RESEARCH ARTICLE OPEN ACCESS

Low Memory Low Complexity Image Compression Using HS-SPIHT Encoder

D.L.Vasundhara[1], B.Tech, (M.Tech), P. A. Nageswara Rao[2], M.E, (Ph. D) A.Vamsidhar[3], M.Tech, (Ph. D)
Associate Professor
Associate Professor,
1,2,3 Department of Electronics & Communication Engineering, Dadi Institute of Engineering and Technology.

Abstract:
Due to the large requirement for memory and the high complexity of computation, JPEG2000 cannot be used in many conditions especially in the memory constraint equipment. The line-based W avelet transform was proposed and accepted because lower memory is required without affecting the result of W avelet transform, In this paper, the improved lifting schem e is introduced to perform W avelet transform to replace Mallat method that is used in the original line-based wavelet transform. In this a three-adder unit is adopted to realize lifting scheme. It can perform wavelet transform with less computation and reduce memory than Mallat algorithm. The corresponding HS_SPIHT coding is designed here so that the proposed algorithm is more suitable for equipment. W e proposed a highly scale image compression scheme based on the Set Partitioning in Hierarchical Trees (SPIHT) algorithm. Our algorithm, called Highly Scalable SPIHT (HS_SPIHT), supports High Compression efficiency, spatial and SNR scalability and provides l bit stream that can be easily adapted to give bandwidth and resolution requirements by a simple transcoder (parse). The HS_SPIHT algorithm adds the spatial scalability feature without sacrificing the S NR embeddedness property as found in the original SPIHT bit stream. Highly scalable image compression scheme based on the SPIHT algorithm the proposed algorithm used, highly scalable SPIHT (HS_SPIHT) Algorithm, adds the spatial scalability feature to the SPIHT algorithm through the introduction of multiple resolution dependent lists and a resolution-dependent sorting pass. SPIHT keeps the import features of the original SPIHT algorithm such as compression efficiency, full SNR Scalability and low complexity.

Keywords: wavelet transforms; lifting schemes; Scalable SPHIT; Compression; low complexity.

I. INTRODUCTION

Although JPEG2000 can provide both objective and subjective image quality superior to existing standards. The large requirement for memory and the high complexity of computation, as its bottleneck, make it cannot be used in many conditions especially in the memory constraint equipment, such as printer or digital camera, due to the need to maintain low costs. The line-based wavelet transform algorithm was proposed in [1] based on conventional Mallet algorithm. it comes from a progressive process to perform the vertical filtering in the wavelet transform and output coefficients, and only several image lines is to be stored while the whole image must be stored with the conventional wavelet transform, so the memory requirement is reduced greatly.

Lifting scheme was proposed by Sweldens and has become the key technology of the second generation wavelet. In this paper, the lifting scheme is discussed to replace Mallet algorithm in the line-based wavelet image compression. It can perform wavelet transform with less computation and reduced memory than Mallet algorithm: Considering the character of the lifting step, the three-adder unit is used to implement the lifting scheme. The coefficient are outputted and coded progressively by the line-based wavelet transform.

Traditional image coding systems have only focused on efficient compression of image data.

The main objective of such systems is optimizing image quality at given bit rate. Due to the explosive growth of the Internet and networking technology, nowadays a huge number of users with different capabilities of processing and network access bandwidth can transfer and access data easily.

For Transmission of visual data on such a heterogeneous network, efficient compression itself is not sufficient. There is an increasing demand for scalability to optimally service each user according to his bandwidth and computing capabilities. A scalable image coder generates a bit stream which consists of a set of embedded parts that offer increasingly better signal-to-noise ratio (SNR) or/and greater spatial resolution. Different parts of this bit stream can be selected and decoded by a scalable decoder to meet certain requirements.

In the case of an entirely scalable bit stream, different types of decoders with different complexity and access bandwidth can coexist.

An improved scheme, called Set Partitioning in Hierarchical Trees (SPIHT) was developed by Said and Pearlman [2]. This coder uses the spatial orientation trees shown in fig.1 and partitions them as needed to sort wavelet of SPIHT. Although the SPIHT coder is fully SNR scalable with excellent compression properties, it does not explicitly support spatial scalability and does not provide a bit stream that can be parsed easily according to the type of scalability desired by the decoder.

In this paper, a fully scalable image coding scheme based on the SPIHT algorithm is presented. We modify the SPIHT algorithm to support both spatial and SNR scalability features. The encoder creates a bit stream that can be easily parsed to achieve different levels of resolution or/and quality requested by the decoder.

A distinct feature of the presented coder is that the reordered bit streams for different spatial resolutions, which are obtained after parsing the main bit stream, are fully embedded and can be truncated at any point to obtain the best reconstructed image at the desired spatial resolution and bit rate. In the other words, HS- SPIHT algorithm provides spatial scalability without sacrificing SNR scalability in any way.

Chapter 2 discusses the details of wavelets and SPIHT algorithm Chapter 3 discusses the implementation details of HS_SPIHT and line based wavelet Chapter 4 discuses the results Finally chapter 5 concludes the thesis and identifies the direction for the future work References

II. WAVELETS AND SPHIT ALGORITHM

2.1 Wavelet Transform

One of the most important characteristics of wavelet transform is multi resolution decomposition. An image decomposed by wavelet transform can be reconstructed with desired resolution. In the thesis, a face image is first transformed to the wavelet domain using pyramid decomposition Fig. 2 shows

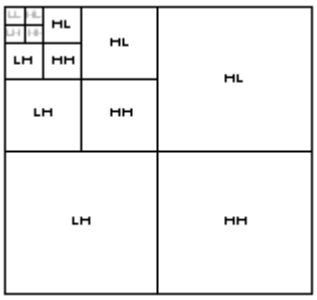


Fig.1 Four-level wavelet decomposition.

four-level wavelet decomposition. In our experiments, an image is decomposed using the Daubechies orthogonal filter of length 8 up to five decomposition levels. The procedure goes like this. A low pass filter and a high pass filter are chosen, such that they exactly halve the frequency range between themselves. This filter pair is called the Analysis Filter pair. First, the low pass filter is applied for each row of data, thereby getting the low frequency components of the row. But since the LPF is a half band filter, the output data contains frequencies only in the first half of the original frequency range. So, by Shannon's Sampling Theorem, they can be sub sampled by two, so that the output data now contains only half the original number of samples. Now, the high pass filter is applied for the same row of data, and similarly the high pass components are separated, and placed by the side of the low pass components. This procedure is done for all rows. Next, the filtering is done for each column of the intermediate data. The resulting two-dimensional array of coefficients contains four bands of data, each labeled as LL (low-low), HL (high-low), LH (low-high) and HH (high-high). The LL band can be decomposed once again in the same manner, thereby producing even more sub bands. This can be done up to any level, thereby resulting in a pyramidal decomposition as shown below.

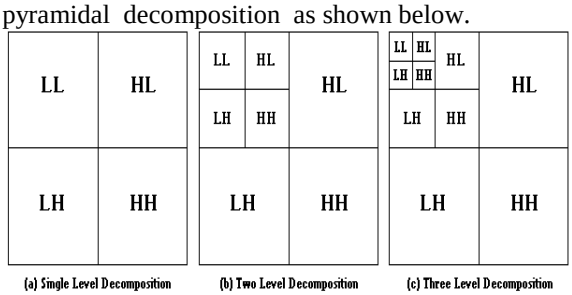


Fig.2. Pyramidal Decomposition of an Image

As mentioned above, the LL band at the highest level can be classified as most important, and the other detail' bands can be classified as of lesser importance, with the degree of importance decreasing from the top of the pyramid to the bands at the bottom.

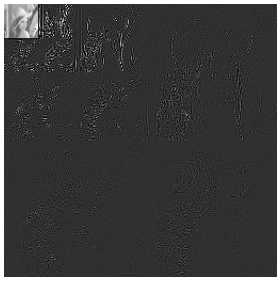


Fig.3. Three layer decomposition of the 'Lena' image.

2.2 The Inverse DWT of an Image

Just as a forward transform to use to separate the image data into various classes of importance, a reverse transform is used to reassemble the various classes of data into a reconstructed image. A pair of high pass and low pass filters is used here also. This filter pair is called the Synthesis Filter pair. The filtering procedure is just the opposite - we start from the topmost level, apply the filters column wise first and then row wise, and proceed to the next level, till we reach the first level.

2.3 SPIHT (Set partitioning In Hierarchical Trees)

The SPIHT method is not a simple extension of traditional methods for image compression, and represents an important advance in the field. The SPIHT (set partitioning in hierarchical trees) is an efficient image coding method using the wavelet transform. Recently, image-coding using the wavelets transform has attracted great attention. Among the many coding algorithms, the embedded zero tree wavelet coding by Shapiro and its improved version, the set partitioning in hierarchical trees (SPIHT) by Said and Pearlman have been very successful. Compared with JPEG - the current standard for still image compression, the EZW and the SPIHT are more efficient and reduce the blocking artifact.

The method deserves special attention because it provides the following:

- Good image quality, high PSNR, especially for color images;
- It is optimized for progressive image transmission; Produces a fully embedded coded file;
- Simple quantization algorithm;
- Can be used for lossless compression;
- Can code to exact bit rate or distortion;
- Fast coding/decoding (nearly symmetric);
- Has wide applications, completely adaptive; Efficient combination with error protection.

Each of these properties is discussed below. Note that different compression methods were developed specifically to achieve at least one of those objectives. What makes SPIHT really outstanding is that it yields all those qualities simultaneously.

2.4 **Optimized Embedded Coding**

A strict definition of the embedded coding scheme is: if two files produced by the encoder have size M and N bits, with M > N, then the file with size N is identical to the first N bits of the file with size M. Let's see how this abstract definition is used in practice. Suppose you need to compress an image for three remote users. Each one has different needs of image reproduction quality, and you find that those qualities can be obtained with the image compressed to at least 8 Kb, 30 Kb, and 80 Kb, respectively. If you use a non-embedded encoder (like JPEG), to save in transmission costs (or time) you must prepare one file for each user. On the other hand, if you use an embedded encoder (like SPIHT) then you can compress the image to a single 80 Kb file, and then send the first 8 Kb of the file to the first user, the first 30 Kb to the second user, and the whole file to the third user. Surprisingly, with SPIHT all three users would get (for the same file size) an image quality comparable or superior to the most sophisticated non-embedded encoders available today. SPIHT achieves this feat by optimizing the embedded coding process and always coding the most important information first.

SPIHT codes the individual bits of the image wavelet transform coefficients following a bit-plane sequence. Thus, it is capable of recovering the image perfectly (every single bit of it) by coding all bits of the transform. However, the wavelet transform yields perfect reconstruction only if its numbers are stored as infinite-precision numbers. In practice it is frequently possible to recover the image perfectly using rounding after recovery, but this is not the most efficient approach.

A codec that uses this transformation to yield efficient progressive transmission up to lossless recovery is among the SPIHT .A surprising result obtained with this codec is that for lossless compression it is as efficient as the most effective lossless encoders (lossless JPEG is definitely not among them). In other words, the property that SPIHT yields progressive transmission with practically no penalty in compression efficiency applies to lossless compression too.

2.5 Hierarchical Tree

In this subsection, we will describe the proposed algorithm to code the wavelet coefficients. In general, a wavelet decomposed

image typically has non-uniform distribution of energy within and across sub bands. This motivates us to partition each sub band into different regions depending on their significance and then assign these regions with different quantization levels. The proposed coding algorithm is based on the set partitioning in hierarchical trees (SPIHT) algorithm, which is an elegant bit-plane encoding method that generates M embedded bit sequence through M stages of successive quantization. Let s_0, s_1… s_{M-1} denote the encoder's output bit sequence of each stage. These bit sequences are ordered in such a way that s_0 consists of the most significant bit; s_1 consists of the next most significant bit, and so on. The SPIHT algorithm forms a hierarchical quad tree data structure for the wavelet transformed coefficients. The set of root node and corresponding descendents are referred to as a spatial orientation tree (SOT) (see Fig.4). The tree is defined in
such a way that each node has either no leaves or four offspring, which are from 2 x 2 adjacent pixels. The pixels on the LL sub imageof the highest decomposition level are the tree roots and are also grouped in 2 x 2 adjacent pixels. However, the upper- left pixel in 2 x 2 adjacent pixels (as shown in Fig. 4) has no descendant. Each of the other three pixels has four children.

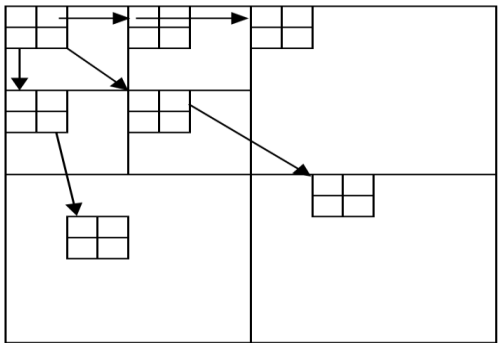


Fig.4. Quad tree organization of wavelet coefficients in SPHIT algorithm.

For the convenience of illustrating the real implementation of SPIHT, the following sets of coordinates are defined.

(1) O(i, j): set of coordinates of all offspring of node (i, j);

(2) D(i, j): set of coordinates of all descendants of the node (i, j);

(3) H: set of coordinates of all spatial orientation tree roots (nodes in the highest pyramid level);

(4) L (i, j) =D (i, j)-O (i, j).

Thus, except at the highest and lowest levels, we have

O (i, j) = {(2i, 2j), (2i, 2j+1), (2i+1, 2j), (2i+1, 2j+1)}. Define the following function.

$$S_n(\gamma) = \begin{cases} 1, & \max_{(i,j)\in\gamma}\{|c_{i,j}|\} \geq 2^n, \\ 0, & \text{Otherwise.} \end{cases} \quad (1)$$

$S_n(\gamma)$ indicates the significance of a set of coordinates, γ
where T (n) is the preset significant threshold used in
the *n*th stage. A detailed description of the SPIHT
coding algorithm is given as follows.
Initially, $T(0)$ is set to be 2^{-1} where M is selected such that the largest coefficient magnitude, say cmax, satisfies $2M-1 \leq cmax < 2M$. The encoding is progressive in coefficient magnitude for successively using a sequence of thresholds T (n) = 2(M-1)-n, n=0, 1… M-1. Since the thresholds are a power of two, the encoding method can be regarded as "bit-plane" encoding of the wavelet coefficients. At stage *n*, all coefficients with magnitudes between *T (n)* and *2T (n)* are identified as "significant,"
and their positions and sign bits are encoded. This
process is called a *sorting pass*. Then, every coefficient with magnitude at least 2*T* (n) is "refined" by encoding the *n*th most significant bit. This is called a *refinement pass*. The encoding of significant coefficient position and the scanning of the significant coefficients for refinement are efficiently accomplished through using the following three lists: the list of significant pixels (LSP), the list of insignificant pixels (LIP), and the list of insignificant set (LIS). Each entry in the LIP and LSP represents an individual pixel which is identified by coordinates (i, j). While in the LIS, each entry represents either the set D (i j) or L (i, j). An LIS entry is regarded as of type A if it represents D (i, j) and of type B if it represents L (i, j).

2.6 Algorithm: SPIHT coding algorithm

Step 1: (Initialization)

Output $n = \lfloor \log_2 (\max_{(i,j)} \{|i, j|\}) \rfloor$, set the LSP as an empty list,
add the coordinates (i, j) ε H to the LIP,

add the coordinates (i, j) ε H with descendants
to the list LIS, as type A entries, Step 2: (Sorting Pass)
2.1) for each entry (i, j) in the LIP do:
Output Sn (i, j), if Sn (i, j) =1
move (i, j) to the LSP,

output the sign of $c_{i,j}$,

2.2) for each entry (i, j) in the LIS do:

2.2.1) if the entry is of type A then
output Sn (D(i, j)),
if Sn (D(i, j)) = 1 then

* for each (k,l)

output Sn(k, l),

if Sn (k, l) = 1 then
add (k, l) to the LSP, output the sign of ck,l, if Sn (k, l) = 0 then
add (k, l) to the end of the LIP,

*if L (i, j) ≠ 0 then

move (i, j) to the end of the LIS, as an entry of type B,

go to Step 2.2.2). otherwise
remove entry (i, j) from the LIS,

2.2.2) if the entry is of type B

output Sn (L(i, j)),

if Sn (L(i, j)) = 1 then

*add each (k, 1) ε O (i, j) to the end of the LIS as an entry of type A,

*remove (i, j) from the LIS, Step 3: (Refinement Pass)

For each entry (i, j) in the LSP, except those included in the last sorting pass (i.e., with the same n), output the *n*th most significant bit of | ci, j|,
Step 4: (Quantization-Step Update)
Decrement *n* by 1 and go to Step 2.

2.7 Proposed and Existing Block Diagrams:
Proposed

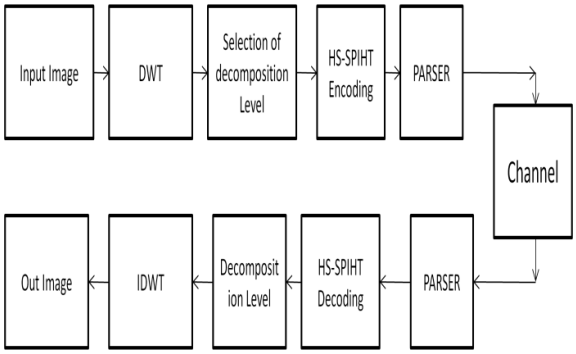


Fig.5. Block Diagram of proposed Approach

Existing:

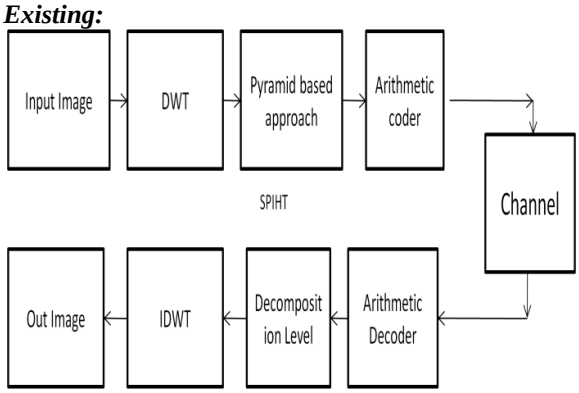


Fig.6. Block Diagram of Existing

III. LINES-BASED WAVELET TRANSFORM:

The conventional wavelet transform requires that all the lines be horizontally filtered before the vertical filtering and thus the total image data should be stored. However, the line-based wavelet transform is to start vertical filtering as soon as a sufficient number of lines, as determined by the filter length have been horizontally filtered. Then this algorithm only requires storing a minimum number of lines. The image data are stored in memory only while they are used to generate output coefficients, and released from memory immediately once no longer being needed, therefore the memory requirements are greatly reduced.

The filter length will be denoted as L and buffer size is indicated in terms of number of lines. The memory requirements of this algorithm depend on the length of the filters, the width of the image and also the levels of the transform. One more level of the transform will be performed; one more filter buffer will be needed.

In order to keep the output queues encoder and decoder synchronous, the high pass data of each transform level do not output until the highest level of transform has been performed. Therefore the additional synchronization buffers are needed to store the intermediate high pass data.

The lifting scheme has some properties which are not found in many other transforms.

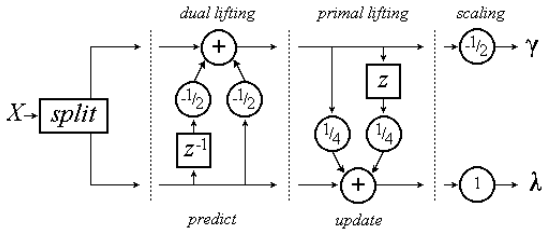


Fig.7. Lifting scheme

The inverse transform is immediately clear: change the signs of all the scaling factors, replace "*split*" by "*merge*" and go from right to left, i.e. reverse the data flow. This easy invertibility is always true for the lifting scheme. Lifting can be done in-place: we never need samples other than the output of the previous lifting step and therefore we can replace the old stream by the new stream at every summation point. Not immediately clear from this figure is that when we iterate a filter bank using in-place lifted filters we end up with *interlaced coefficients*. This can be seen as follows. We split the input in odd- and even-numbered samples and perform the in-place lifting steps. After one complete step the high-pass filtered samples, the wavelet coefficients, sit in the odd-numbered places and the low-pass filtered samples sit in the even- numbered places. Then we perform another transform step, but only using the low-pass filtered samples, so that this sequence will again be divided into odd- and even-numbered samples. Again the odd-numbered

samples are transformed into wavelet coefficients, while the even-numbered samples will be processed further so that in the end all wavelet coefficients will be interlaced.

The third important property has not been mentioned yet, but it shows clearly from figure lifting is not causal. Usually this is not really a problem, we can always delay the signal enough to make it causal, but it will never be real- time. In some cases however it is possible to design a causal lifting transform.

The last important property we will mention here is the calculation complexity. In [4] it is proven that for long filters the lifting scheme cuts computation complexity in half, compared to the standard iterated FIR filter bank algorithm. This type of wavelet transform has already a complexity of N, in other words, much more efficient than the FFT with its complexity of *N* log (*N*) and lifting speeds things up with another factor of two. This is where the title of this tutorial comes from: it is a fast wavelet transform and therefore we will refer to it as the *fast lifting wavelet transform* of *FLWT.*

3.1 Integer lifting

The last stage of our voyage to the ultimate wavelet transform is the stage where we make sure that the wavelet coefficients are integers. In classical transforms, including the non-lifted wavelet transforms, the wavelet coefficients are assumed to be floating point numbers. This is due to the filter coefficients used in the transform filters, which are usually floating point numbers. In the lifting scheme it is however rather easy to maintain integer data, although the dynamic range of the data might increase. That this is possible in the lifting scheme has to do with the easy invertibility property of lifting.

The basic lifting step is given in and we rewrite it here a little modified as

$$x_{new}(z) \leftarrow x(z) + s(z)\,y(z) \quad (2)$$

Because the signal part $y(z)$ is not changed by the lifting step, the result of the filter operation can be rounded, and we can write:

$$x_{new}(z) \leftarrow x(z) + \lfloor s(z)\,y(z) \rfloor \quad (3)$$

the even subset and update the even subset from the wavelet coefficients, the lifting steps are as follow, where Un (k) and Pn (k) stand for updater and predictor respectively:

Where we have used $\lfloor \cdot \rfloor$ to denote the rounding operation. is fully reversible:

$$x(z) \leftarrow x_{new}(z) - \lfloor s(z)\,y(z) \rfloor \quad (4)$$

And this shows the most amazing feature of integer lifting: whatever rounding operation is used, the lifting operation will always be reversible.

We have to take care however, because we did not consider the scaling step in the previous paragraph. Scaling usually does not yield integer results but it is a part of the lifting transform. The simplest solution to this problem is to forget all about scaling and just keep in mind that the transform coefficients actually have to be scaled. This is important for instance in denoising applications. If scaling is ignored, then it is desirable to let the scaling factor be as close to one as possible. This can be done using the non-uniqueness of the lifting factorization. Another solution is to factor the scaling into lifting steps as well [4]. As mentioned before the integer lifting

transform cannot guarantee

the preservation of the dynamic range of the input signal. Usually the dynamic range doubles [5], [10] but there are schemes that can keep the dynamic range. In [Cha96] an interesting lifting transform with the so- called *property of precision preservation* (*PPP*) is described. This transform makes use of the two-complement representation of integers in a computer and the wrap- around overflows cause in this representation. The disadvantage of such a transform is that large coefficients may be

represented by small values and it is therefore difficult to take decisions on coefficient values.

3.2 Line based Wavelet Transform Based on lifting scheme

According to the above discussion, line based wavelet image coding is based on traditional Mallat algorithm. Considering the merits of the lifting scheme, it is introduced into line based wavelet image coding that lowers computational complexity and reduces memory requirements further.

A typical wavelet transform constructed by lifting scheme consists of three basic stages, as followed, split, predict, update. Firstly split the original data xi into two disjoint subsets based on lazy wavelet decomposition, named the even subsets si0 E x2i and the odd subset di0 E x2i+ 1. Then predict the odd subset from

din = din-1 Σ Un(k) sin-1 (5)

sin = sin-1 Σ Pn(k) sin-1 (6)

Last normalize the output with factor k0 and k1.

siN = k0 si (7)

diN = k1 di (8) Thus, original signal can be expressed by din , sin , Un (k) , Pn (k) , k0 and k1.

In [6] proved that any FIR wavelet transform could be constructed via several lifting steps. The lifting factorization leads to the following implementation of *3+, where α and γ are the updater, β and δ are the predictor.

si (0) = x2i (9) di (0) = x2i + 1 (10) di (1) = di(0) + α (si(0) + si -1(0)) (11) si (1) = si(0) + β (di (1) + di -1(0)) (12) di (2) = di(1) + γ (si(1) + si -1(1)) (13) si (2) = si(1) + δ (di (2) + di -1(2)) (14) si = ρx si (2) (15) di= di(2) / ρ (16)

From the lifting steps of [3], each step can be expressed with a three adder unit as shown in fig, where R0, R1 and R2 are served as input buffer, Routput is used as output buffer and c is a multiplication factor. In this paper, the lifting scheme implemented by the three- adder unit is to replace Mallat algorithm.

Routput = R1 + c (R0 + R2) (17)

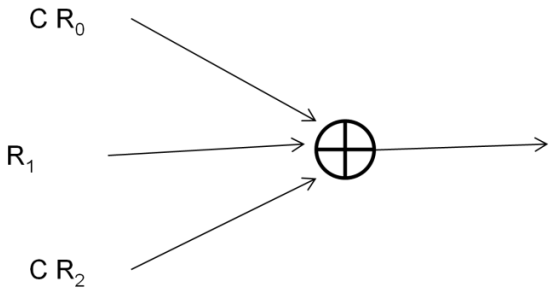


Fig.8. Three-adder Unit

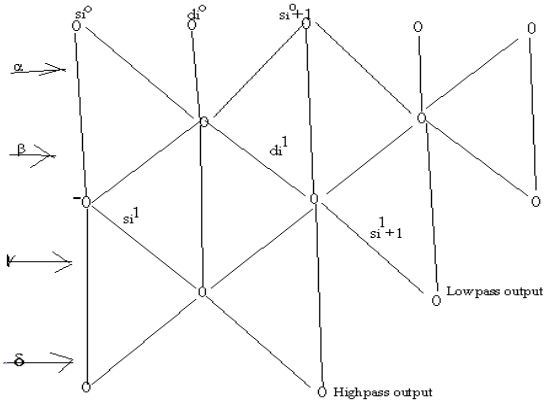


Fig.9.Lifting Steps for Horizontal wavelet transform

Following the line based wavelet transform algorithm, every image line will be horizontally filtered as shown in fig 10 as soon as encoder reads it in. The original image data can be replaced directly by the current transform results because of the in-place character of the three adders lifting steps, so that less computational steps and no auxiliary memory are needed. However, the data belong to the same sub-band cannot be worked out continuously so as to the result data are interlaced with each other.

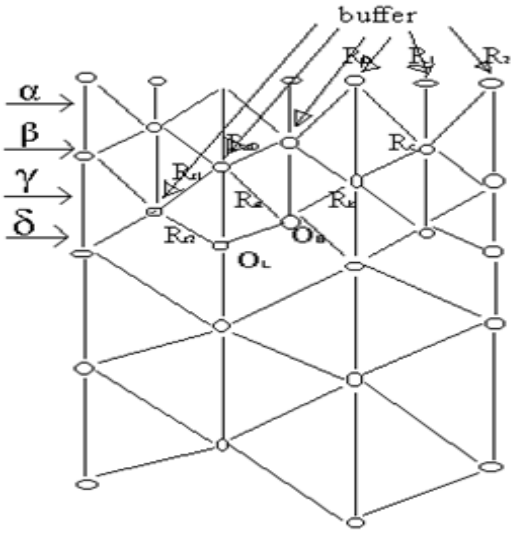


Fig.10. Lifting steps vertical wavelet transforms (Two levels)

The progressive wavelet transform in vertical direction is shown in fig 11. Each point in horizontal direction means a line of data decomposed by the horizontal transform. The points in vertical direction mean the content update of the line. From the Fig 11 We can compute the low pass output OL and high pass output OH :

OL = δ (Rt2 + Rt1 + Ra) = δ (Rt2 + Rt1) + δ γ (Rt1 + Rt0 + Rb)= δ Rt2+ δ (1 + γ) Rt1 + δ γ Rt0 + δ γ β (Rt0 + R0 + Rc= δ Rt2+ δ (1 + γ) Rt1 + δ γ (1 + β) Rt0 + δ γ β R0 +δ γ β α (R0 + R1 + R2)=δ Rt2+ δ (1 + γ) Rt1 + δ γ (1 + β) Rt0 + δ γ β (1 + α) R0 + δ γ β αR1 + δ γ β αR2 (18)

OH =Ra = γ (Rt1 + Rt0 + Rb) = γRt1 + γ Rt0 + γ β (Rt0 + R0 + Rc) = γ Rt1+γ (1 + β) Rt0 + γ β R0 + γ β α (R0 + R1 + R2) = γ Rt1 +γ (1 + β) Rt0 + γ β (1 + α) R0 + γ β αR1 + γ β αR2 (19)

Then we can see that OL and OH are only relative with the signal data R0, R1, R2 and temporary data Rt2, Rt1, Rt0. In general multilevel wavelet transform , only the LL output is required to buffer for the next level and the length of each sub band exponentially decreases by 2, 3 level transforms will be performed comparing to Mallat algorithm, then the signal buffer:

BLS = 3 * (1+ ½ + ¼) = 21/4 (20)
The temporary buffer:
BLT = 3 * ½ * (1 + ½ + ¼) = 21/8 (21)

Total memory required with three adder unit is equal to 63/8 lines while total memory is equal to 43 lines with Mallat algorithm. Much more memory has been saved using lifting scheme with three adder unit.

3.3 HS_SPIHT:

In general, N Level wavelet decomposition allows at most N+1 levels of spatial resolution. To distinguish between different resolutions levels, we denote the lowest spatial resolution level as level N+1.

The full image then becomes resolution level 1. The three sub bands that need to be added to increase the resolution from level k to level k-1 are referred to as level k-1 resolution sub bands.

An algorithm that provides full spatial scalability would encode the different resolution levels separately, allowing a Trans coder or the decoder to directly access the data needed to reconstruct with a desired spatial resolution. The original SPIHT algorithm, however, encodes the entire wavelet tree in a bit plane by bit plane manner and produces a bit stream that contains the information about the different spatial resolutions in no particular order.

The HS_SPIHT [7] algorithm proposed in this paper solves the spatial scalability problem through the introduction of a resolution-dependent sorting pass that uses one additional list, called the list of delayed insignificant sets (LDIS). The HS_SPIHT coder first encodes all bit planes for a given (low) resolution level and then moves to the next higher resolution level. Sets encountered during the sorting pass that lie outside the actually considered spatial resolution are temporarily stored in the LDIS. They are moved back from the LDIS into the LIS when they are required for encoding the next higher resolution. According to the magnitude of the coefficients in the wavelet pyramid, coding of higher resolution bands usually starts from lower bit planes.

Therefore, during the encoding process of resolution level k, the encoder keeps the number of coefficients that went to the LDIS for each quantization level.

After finishing the encoding process for all bit places of resolution level k, the encoder knows which entries in the LDIS that belong to which bit plane. To encode the additional three sub bands for resolution level k-1, it moves the related entries of the LDIS that belong to the actual bit plane to the LIS and carries out the sorting of LIS with the same procedure as before. Altogether, the total number of bits belonging to a particular bit plane

is the same for SPIHT and HS_SPIHT, but HS_SPIHT distributes them differently among the different spatial resolution levels.

The definitions for terms required by HS_SPIHT are the same as for SPIHT and are therefore not listed here. In the following we list the entire HS_SPIHT coding algorithm:

3.4 Algorithm: HS_SPIHT coding algorithm

Step 1: (Initialization)
Output $n = \lfloor \log 2 (\max (i , j) \{| ci , j |\} \rfloor$, set the LSP as an empty list,

add the coordinates (*i, j*) *H* to the LIP,
add the coordinates
(*i, j*) *H* with
descendants to the list LIS, as type A entries,
Step 2: (Sorting Pass)
2.1) for each entry (i, j) in the LIP do:

output Sn(i, j), if Sn(i, j)=1
move (i, j) to the LSP, output the sign of ci,j,
2.2) for each entry (i, j) in the LIS do:

2.2.1) if the entry is of type A then

if all coordinates in the (D(i, j) are located outside the spatial resolution then move(i.j) to LDIS as type A

output Sn(D(i, j)),

if Sn(D(i, j)) = 1 then
* for each (k, l) $\in$

output Sn(k, l),
(i, j) do:
if Sn(k, l) = 1 then

Set n with the maximum quantization level related to the first entry that was mob\ved from the LlS to LDIS during the resolution dependent sorting pass for resolution level k+1

Move back all entries in LDIS which were moved to the add (k, l) to the LSP, output the sign of ck,l,

if Sn(k, l)=0 then

add (k, l) to the end of the LIP,

*if L(i, j) $\neq 0$ then

move (i, j) to the end of the LIS, as an entry of type B,
go to Step 2.2.2). otherwise
remove entry (i, j) from the LIS,
2.2.2) if the entry is of type B if all coordinates in the (D(i, j) are located outside the spatial resolution then move(i.j) to
LDIS as type B

output Sn(L(i, j)), if Sn(L(i, j)) = 1 then
*add each (k, l) $\in$ (i, j) to the end of the LIS as an entry of type A,

*remove (i,j) from the LIS, Step 3: (Refinement Pass)
For each entry (i, j) in the LSP, except those
included in the last sorting pass (i.e., with the same n), output the *n*th most significant bit of
$| c_{i,j} |$,

Step 4: (Quantization-Step Update)
Decrement *n* by 1 and go to
Step 2.
If n is greater than minimum than bit plane then k = kmax then go to step 2 else step 5.1

Step 5

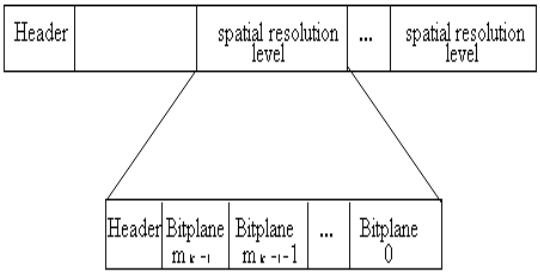
If k>1
Then decrement k by 1
Set the LIP, LIS, and LSP as empty list
LDIS during quantization level
N for sorting pass for resolution level k+1 to the LIS and go to step 2 Else end of coding

To support bit stream parsing by an image server/transcoder, some markers are required to be put into the bit stream to identify the parts of the bit stream that belong to the different spatial resolution levels and bit planes. This additional information does not need to be sent to the decoder.

3.5 Bit Stream Formations and Parsing:
The bit stream generated by the encoder can be sorted in different ways; however it can exist in one specific order at a time only. Fig.11 shows the bit stream structure generated by the encoder. The bit stream is divided into different parts according to the different spatial resolution levels. Inside each resolution level the bits that belong to different bit planes are separable. A header at the beginning of the bit stream identifies the number of spatial resolution levels supported by the encoder, as well as information such as the image dimension, number of wavelet decomposition levels, and the maximum quantization level. At the beginning of each resolution level there is an additional header that provides the information required to identify each biplane. In Fig.11 mk-1 is the first (highest) biplane used for coding of the level k – 1 spatial resolution sub bands.

After encoding the original image at high bit rate, the bit stream is stored on an image server. Different users with different requirements send their request to the server and the server or a transcoder within the network provides them with a properly tailored bit stream that is easily

generated by selecting the related parts of the original bit stream and ordering them in such a way that the user request is fulfilled. To carry out the parsing process, the image server or transcoder does not need to decode any parts of the bit stream.



according to spatial resolution and quality.

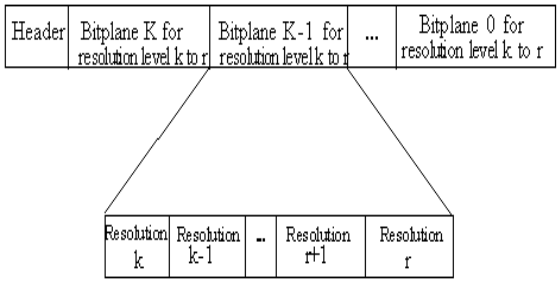


Fig.12. Reordered bit stream for spatial resolution level r

There are two principal ways of manipulating the encoded bit stream by the image server or transcoder to meet the user requests:

1. The bit stream gets truncated to the required resolution level, and in addition, some of the lower bit planes may be removed. This yields a resolution embedded, but not a fully SNR embedded bit stream. To enable decoding, the headers introduced by the encoder need to be kept.
2. The bit stream is reordered bit plane by bit plane for the requested resolution level. This form of reordering results in a fully SNR embedded bit stream for the desired resolution. Fig.12 shows an example of a reordered bit stream for spatial resolution bit planes is only used by the image parser and does not need to be sent to the decoder.

The decoders required for the two parsing methods are different. For the first method the decoder directly follows the encoder with the output command replaced by an input command, similar to the original SPIHT algorithm.

The decoder for the second method additionally needs to keep track of the various lists (LIS, LIP, LSP) for all resolution levels greater or equal to the required one, It can recover all information for updating the lists during sorting pass of each quantization level (bit plane) at each spatial resolution level. The only additional information required by the decoder is the maximum number of spatial scalability levels (K max) supported by the encoder. Note that it is also possible to modify the encoder to directly produce the bit stream for the second decoding method.

3.6 PERFORMANCE EVALUATION

Compression efficiency is measured by the compression ratio and is estimated by the ratio of the original image size over the compressed data size. The *complexity* of an image compression algorithm is calculated by the number of data operations required to perform both encoding and decoding processes. Practically, it is sometimes expressed by the number of operations. For a lossy compression scheme, a *distortion measurement* is a criterion for determining how much information has been lost when the reconstructed image is produced from the compressed data. The most often used measurement is the mean square error MSE.

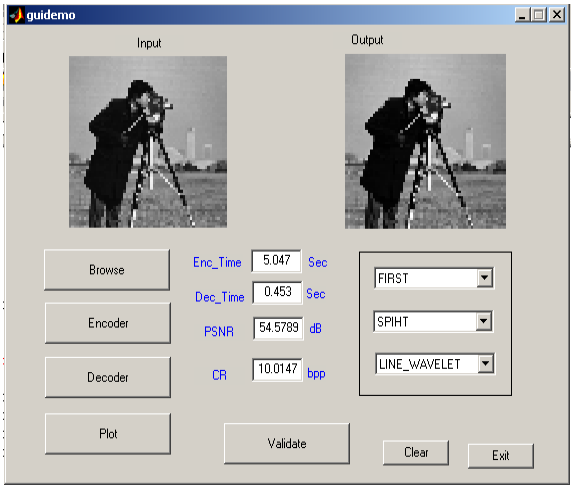
In the MSE measurement the total squared difference between the original signal and the reconstructed one is averaged over the entire signal. Mathematically,

$$\text{MSE} \quad \frac{1}{N}\sum_{i=0}^{N-1}(\hat{x}_i - x_i)^2 \qquad (22)$$

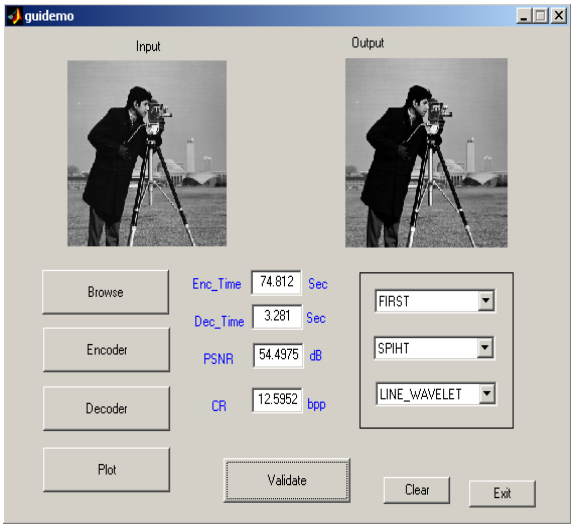
Where $\hat{x}_i$ is the reconstructed value of x_i, N is the number of pixels. The mean square error is commonly used because of its convenience. A measurement of MSE in decibels on a logarithmic scale is the Peak Signal-to-Noise Ratio (PSNR), which is a popular standard objective measure of the lossy codec. We use the PSNR as the objective measurement for compression algorithms throughout this paper. It is defined as follows.

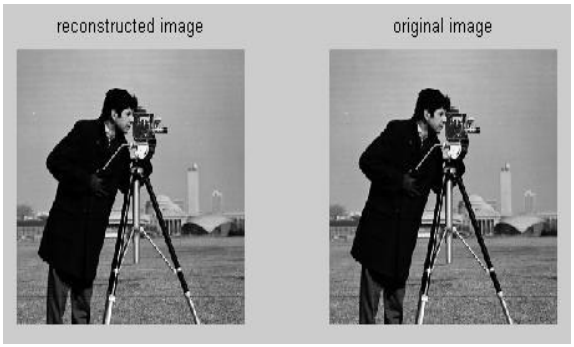
$$\text{PSNR} = 10 \log_{10} [255^2 / \text{MSE}] \qquad (23)$$

RESULTS (GUI IMAGE COMPRESSION)

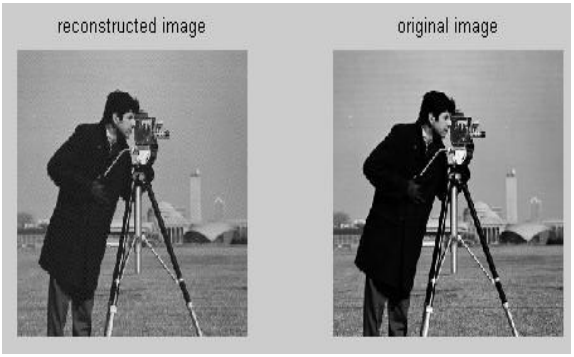


4.1 FIRST LEVEL (SPIHT)

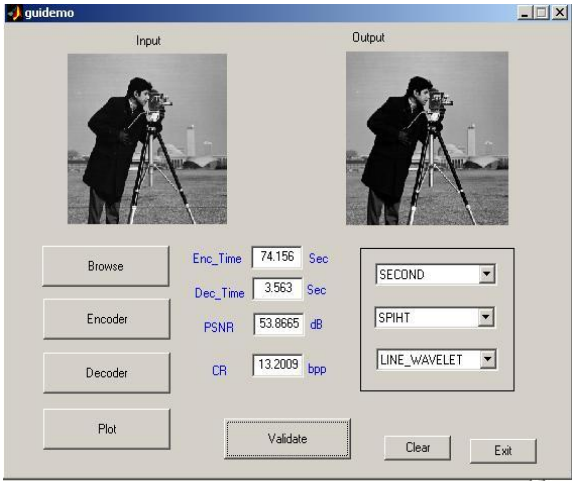




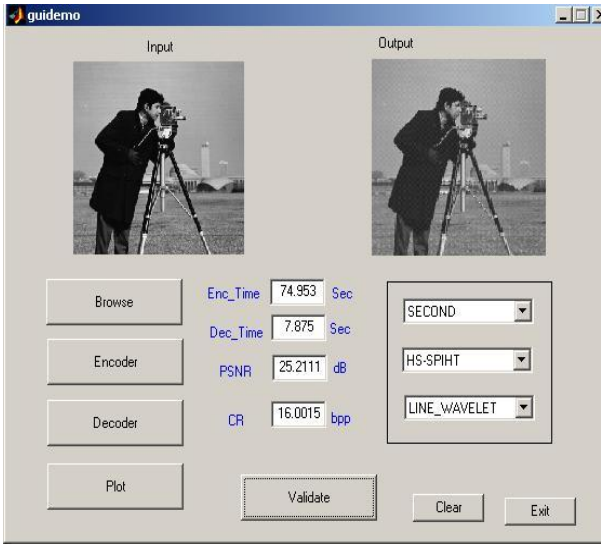
4.2FIRST LEVEL (HS-SPIHT)



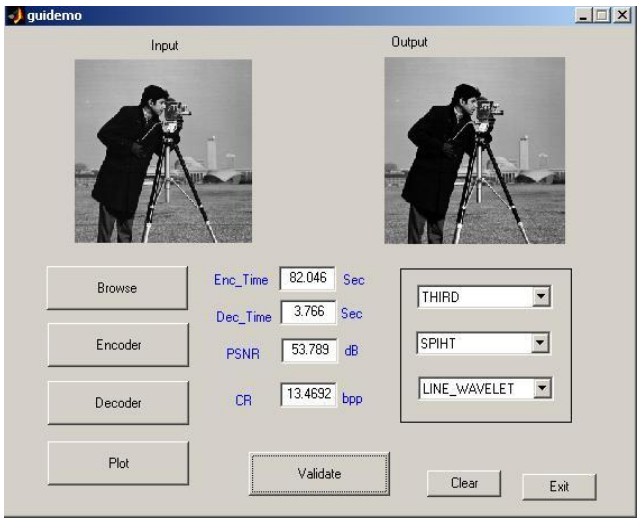
4.3 SECOND LEVEL (SPIHT)

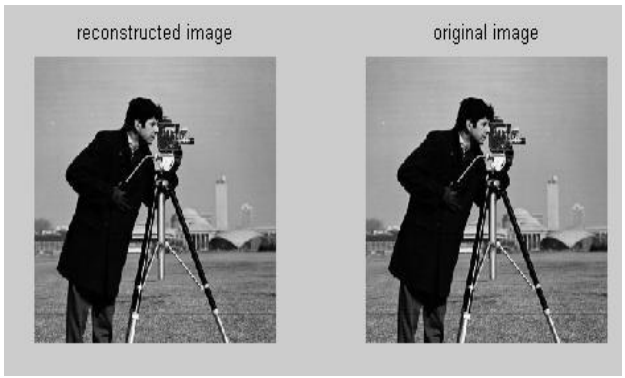


4.4 SECOND LEVEL (HS- SPIHT)

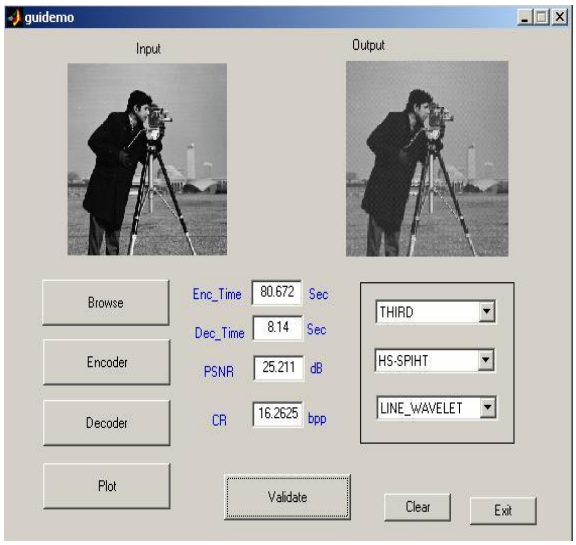


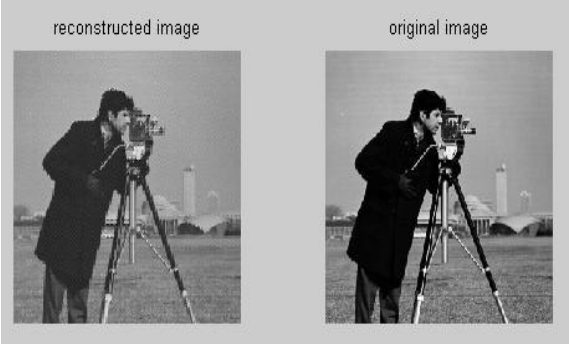
4.6 THIRD LEVEL (SPIHT)





4.7 THIRD LEVEL (HS-SPIHT)





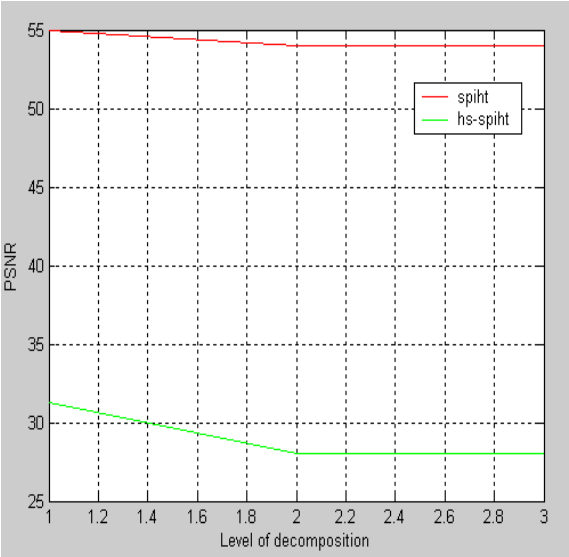


Fig.13. PSNR Vs level of decomposition

CONCLUSION AND FUTURE WORK

In this paper, the improved lifting scheme is proposed to replace Mallat algorithm in the line-based wavelet image coding. By utilizing the three-adder unit and the in-place character of the lifting scheme, lower memory and computation costs have been achieved. According to the lack of global information during the line-based wavelet image coding, the corresponding structure has been designed to build up the context, the memory and computation has been reduced further. We have presented a highly scalable SPIHT algorithm that produces a bit stream which supports spatial scalability and can be used for multiresolution transcoding. This bit stream not only has spatial scalability features but also keeps the full SNR embeddedness properly for any required resolution level after a simple reordering which can be done in a transcoder without decoding the bit stream. The embeddedness is so fine granular that almost each additional bit improves the quality and the bit stream can be stopped at any point to meet a budget during the coding or decoding process. The proposed multiresolution image codec is a good candidate for multimedia applications such as image storage and retrieval systems, progressive web browsing and multimedia information transmission, especially over heterogeneous networks where a wide variety of users need to be differently serviced according to their network access and data processing capabilities. In this paper we have developed a technique for line based wavelet transforms. We pointed out that this s transform can be assigned to the encoder or the decoder and that it can hold compressed data. We provided an analysis for the case where both encoder and decoder are symmetric in terms of memory needs and complexity. We described highly scalable SPIHT coding algorithm that can work with very low memory in combination with the line-based transform, and showed that its performance can be competitive with state of the art image coders, at a fraction of their memory utilization. To the best of our knowledge, our work is the first to propose a detailed implementation of a low memory wavelet image coder. It others a significant advantage by making a wavelet coder attractive both in terms of speed and memory needs. Further improvements of our system especially in terms of speed can be achieved by introducing a lattice factorization of the wavelet kernel or by using the lifting steps. This will reduce the computational complexity and complement the memory reductions mentioned in this work

REFERENCES

[1] C. Chrysas and A. Ortega, \Line Based Reduced Memory Wavelet Image Compression," in Proc. IEEE Data Compression Conference, (Snowbird, Utah), pp. 398{407, 1998.

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

[3] D. Lee, \New work item proposal: JPEG2000 image coding system." ISO/IEC JTC1/SC29/WG1 N390, 1996.

[4] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Lossless image compression using integer to integer wavelet transforms. In International Conference on Image Processing (ICIP), Vol. I, pages 596–599. IEEE Press, 1997.

[5] Uytterhoeven G. and D. Roose, A. Bultheel. WAVELET TRANSFORMS USING THE LIFTING SCHEME. Re port ITA-Wavelets-WP1.1, Department of Computer Science. Leuven: Katholieke Universities Leuven, 1997.

[6] (Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *Journal of Fourier Analysis and Applications*, vol. 4, no. 3, pp. 247-269, 1998.)

[7] Danyali, H & Mertins, A, Highly scalable image compression based on SPIHT for network applications, Proceedings International Conference on Image Processing, 22-25 September 2002 , 1, 1-217 - 1-220. Copyright IEEE 2002.

[8] Rafael C.Gonzalez, RichardE.Woods, Stevenl.Eddins, *Digital Image Processing Using MATLAB*, Prentice-Hall, 2004.

[9] A. M. Tekalp, *Digital Video Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[10] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "*Image coding using wavelet transform*," IEEE Trans. Image Processing, vol. 1, pp. 205–220, Apr. 1992

Results Comparison TABLE.1

	PSNR			CR		
TYPE	LEVEL 1	LEVEL 2	LEVEL 3	LEVEL 1	LEVEL 2	LEVEL 3
SPIHT	54.4 db	53.66 db	53.7 db	12.5	13.2	13.46
HS-SPIHT	29.7 db	25.2 db	25.2 db	13.9	16	16.26