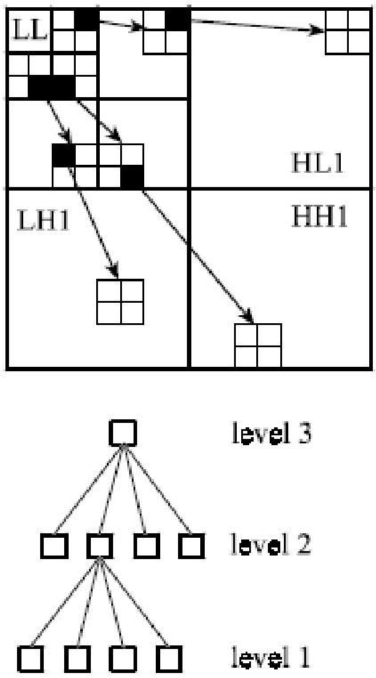

Figure 1: The relations between wavelet coefficients in different sub bands as quad-trees.

We can now give a definition of the zerotree. A zerotree is a quad-tree of which all nodes are equal to or smaller than the root. The tree is coded with a single symbol and reconstructed by the decoder as a quad-tree filled with zeroes. To clutter this definition we have to add that the root has to be smaller than the threshold against which the wavelet coefficients are currently being measured.

The EZW encoder exploits the zerotree based on the observation that wavelet coefficients decrease with scale. It assumes that there will be a very high probability that all the coefficients in a quad tree will be smaller than a certain threshold if the root is smaller than this threshold. If this is the case then the whole tree can be coded with a single zerotree symbol. Now if the image is scanned in a predefined order, going from high scale to low, implicitly many positions are coded through the use of zerotree symbols. Of course the zerotree rule will be violated often, but as it turns out in practice, the probability is still very high in general. The price to pay is the addition of the zerotree symbol to our code alphabet.

IV. How does it work?

Now that we have all the terms defined we can start compressing. Let's begin with the encoding of the coefficients in decreasing order.

A very direct approach is to simply transmit the values of the coefficients in decreasing order, but this is not very efficient. This way a lot of bits are spend on the coefficient values and we do not use the fact that we know that the coefficients are in decreasing order. A better approach is to use a threshold and only signal to the decoder if the values are larger or smaller than the threshold. If we also transmit the threshold to the decoder, it can reconstruct already quite a lot. To arrive at a perfect reconstruction we repeat the process after lowering the threshold, until the threshold has become smaller than the smallest coefficient we wanted to transmit. We can make this process much more efficient by subtracting the threshold from the values that were larger than the threshold. This results in a bit stream with increasing accuracy and which can be perfectly reconstructed by the decoder. If we use a predetermined sequence of thresholds then we do not have to transmit them to the decoder and thus save us some bandwidth. If the predetermined sequence is a sequence of powers of two it is called bitplane coding since the thresholds in this case correspond to the bits in the binary representation of the coefficients. EZW encoding as described in [3] uses this type of coefficient value encoding.

One important thing is however still missing: the transmission of the coefficient positions. Indeed, without this information the decoder will not be able to reconstruct the encoded signal (although it can perfectly reconstruct the transmitted bit stream). It is in the encoding of the positions where the efficient encoders are separated from the inefficient ones. As mentioned before, EZW encoding uses a predefined scan order to encode the position of the wavelet coefficients (see figure 2). Through the use of zero trees many positions are encoded implicitly. Several scan orders are possible (see figure 3), as long as the lower sub bands are completely scanned before going on to the higher sub bands. In [3] a raster scan order is used, while in [1] some other scan orders are mentioned. The scan order seems to be of some influence of the final compression result.

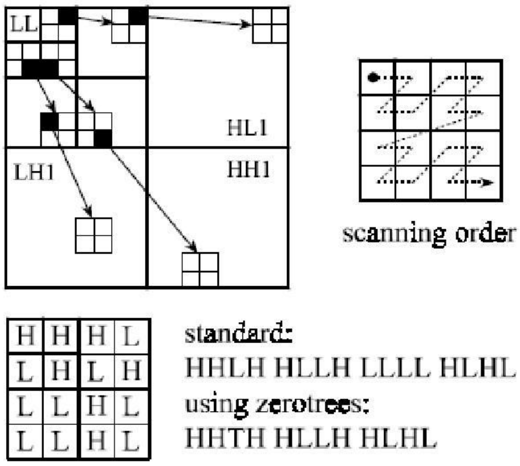

Figure 2: The relations between wavelet coefficientsin different subbands (left), how to scan them (upper right) and the result of using zerotree (lower right) symbols (T) in the coding process. An H means that the coefficient is higher than the threshold and an L means that it is below the threshold. The zerotree symbol (T) replaces the four L's in the lower left part and the L in the upper left part.

Now that we know how the EZW scheme codes coefficient values and positions we can go on to the algorithm.

V. The algorithm

The EZW output stream will have to start with some information to synchronize the decoder. The minimum information required by the decoder is the number of wavelet transform levels used and the initial threshold, if we assume that always the same wavelet transform will be used. Additionally we can send the image dimensions and the image mean. Sending the image mean is useful if we remove it from the image before coding. After imperfect reconstruction the decoder can then replace the imperfect mean by the original mean. This can increases the PSNR significantly.

The first step in the EZW coding algorithm is to determine the initial threshold. If we adopt bitplane coding then our initial threshold t_0 will be

$$t_0 = 2^{\lfloor \log_2 (MAX (|\gamma(x,y)|)) \rfloor} \quad (1)$$

Here MAX (.) means the maximum coefficient value in the image and $\gamma(x,y)$ denotes the coefficient. With this threshold we enter the main coding loop

Then taking the obtained threshold as the initial value the scaled sub-band samples are been passed for dominant pass and subordinate pass. Under each pass the threshold is decreased by half the value. This comparison is carried out until the threshold reaches to the minimum threshold, the algorithm implemented is;

```
threshold = initial_threshold; do {dominant_pass(image); subordinate_pass(image); threshold = threshold/2;
} while (threshold>minimum_threshold);
```

We see that two passes are used to code the image. In the first pass, the *dominant pass*, the image is scanned and a symbol is outputted for every coefficient. If the coefficient is larger than the threshold a **P** (positive) is coded, if the coefficient is smaller than minus the threshold an **N** (negative) is coded. If the coefficient is the root of a zerotree then a **T** (zerotree) is coded and finally, if the coefficient issmaller than the threshold but it is not the root of a zerotree, then a **Z** (isolated zero) is coded. This happens when there is a coefficient larger than the threshold in the tree. The effect of using the **N** and **P** codes is that when a coefficient is found to be larger than the threshold (in absolute value or magnitude) its two most significant bits are outputted (if we forget about sign extension).

Note that in order to determine if a coefficient is the root of a zerotree or an isolated zero, we will have to scan the whole quad-tree. Clearly this will take time. Also, to prevent outputting codes for coefficients in already identified zero trees we will have to keep track of them. This means memory for book keeping.

Finally, all the coefficients that are in absolute value larger than the current threshold are extracted and placed without their sign on the subordinate list and their positions in the image are filled with zeroes. This will prevent them from being coded again.

The second pass, the *subordinate pass*, is the refinement pass. In [3] this gives rise to some juggling with uncertainty intervals, but it boils down to outputting the next most significant bit of all the coefficients on the subordinate list. In [3] this list is ordered (in such a way that the decoder can do the same) so that the largest coefficients are again transmitted first. Based on [1] we have not implemented this sorting since the gain is very small but the costs very high.

The main loop ends when the threshold reaches a minimum value. For integer coefficients this minimum value equals zero and the divide by two can be replaced by a shift right operation. If we add another ending condition based on the number of outputted bits by the arithmetic coder then we can meet any target bit rate *exactly* without doing too much work.

We summarize the above with the following code fragments, starting with the dominant pass.

Dominant pass

The image is scanned and a symbol is returned for every coefficient.

1. If the coefficient is larger than the threshold a **P** (positive) is coded.
2. If the coefficient is smaller than negative of threshold an **N** (negative) is coded.
3. If the coefficient is the root of a zero tree then a **T** (zero tree) is coded and finally,
4. If the coefficient is smaller than the threshold but it is not the root of a zero tree, then a **Z** (isolated zero) is coded. This happens when the coefficient larger than the threshold in the sub tree.

Finally, all the coefficients that are in positive value, larger than the current threshold are extracted and placed without their sign on the subordinate list and their positions in the image are filled with zeroes. This prevents them from being coded again.

After the dominant pass follows the subordinate pass:

Subordinate pass

```
subordinate_threshold = current_threshold/2; for all elements on subordinate list do { if coefficient>subordinate_threshold then
{output a one;coefficient = coefficient-subordinate_threshold;}
else output a zero;
}
```

If we use thresholds that are a power of two, then the subordinate pass reduces to a few logical operations and can be very fast.

EZW Decoding

The decoding unit reconstructs the values by identifying the symbols as positive, negative, zero tree and isolated zero tree. The reconstructed values are taken as threshold for positive coded coefficients and negative of threshold for negative coded coefficients. The zero tree coefficients and the isolated zero tree coefficients are assigned with 0 value.

VI. Example

Here we will give the complete output stream of the algorithm described above. The example data is shown in figure 3.

63	-34	49	10	7	13	-12	7
-31	23	14	-13	3	4	6	-1
15	14	3	-12	5	-7	3	9
-9	-7	-14	8	4	-2	3	2
-5	9	-1	47	4	6	-2	2
3	0	-3	2	3	-2	0	4
2	-3	6	-4	3	6	3	6
5	11	5	6	0	3	-4	4

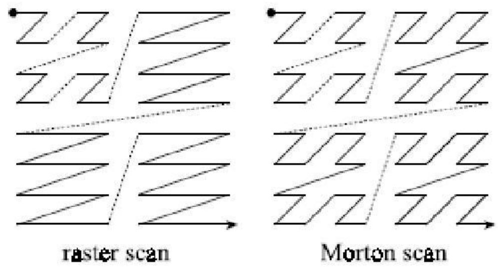

Figure 3: The example data from together with two scan orders. We used the Morton scan order.

The resulting stream, cut in dominant and subordinate passes per level, is: (Once more: beware of the scan order [1].)

D1: PNZTPTTTTZTTTTTTTPTT S1: 1010
D2: ZTNPTTTTTTTT S2: 100110
D3: ZZZZZPPNPPNTTNNPTPTTNTTTTTTTTPT TTPTTTTTTTTTPTTTTTTTTTTT
S3: 10011101111011011000
D4: ZZZZZZZTZTZNZZZZPTTPTPPTPNPTNTT TTTPTPNPPPPTTTTTPTPTTTPNP
S4: 11011111011001000001110110100010010101100
D5: ZZZZZTZZZZZTPZZZTTPTTTTNPTPPTTP TTTNPPNTTTTPNNPTTPTTPPTTT
S5: 10111100110100010111110101101100100000000110110110011000111
D6: ZZZTTZTTTZTTTTTNNTTT

The subordinate pass at the last level can be omitted because the subordinate threshold is at that moment already zero. Obviously, it makes no sense in performing it.

References

[1] Algazi, V. R. and R. R. Estes. ANALYSIS BASED CODING OF IMAGE TRANSFORM AND SUBBAND COEFFICIENTS. Proceedings of the SPIE, Vol. 2564 (1995), p. 11-21.

[2] Creusere, C. D. A NEW METHOD OF ROBUST IMAGE COMPRESSION BASED ON THE EMBEDDED ZEROTREE WAVELET ALGORITHM. IEEE Transactions on Image Processing, Vol. 6, No. 10 (1997), p. 1436-1442.

[3] Shapiro, J. M. EMBEDDED IMAGE CODING USING ZEROTREES OF WAVELET COEFFICIENTS. IEEE Transactions on Signal Processing, Vol. 41, No. 12 (1993), p. 3445-3462.