An Input Dependent Algorithm for the Inverse Discrete Wavelet Transform

Paul Fernandez, Antonio Ortega
Integrated Media Systems Center
Department of Electrical Engineering-Systems
University of Southern California, Los Angeles, CA 90089
{pfernand,ortega}@sipi.usc.edu

Abstract— **We propose a fast algorithm for computing the Inverse Discrete Wavelet Transform (IDWT). The method takes advantage of the large number of zero coefficients after quantization. A bit map of the wavelet coefficients is used to test for streams of zero coefficients and inverse transform filtering is omitted for these coefficients. The zero testing algorithm is a tree-like search operation. Information on the statistics of the wavelet transform coefficients of a few typical images, together with estimates of the cost of testing and the cost of filtering, are used to obtain optimal sizes of the root and leaves of the search tree. Results show that our algorithm is faster than the baseline inverse wavelet transform algorithm by about 20% to 50% for PSNRs in the range 35dB to 29dB.**

I. Introduction

Transforms such as the Discrete Wavelet Transform (DWT) and the Discrete Cosine transform (DCT) are used as tools to remove the correlation between neighboring pixels. DCT has been given more attention than other transforms because it does not have high memory requirements and its block based operation makes is easier to parallelize. Efficient implementations of the DCT have been widely studied over the past few years. Recently, prompted by the wide use of software - based image coding and decoding schemes, input dependent methods have been proposed to minimize the complexity of the inverse DCT. In such algorithms, when a encoder/decoder receives data that does not need to be processed (such as zero transform coefficients), it skips these computations and go on to the next set of input data. Algorithms of this type(see [1] and [2]) are by nature variable complexity and thus better suited for software implementations.

Recent tests within the JPEG 2000 standardization process show that the DWT has better performance than DCT based schemes due to both the transform and the better structures it enables. However, complexity issues related to the wavelet transform have received less attention until recently. Examples of such work include reduced memory implementations of the Discrete Wavelet transform in [4] and [5] and an input dependent algorithm for the inverse DWT[6].

The wavelet transform decorrelates the data and concentrates most of the energy in a few coefficients - hence there will be a large number of zero coefficients after quantization. Figure 1 is a histogram plot of the percentages of streams of zero coefficients for a 512x512 image with 3 levels of decomposition after quantization corresponding to a PSNR of 32dB. The streams were obtained through raster scanning. The subbands in the first and second level of decomposition have a significant percentage of zero streams of length close to the size of columns/rows in that subband. For example, the HighLow (HL) subband of the first level of decomposition has about 50% zero streams of length 256, i.e half the lines are all zero.

Work done in [6] proposes a bit-plane based progressive decoding scheme that takes advantage of zero coefficients. In this work the wavelet coefficients are decoded from the most significant bitplane to the least significant bitplane. Multiplications are avoided and the IDWT operation becomes a set of additions and right shifts. The reconstruction is done coefficient by coefficient. Our work does not use a progressive bitplane decoding scheme. However, we propose a systematic way of testing for streams of zero coefficients and a method of optimizing this scheme. This approach would give more savings than a coefficient by coefficient zero test particularly for a data set that has a significant number of zero streams. For a one dimensional coefficient array of length N and filter length is L, a k stage decomposition with b bitplanes will have of order of $k*L*N*b$ operations (consisting of zero-tests, additions and shiftings) by Guo's approach. Our method would have of the order of $k*L*N$ operations consisting of multiplications, zero-tests and additions. The cost of a multiplication is about 2-3 times that of a shift or addition.The number of bitplanes in a typical implementation is 8-16. If the cost of multiplications is not much larger than that of addition, increasing the number of additions over several bitplanes, in order to avoid multiplications, would most likely not give much reduction in complexity. Also, in a two dimensional implementation of the bit-plane scheme in software, access of bits during filtering in the second direction would require some extra overhead, increasing complexity.

The coefficient map of a 1-level decomposition is shown in figure 2(a). The standard implementation of the IDWT initially involves low-pass filtering of the up-sampled columns of the low-low (LL) and low-high (LH) subbands and the high-pass filtering of the up-sampled columns of the high-low (HL) and high-high (HH)subbands. The summed output of these two filtering operations gives the result in Figure 2(b). The next stage involves the low-pass filtering

This work was supported in part by Texas Instruments

of the rows of the up-sampled low (L) subband and the high-pass filtering of the rows of the up-sampled high (H) subband. In our algorithm, zero-testing on the columns and rows is carried out prior to filtering. The zero testing is done in a tree-like manner - the top of the tree representing testing for zero streams of a certain length and progressively lower levels of the tree corresponding to testing for zero steams of progressively shorter lengths. A method of optimizing the testing is proposed that uses the statistics of typical image coefficients.

This paper is organized as follows: in Section 2 the basic zero testing algorithm is described. Section 3 describes the optimal algorithm. In Section 4 our results are presented.

II. Basic Zero Testing Algorithm

Since most of the coefficients in the LL subband in Figure 2(a) are typically non-zero, no zero testing of coefficients is done when doing column filtering on this subband. Zero testing is done when column filtering the HL, HH and LH subbands. Figure 2(b) is the output after the column filtering.

When doing the row filtering, no zero testing is done for the L subband since it is the result of filtering the LL subband, which consisted mainly of non-zero coefficients. Zero testing for row filtering of the H subband is done as follows: Consider the set of coefficients C in Figure 2(b) - they are formed by column filtering the set of coefficients A and B in Figure 2(a) and adding the resulting outputs. Hence, when row filtering the first column of the H subband we would simultaneously test the first columns in the LH and HH subbands; if any significant coefficients are detected, the corresponding coefficients in the first column of the H subband would be row filtered. So in general, row filtering of columns in the H subband in Fig 2(b) is done by simultaneously testing corresponding columns in the LH and HH subbands. The row and column filtering is done in this manner for each level of decomposition.

We use significance bitmaps to do zero testing of coefficient streams. Each bit in the bitmap represents a coefficient. Each coefficient is tested and its corresponding bit in the bitmap set to one or zero depending on whether the magnitude of the coefficient is greater or equal to zero. In the system used in our experiments, the largest possible integer representation is 32 bits long - hence the bitmap consists of 32 bit numbers. Since column-filtering is done first, the bitmap is formed column-wise. The zero testing for the basic implementation of our algorithm will therefore start with streams of length 32.

Figure 3 illustrates the testing method. It is a left-first tree search, which was chosen because it was simple to implement. It was implemented without recursive function calls as this would decrease the speed of the algorithm. When performing the inverse filtering on a column, a bitmap number for that column is zero tested - this corresponds to the root of the tree. If it is zero, no filtering is done on the corresponding pixels. If it is non zero, the left 16 bits are tested. If they are all zero, the right 16 bits are tested. If the left 16 bits are not all zero the left 8 bits are tested, and so on. The leaf of the tree corresponds to four bits.If not all the four bits in a leaf are found to be zero, filtering is done on the corresponding four coefficients by adding the impulse response of each of the coefficients.

A deeper tree search would have larger overhead of zero testing coefficient steams. This overhead will be offset if there are a significant number of streams of zero coefficients of shorter length. For this basic implementation four bits was chosen for the leaf size after some experimentation showed that going down to two bits would increase complexity (The next section will describe a more systematic method of calculating the best tree structure). This tree structure zero testing is used for both the column and row filtering. It is easier to implement this structure if the size of the columns and rows are even.

When performing the row filtering, one has to take into account the overlap of impulse responses: We look here at the case of filters whose origin is the first coefficient. From Fig 2. if one of the leaves A or B (of length 4) are found to be non-zero, row filtering on the group of coefficients C, of length $2 * 4 + (L - 1)$ (where L = filterlength), is performed. The extra L-1 coefficients in C are the result of column filtering any significant coefficients amongst the(L-1)/2 coefficients immediately below groups A and B as well as column filtering some of the significant coefficients in A and B. Hence, when testing the (L-1)/2 coefficients just below A and B, these overlapping L-1 coefficients in C should not be row filtered if they had previously been filtered. In terms of implementation this means certain parts of the tree will not be traversed.

In general, the complexity of our algorithm would be proportional to filter length.

III. Optimal Algorithm

We address two questions that arise from the method described above: how many bits should we start the tree search at (optimal size of the root) and how many bits should we stop at (optimal size of the leaf).

To determine the optimal size of the root and leaf of the search tree, we use information on the statistics of the coefficients and estimates for the cost of testing for zero streams and the cost of filtering.

Consider zero testing starting with streams of size N. Let us define the following probabilities:

P_N= *probability of all N coefficients being zero.*

$P_{\frac{N}{2}/\overline{N}}$ = *probability of a stream of N/2 coefficients being zero given that not all N coefficients are zero*

with $P_{\frac{N}{2^{k-1}}/\overline{\frac{N}{2^k}}}$ defined in the same way.

We define the following costs:

C_f = *cost of filtering a coefficient and*

C_z = *cost of testing if a stream of bits is zero.*

The costs C_f and C_z were calculated by taking into account the number of additions, subtractions, conditional statements and multiplications. An addition, subtraction or conditional statement was given a cost of 1 while a multiplication was given a cost of 2. These estimates were based on work done by Lengwehasatit[2].

Using these probabilities and costs, complexity costs of possible trees were calculated as follows: Suppose the root of the tree is of size N and the size of a column is M: The complexity cost of testing if all N coefficients are zero and performing the filtering if they are not all zero (corresponding to a one-level tree) is:

$$C_{N->N} = C_z * \frac{M}{N} + (1 - P_N) * C_f * N * \frac{M}{N} \qquad (1)$$

The first term is the cost of testing and the second term is the cost of filtering. There are $\frac{M}{N}$ streams of length N. The number of N length streams that do not have all zero coefficients is $(1 - P_N) * \frac{M}{N}$. Hence the cost of filtering is $(1 - P_N) * C_f * N * \frac{M}{N}$

If we traverse down one level (2-level tree) the complexity cost would be:

$$C_{N->\frac{N}{2}} = Cz * \frac{M}{N} + (1 - P_N) * C_z * \frac{M}{\frac{N}{2}} + (1 - P_N) * (1 - P_{\frac{N}{2}/\overline{N}}) * C_f * \frac{M}{\frac{N}{2}} * \frac{N}{2} \qquad (2)$$

The second term is the cost of testing $\frac{N}{2}$ length streams. The probability of such streams is $(1 - P_N)$. There are $\frac{M}{\frac{N}{2}}$ such streams.

The third term in the coefficient is the cost of filtering. The probability of a non-zero stream of length $\frac{N}{2}$ is $(1 - P_N) * (1 - P_{\frac{N}{2}}/\overline{N})$ (using the chain rule from probability theory). There are $\frac{M}{\frac{N}{2}}$ such streams, and $\frac{N}{2}$ elements filtered per stream.

If the complexity cost $C_{N->\frac{N}{2}}$ was less than $C_{N->N}$, the two level tree structure would be better. The task of finding the optimal tree (i.e. optimal root and leaf size) is to find, out of all possible trees, the the tree corresponding to the minimum complexity cost.

In general, if we had a k+1 level tree (i.e going all the way down to a leaf of size $\frac{N}{2^k}$ bits and then filtering), the complexity cost would be:

$$C_{N->\frac{N}{2^k}} = Cz * \frac{M}{N} + (1 - P_N) * C_z * \frac{M}{\frac{N}{2}} + ... + (1 - P_N) * ..... * (1 - P_{\frac{N}{2^{k-1}}/\overline{\frac{N}{2^{k-2}}}}) * (1 - P_{\frac{N}{2^k}/\overline{\frac{N}{2^{k-1}}}}) * C_f * \frac{N}{2^k} * \frac{M}{\frac{N}{2^k}} \qquad (3)$$

The ranges of N and k are:$N = M....2$, $k = 0....log_2\frac{N}{2}$. Therefore there are $(log_2\frac{M}{2} + 1) * log_2\frac{N}{2}$ possible trees from which the optimal is determined.

The probabilities defined previously were gathered for each subband of each level for three training images. Then the optimal root and leaf sizes for each subband averaged over the three images were calculated.

Since the computation of the optimal trees is done off-line, the complexity of this search is not of great concern to us.

It was found that the different subbands within one level had optimal root and leaf sizes that were either identical or very close. Also, it was noted that if the LH and HH subbands had different root and leaf sizes, the complexity of doing the simultaneous zero-test on each of these subbands when performing the row filtering of the H subband increases significantly. Hence an average optimal root and leaf size was used for all the subbands in one particular level of decomposition.

If the size of the leaf of the tree is k_{opt} we can use one bit of the bit-map to represent k_{opt} pixels.

IV. Results and Discussion

We used wavelet decomposition with three levels of decomposition. The filters were conjugate-quadrature Daubechies filters of length 8. The algorithm was trained on three images: Barbara, Goldhill and Lena, and then tested on three other images (Boat, Creek and Lake) in addition to the training images.

Figure 4 represents the percentage saving in computation time of the IDWT compared to the baseline IDWT implementation vs distortion. Note that we include the time to compute the bitmap when determining the complexity for the Inverse DWT with zero testing.

Our results show a marked improvement over the baseline case. The algorithm also performed well with the three images it was not trained on. The complexity saving begins to level off after certain distortions when most insignificant coefficients are zero.

In order to determine the robustness of the algorithm it was tested on images Boat, Creek and Lake for seven tree sizes with PSNR values of 29.6dB and 31dB. Tables 1 and 2 shows the percentage decrease in complexity. The calculated optimal was case 7. It was determined from data obtained for PSNR of 31dB. At a PSNR of 29.6, the optimal still has among the better results. There is greater variation in complexity saving at PSNR of 31db compared to 29.6dB for image Lake. The reason is that at high PSNR values, there are larger number of zero streams of shorter length - hence the depth of the tree matters. On the other hand at low PSNR values there are more zero streams of longer length. In this case the algorithm does not typically traverse down most of the tree - so the structure of the tree does not matter.

We have proposed an algorithm that is about 20% to 50% faster than the standard implementation. We propose a method of calculating an optimal test structure, which is shown to be robust to different distortions. Future work could look at combining it with other implementations. One such implementation is the embedded zero tree wavelet coding scheme.

References

[1] K. Froitzheim and H.Wolf, "Knowledge-based approach to JPEG acceleration", *in Proc. of IS & T/SPIE Symp. on Electr. Imaging Science and Technology, (San Jose), Feb 1995.*

[2] K. Lengwehasatit and A.Ortega, "DCT computation with minimal number of operations", *in Proc. of VCIP 97,(San Jose, CA), Feb 1997.*

Case	L3 (R,L)	L2 (R,L)	L1 (R,L)	Boat	Creek	Lake
1	(64,8)	(64,8)	(64,8)	40	3.4	17.5
2	(32,4)	(32,4)	(32,4)	42.1	8.6	22.25
3	(16,2)	(32,4)	(64,8)	42.1	3	18.58
4	(8,2)	(8,2)	(8,2)	41.6	8.2	24.6
5	(16,4)	(16,4)	(64,8)	42.4	3	19.1
6	(16,2)	(16,2)	(16,2)	42.4	10.7	24.6
7	(8,2)	(32,4)	(32,4)	42.4	7	21.98

TABLE I

PERCENTAGE SAVING IN COMPLEXITY FOR PSNR 29.6DB FOR SEVEN CASES, KEY: L1 (R,L): LEVEL 1 (ROOTSIZE,LEAFSIZE) ETC

Case	L3 (R,L)	L2 (R,L)	L1 (R,L)	Boat	Creek	Lake
1	(64,8)	(64,8)	(64,8)	36.9	0	16.2
2	(32,4)	(32,4)	(32,4)	38.7	2.1	14.4
3	(16,2)	(32,4)	(64,8)	38.7	1	16.5
4	(8,2)	(8,2)	(8,2)	37.7	6	22.3
5	(16,4)	(16,4)	(64,8)	39	1	17
6	(16,2)	(16,2)	(16,2)	39.26	4	21.7
7	(8,2)	(32,4)	(32,4)	39	4	20.7

TABLE II

PERCENTAGE SAVING IN COMPLEXITY FOR PSNR 31DB FOR SEVEN CASES, KEY: L1 (R,L): LEVEL 1 (ROOTSIZE,LEAFSIZE) ETC

[3] W. Pennebaker and J. Mitchell, "JPEG Still Image Data Compression Standard", *Van Nostrand Reinhold, 1994.*

[4] C. Chrysafis and A. Ortega, "Line-based, Reduced Memory, Wavelet Image Compression", *Data Compression Conference, Snowbird, Utah, March 1998.*

[5] P.C. Cosman and K.Zeger, "Memory constrained wavelet-based image coding", *Signal Processing Letters, vol5, pp221-223, September 1998*

[6] H. Guo, "Mapped Inverse Discrete Wavelet Transform for Data Compression", *ICASSP, 1998.*

[7] J.M. Shapiro, "Embedded Image coding Using Zerotrees of wavelet coefficients", *IEEE trans. Signal Processing, vol 41, pp 3445-3462, December 1993.*

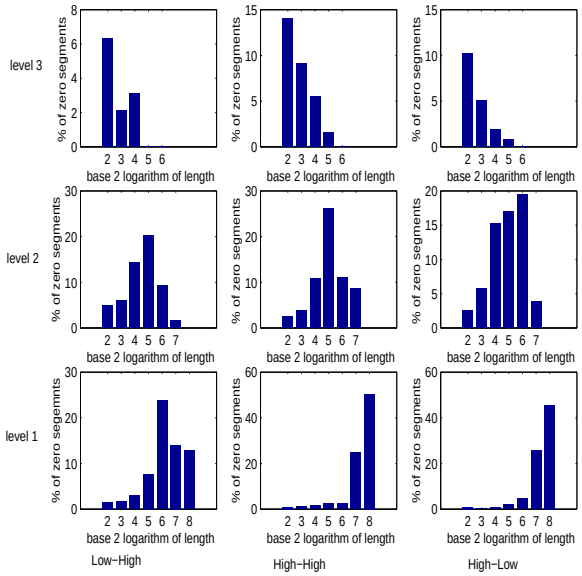


Fig. 1. Histograms of zero streams. The x-axis is the logarithm of the length of zero streams. The y-axis is the percentage of each of the zero streams:Calculated as the number of zero streams of a length divided by the total possible number of such non-overlapping streams

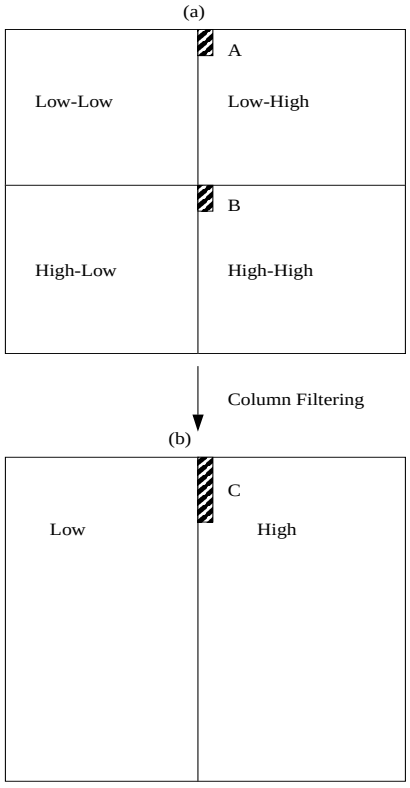
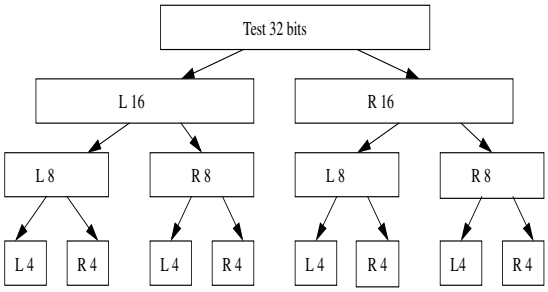


Fig. 2. Subbands for One-level Decomposition



Key:

L 16: Left 16 bits test

R 16: Right 16 bits test

Fig. 3. Zero Testing Algorithm

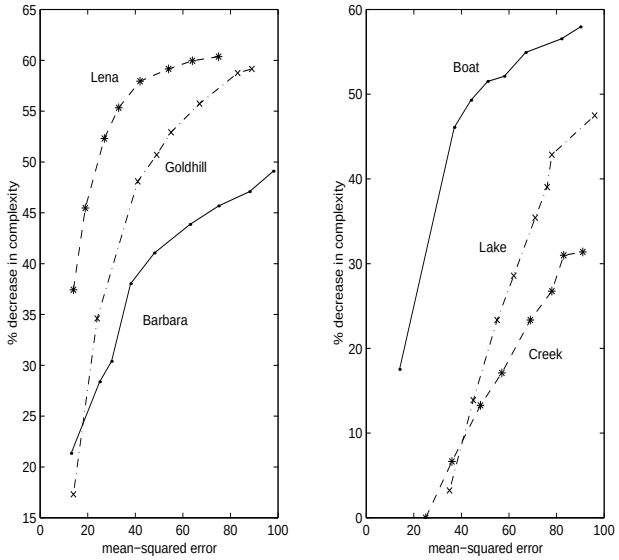


Fig. 4. Complexity Saving vs Distortion