Bul. Ştiinţ. Univ. Baia Mare, Ser. B,
Matematică – Informatică, Vol. XVIII(2002), Nr. 2, 193 - 198

THE IMPLEMENTATION OF A SPIHT WAVELET CODEC

Ovidiu COSMA

Abstract. Finding an efficient way to encode the subband coefficients is essential for revealing the potential of the wavelet transform. Saphiro's paper [1] introduces the zerotrees, as an instrument for taking advantage of the correlation between the coefficients in several subbands. The SPIHT codec presented in [2] is also based on zerotrees, but achieves better compression. In this paper are presented the results of implementing a Java codec based on the SPIHT algorithm, with little optimisations.

MSC: 68P30
Keywords: image compression

1. Zerotrees

A zerotree is a quad-tree in which all the nodes are smaller or equal with the root. Such a tree can be coded with a single symbol and reconstructed by the decoder as a quad-tree filled with zeros.

The coefficients obtained by applying the nonstandard wavelet transform [3] to an image can be represented with trees, because of the subsampling that is performed in the transform. Each coefficient in a low subband has four descendents in the next higher subband, and every descendent has four other descendents in the next higher subband.

Natural images usually have a low pass spectrum. The energy in the subbands decreases as the scale decreases, so the wavelet coefficients will be smaller in the higher subbands than in the lower subbands [4], [5]. Fig. 1 presents the first 16 x 16 wavelet coefficients, (the first four subbands) for the Lena portrait, generated with the Villasenor 18/17 filter [6].

2. The SPIHT Codec

The SPIHT encoder exploits the zerotrees, based on the property that the wavelet coefficients usually decrease with scale, and there is a good probability that all the coefficients in a quad-tree will be smaller than a certain treshold, if the root is smaller than this treshold. If the image is scanned going from the lowest subband to higher subbands, many positions can be coded efficiently with zerotrees symbols.

The algorithm builds the following data structures: The *List of Insignificant Sets* LIS that contains the roots of the zerotrees, and two lists that hold the coefficients extracted from the transformed image: the *List of Insignificant Pixels* LIP, and the *List of Significant Pixels* LSP. The LIP keeps the coefficients, that are lower than a certain treshold, and the LSP holds the coefficients that are bigger than the treshold [2].

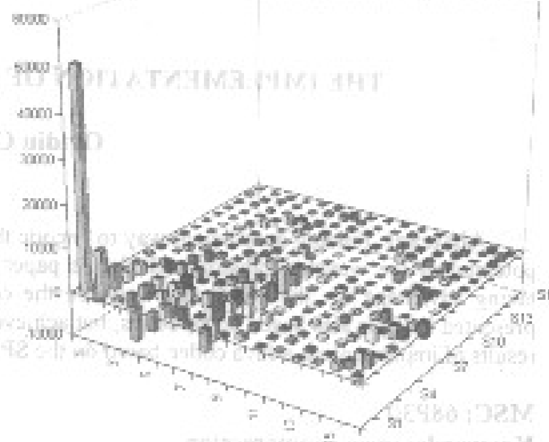

Figure 1

Coefficients of the wavelet transform.

The following sets of coordinates are used for presenting the SPIHT algorithm:

- $D(i,j)$ = all the descendents of the node (i,j) in the quad-tree
- $O(i,j)=\{(2i,2j),(2i,2j+1),(2i+1,2j),(2i+1,2j+1)\}$
- $L(i,j)=D(i,j)-O(i,j)$

$$S_n(i,j) = \begin{cases} 1, \text{ if } \max_{i,j} |c_{i,j}| \ge n \\ 0, \text{ otherwise} \end{cases}$$

The LIS contains elements (i,j) of type A that represent $D(i,j)$, and elements of type B, that represent $L(i,j)$.

2.1. The encoding algorithm

The image has $l \times l$ pixels, and l is a power of 2.

$c_{i,j}$, $0 \le i < l$, $0 \le j < l$, are the wavelet transform coefficients.

The LIP contains the coefficients $c_{i,j}$ that are below the treshold n, and LSP the coefficients that exceeded the treshold, in absolute value. The elements of the LIS have the following structure: $(i,j,type)$.

1. $p = \lfloor \log_2(\max_{i,j} |c_{i,j}|) \rfloor$

   output $n = 2^p$, l and the overall average c_{00}

   initialize LSP as an empty list

   add the coefficients $c_{i,j}$ in the lowest subband to the LIP

   add the coordinates (i,j) in the lowest subband to the LIS, as type A entries

2. for each coefficient c_z in the LIP do:

if $|c_x| \geq n$, then
    output 1
    add the value $|c_x| - n$ to the LSP
    output the sign of the coefficient c_x
    remove the coefficient c_x from the LIP
else output 0
for each entry *(i,j)* in the LIS do:
    if the entry is of type A, then
        output $S_n(D(i,j))$
        if $S_n(D(i,j)) = 1$, then
            for each $(k,l) \in O(i,j)$ do:
                if $|c_{k,l}| \geq n$, then
                    output 1
                    add the value $|c_{k,l}| - n$ to the LSP
                    output the sign of the coefficient $c_{k,l}$
                else
                    output 0
                    add the coefficient $c_{k,l}$ to the LIP
            if $L(i,j) \neq \phi$, then
                set the type B for the element *(i,j)* in the LIS
                encode_type_B_element*(i,j)*
            else
                remove the element *(i,j)* from the LIS.
    else
        encode_type_B_element*(i,j)*

3. for each element c_x in the LSP, excepting the ones included in the last pass, do:
    if $c_x \geq n$, then:
        output 1, $c_x = c_x - n$
    else output 0
4. $n = n/2$, go to step 2.

The condition for ending the process, can be related to the reach of a certain value for the current treshold n, or to the length in bits of the resulting code.

encode_type_B_element*(i,j)*:
output $S_n(L(i,j))$
if $S_n(L(i,j)) = 1$, then
    add each element $(k,l) \in O(i,j)$ to the end of LIS, as a type A element.
    remove the element *(i,j)* from the LIS

2.2. The decoding algorithm

The elements of the LSP have the structure: $(i,j,c_{i,j})$, those of the LIP have the structure: (i,j), and those of the LIS have the structure: $(i,j,\ type)$.

1. read n, l and the overall average $c_{0,0}$
   add the coordinates of the lowest subband to the LIS, as elements of the form (i,j,A)
   add the elements (i,j) in the lowest subband to the la LIP
   initialize the LSP, as an empty list
2. for each element (i,j) in the LIP do:
   - read the next bit of the coefficient $c_{i,j}$. The coefficients are read starting with the MSB.
   - if the read bit = 1, then
     - read the sign $s_{i,j}$ of the coefficient $c_{i,j}$
     - add the element $(i,j,\ s_{i,j}\cdot n\ )$ to the LSP
     - remove the element (i,j) from the LIP

   for each element (i,j) in the LIS do:
   - if the element (i,j) is of type A, then
     - read $S_n(D(i,j))$
     - if $S_n(D(i,j)) = 1$, then
       - for each $(k,l) \in O(i,j)$ do:
         - read the next bit of the coefficient $c_{k,l}$
         - if the read bit = 1, then
           - read the sign $s_{k,l}$ of the coefficient $c_{k,l}$
           - add the element $(k,l,\ s_{k,l}\cdot n\ )$ to the LSP
         - else
           - add the element (k,l) to the LIP
       - if $L(i,j) \neq \phi$, then
         - set the type B for the element (i,j) in the LIS
         - decode_type_B_element(i,j)
       - else
         - remove the element (i,j) from the LIS
   - else
     - decode_type_B_element(i,j)
3. for each element $(i,j,\ c_{i,j})$ in the LSP, excepting the ones added in the last pass, do:
   - read the next bit of the coefficient $c_{i,j}$
   - if the read bit =1, then
     - $c_{i,j} = c_{i,j} + s_{i,j}\cdot n$, where $s_{i,j}$ is the sign of the coefficient $c_{i,j}$
4. $n = n/2$, go to step 2.

The process can continue until it is reached a certain treshold n, or a certain compression ratio, or to the end of the input data. In the end of the algorithm, the reconstructed coefficients are in the LSP, in the form of $(i,j,c_{i,j})$ elements.

decode_type_B_element(i,j):

read $S_n(L(i,j))$

if $S_n(L(i,j)) = 1$, then

  add each element (k,l) in $O(i,j)$ to the LIS, as a type A element

  remove (i,j) from the LIS

3. Implementation results

My implementation follows the SPIHT algorithm, with a few optimisations. The next two images present the performance of the original SPIHT algorithm, compared to my implementation (MTE4).

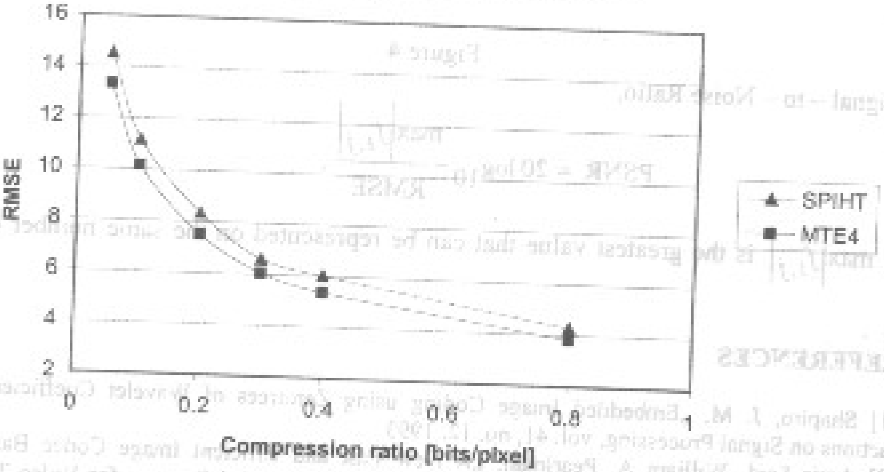

Figure 3

Root Mean Square Error:

$$RMSE = \frac{1}{L}\sqrt{\sum_{i=0}^{L-1}\sum_{j=0}^{L-1}(f_{i,j}-\tilde{f}_{i,j})^2}$$

where $f_{i,j}$ and $\tilde{f}_{i,j}$ are the pixels of the original, and those of the reconstructed image.

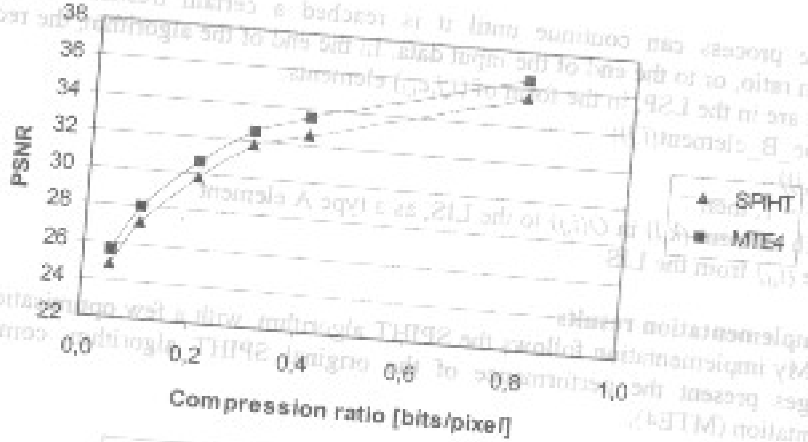

Figure 4

Peak Signal – to – Noise Ratio,

$$\mathrm{PSNR} = 20\log_{10}\frac{\max\left|f_{i,j}\right|}{\mathrm{RMSE}},$$

where $\max\left|f_{i,j}\right|$ is the greatest value that can be represented on the same number of bits as $f_{i,j}$.

REFERENCES

[1] Shapiro, J. M, „Embedded Image Coding using Zerotrees of Wavelet Coefficients", IEEE Transactions on Signal Processing, vol. 41, no. 12, 1993

[2] Amir Said, William 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, june 1996

[3] Ovidiu Cosma, Wavelet Based Multiresolution Analysis, Buletins for Applied & Computer Mathematics, Technical University of Budapest, 2001

[4] Ovidiu Cosma, Choosing a Color Space for Image Compression, Buletins for Applied & Computer Mathematics, Technical University of Budapest, 2001

[5] Ovidiu Cosma, Optimizing the color space conversion for image compression, Bul. Ştiinţ. Baia Mare Ser. B, Matematică – Informatică, Vol. XVII, 2001

[6] J. Villasenor, B. Belzer, J. Liao, "Wavelet filter evaluation for image compression", IEEE Transactions on Image Processing, vol. 2, Aug. 1995

Received: 19.09.2002

Department of Mathematics and Computer Science
Faculty of Sciences, North University of Baia Mare
E-mail: cosma@alphanet.ro