MEMORY EFFICIENT SET PARTITIONNING IN HIERARCHICAL TREE (MESH) FOR WAVELET IMAGE COMPRESSION

*Harish Arora, Pramit Singh, Ekram Khan**
Department of Electronics Engineering
Aligarh Muslim University, Aligarh, India
**ekhan@lycos.com*

Farid Ghani
School of Electronics and Electrical Engg.
Universiti Sains Malaysia,
14300 Nibong Tebal, Penang, Malaysia

ABSTRACT

This paper presents a memory efficient version of set partitioning in hierarchical tree (SPIHT). The proposed coder termed as **M**emory **E**fficient **S**PI**H**T (MESH), uses a single re-usable list instead of three continuously growing linked lists as in conventional SPIHT. The list is re-initialized at the beginning of each bit-plane (coding pass) and is exhausted within that bit-plane itself. Another feature of the proposed coder is that it uses a single pass for each bit-plane by merging the sorting and refinement passes of conventional SPIHT together. The re-initialization of list in each bit-plane makes the proposed coder inherently error resilient. The performance of the proposed coder is measured in terms of coding efficiency and the worst case memory requirements for list entries in each bit-plane. The performance comparison with SPIHT shows that the proposed algorithm results in 50-70% memory saving while retaining the coding efficiency comparable to SPIHT.

1. INTRODUCTION

The SPIHT algorithm [1] is a fast and efficient state-of-art technique for image compression. Like EZW [2] , SPIHT generally operates on an entire image at once. The whole image is loaded and transformed, and then the algorithm requires repeated access to all coefficient values. There is no structure to the order in which the coefficient values are accessed. The random coefficient access requirement of the SPIHT algorithm hinders its use in certain memory-constrained environments.

The capability to encode a large image without storing the entire image in memory was an important feature in the JPEG2000 requirements specification. Though SPIHT [1] was a strong contender for JPEG2000, but due to the use of pixel/set lists that significantly increase working memory requirements, it was subsequently rejected. Since then the need for reducing the memory uses in zero-tree like algorithms is realized. Embedded block coding with truncation (EBCOT) [3] , which is included in JPEG2000, restricts the memory usage through encoding of pixel blocks but is of higher complexity due to the use of adaptive arithmetic coding, multiple coding passes & use of rate distortion optimizers.

The No List SPIHT (NLS) [4] is developed as a list free version of SPIHT. It uses fixed size arrays in-place of lists and the working memory is always fixed, independent of the number of passes to be executed. Another extension of the SPIHT algorithm in memory constrained environment is proposed by Pearlman [5] , in which coefficients are grouped in many small spatial blocks in such a way that each hierarchical coefficient tree appears in one of the spatial blocks. The basic SPIHT algorithm is applied to each spatial block independently before making the final bitstream, which no longer remains progressive.

In this paper, our emphasis is to reduce the working memory of the SPIHT algorithm by using a single re-usable list. The proposed algorithm termed as **M**emory **E**fficient **S**PI**H**T (MESH) uses a single list that is initialized, updated and exhausted within the same bit-plane. Within each coding pass, the list is initialized with the wavelet coefficients of the LL-subband having descendents. The significant coefficients corresponding to a threshold are obtained by using the set-partitioning rule, similar to that in SPIHT, but in a single pass by merging the sorting and refinement passes of SPIHT together.

The remainder of the paper is organized as follows. Section 2 contains a brief overview of the SPIHT algorithm. The proposed MESH algorithm is described in section 3. The working memory requirement of this coder is calculated and compared with that of the SPIHT and NLS algorithms in section 4. Experimental results are presented in Section 5 followed by concluding statements and future scope in Section 6.

2. SPIHT

The SPIHT consists of two main stages namely sorting and refinement. For practical implementation, SPIHT

maintains three linked lists viz. the list of insignificant pixels (LIP), the list of significant pixels (LSP) and the list of insignificant sets (LIS). At the initialization stage, SPIHT initializes the LIP with all the pixels in the highest level of the pyramid (i.e. LL subband), the LIS with all the coefficients at the highest level of the pyramid except those, which don't have descendents, and LSP as an empty set. During the sorting pass, the algorithm first traverses through the LIP, testing the magnitude of its elements against the current threshold and representing their significance by 0 or 1. Whenever a coefficient is found significant, its sign is coded and it is moved to LSP. The algorithm then examines the LIS and performs a magnitude check on all coefficient of set. If a particular tree/set is found to be significant, it is partitioned into its subsets (children and grandchildren) and tested for significance. Otherwise a single bit is appended to the bit stream to indicate an insignificant set (or zero-tree). After each sorting pass SPIHT outputs refinement bits at the current level of bit significance of those pixels which had been moved to LSP at higher threshold, resulting in the refinement of significant pixels with bits that reduce maximum error. This process continues by decreasing current threshold by factor of two until desired bit rate is achieved.

Some of the drawbacks of SPIHT are as obvious. Firstly, lists are initialized only once in the beginning and they continue to grow during the encoding and decoding process. This increases the memory requirements. Secondly, once an element enters into LIP, at least one 'zero' bit per bit-plane is generated until it becomes significant and is transferred to LSP. Thirdly, encoding and decoding of a bit-plane depend a lot on the previous bit-planes, as lists updated in previous bit-planes are used in subsequent bit-planes which cause the error propagation throughout the image.

3. MEMORY EFFICIENT SPIHT (MESH)

In this paper we propose a novel modification in SPIHT algorithm such that the function of all three lists can be performed by a single list. The list is re-initialized at the beginning of each bit-plane to avoid the inter-dependency on the previous bit-planes. The algorithm works as follows.

After the 'N_d' level of wavelet decomposition of input image, depending upon the value of the largest wavelet coefficient, the initial threshold is calculated as 2^n where $n = \left\lfloor \log_2 \left(\max_{\forall(i,j)} |c_{i,j}| \right) \right\rfloor$. The LL-sub-band is coded separately. The coefficients of LL-sub-band are represented in (n+1) bit sign-magnitude form. In the first (most significant) bit-plane, the sign bit and the most significant bit from magnitude of the LL sub-band coefficients are embedded in the bitstream. In the subsequent bit-planes, only the n^{th} most significant bits of LL-sub-band coefficient are transmitted. In order to encode the remaining sub-band coefficients, they are linked through spatial orientation trees with their roots in the LL-sub-band. To define a uniform child-parent relationship, it is assumed that first quarter of LL-sub-band coefficients have no descendents and remaining three-quarter of the coefficients have their children in sub-bands of correspondingly same orientation.

A single list referred as **L**ist of **P**ixel **S**ets (LPS) is used in this algorithm. At the beginning of each bit-plane, LPS is initialized with address of those coefficients of LL-subband that have decedents. For each entry in LPS, test the significance of the set. For insignificant sets, a single bit is appended to the bitstream to indicate a zerotree. However, if a particular set is found to be significant, it is then partitioned into two its subsets (children and grandchildren). First the significance of each of the children is tested and if a particular child is found significant, its sign bit is transmitted and current threshold is subtracted from its magnitude. Then the significance of grandchildren is tested. If the set of grandchildren (provided they exist) is found to be significant, the addresses of children are added at the end of LPS. After testing every entry of LPS (including those which are added in the current pass), it is re-initialized and threshold is reduced by a factor of two. The entire process is repeated for the next pass (bit-plane). It should be noted that same coefficients might become significant in more than one pass, depending upon its absolute value. In order to avoid the multiple transmission of its sign bit, sign bits are masked (or flagged) once they are transmitted at the first significant test of the corresponding coefficient. If in the subsequent passes, the same coefficient again tests significant at lower thresholds, the transmission of sign bit is avoided as it has already been masked.

At the decoder, the reverse process is performed. At the arrival of first significant bit of a coefficient, it is reconstructed as $\pm 1.5 \times 2^n$. The decoder adds or subtract 2^{n-1} to its current reconstructed value depending upon whether it inputs significant bit in the current pass or not.

It can be observed that the working memory requirement in any pass is proportional to the maximum number of entries in LPS during that coding pass. Further, since in MESH, LPS is re-initialized, the same memory can be used in the next and subsequent passes, whereas in SPIHT, the lists keep growing progressively thereby increasing memory requirements. Additionally, the entire encoding and decoding of a bit-plane is performed in a single pass, whereas SPIHT requires two passes, sorting and refinement. Another attractive feature of the MESH

algorithm is its improved error resilience. As the encoding pass is independent of the previous passes, if an error occurs in any pass, it will affect the decoding of that pass only.

4. MEMORY ANALYSIS

In this section we will compare the memory requirements of the auxiliary list(s) in MESH with that of SPIHT and NLS algorithms. There are two types of memory requirement in zerotree coders: fixed memory to store wavelet coefficients and variable working memory to store list entries. For the comparison purpose, we consider only variable working memory requirements. Though NLS does not use any lists, but needs fixed size arrays and state tables. For an image of size $M \times N$, the working memory requirement in NLS [4] is $M_{NLS} = \left(\frac{MN}{4} + \frac{MN}{16}\right)W + \frac{MN}{2}$ bytes, where W is the number of bytes used to store each wavelet coefficient (say two bytes). This working memory is always fixed and is independent of number of bit-planes to be executed. However, in SPIHT and the proposed coder, the size of required working memory at any instant depends upon the current number of entries in the list(s). For the purpose of comparisons, memory requirements at the end of each pass (or bit-plane) and the maximum (or worst case) memory requirement are considered.

4.1 SPIHT

In SPIHT three linked lists are used namely LIP, LSP and LIS. Each entry in LIP and LSP is a single coordinate of a wavelet coefficient whereas LIS also requires type (A or B) information to distinguish nodes.

Let:

N_{LIP} = number of entries in LIP.

N_{LSP} = number of entries in LSP.

N_{LIS} = number of entries in LIS.

c= number of bits to store addressing information of a coefficient [$2*\log_2(\max(M,N))$]

M_{SPIHT}= total memory required in SPIHT (in bits).

Then

$$M_{SPIHT} = c\, N_{LIP} + (c-1)\, N_{LIS} + c\, N_{LSP} \quad (1)$$

Where each element in LIS requires (c-2) bits for addressing, since it contains coefficients with descendents and an extra bit is required for defining the 'type' of entries.

In the worst case,

$N_{LIP} + N_{LSP} = MN$

$N_{LIS} = MN/4$ (as the coefficients having no descendents or the highest frequency sub-bands will never enter into LIS.) Thus the maximum working memory requirement in SPIHT is

$$M_{SPIHT}^{\max} = (5c-1)\frac{MN}{4} \quad (2)$$

4.2 MESH

In our proposed algorithm, we have used only one list namely LPS. In the list, as entries correspond to address of a wavelet coefficient (except the coefficients of LL-sub-band and those which have no offsprings), so each element can be stored by using only (c-2) bits. If N_{LPS} be the numbers of entries in LPS and M_{MESH} (in bits) denotes the total working memory required in the MESH algorithm, then

$$M_{MESH} = (c-2)\, N_{LPS} \quad (3)$$

In the worst case,

$N_{LPS} = MN/4$

Thus, the maximum working memory requirement in MESH is $M_{MESH}^{\max} = (c-2)\frac{MN}{4}$ (4)

For example, for any 512×512 image, $c=2*\log_2(512)=18$ bits, W= 2 bytes (say),

$M_{SPIHT}^{\max}$ = 729088 bytes

$M_{NLS}^{\max}$ = 294912 bytes

$M_{MESH}^{\max}$ = 131072 bytes

Thus, $M_{SPIHT}^{\max} : M_{NLS}^{\max} : M_{MESH}^{\max} = 5.56 : 2.25 : 1$. It should be noted that NLS uses fixed working memory equal to $M_{NLS}^{\max}$. Thus it can be seen that the proposed MESH algorithm has reduced memory requirement by factors of 5.56 and 2.25 in comparisons to SPIHT and NLS respectively.

5. SIMULATION RESULTS

In order to compare the performance (in terms of memory requirements and coding efficiency) of MESH, experiments are performed for the 512×512 grayscale *Lena* and *Barbara* images. For these experiments, we used a five level decomposition with the 9/7 biorthogonal filter [6] .

5.1 Memory Analysis:

The bit-plane wise working memory requirement of the MESH and SPIHT algorithms for the first nine passes of 'LENA' image is compared in Fig. 1. At the end of each pass, based on the number of entries in the corresponding list(s), M_{SPIHT} and M_{MESH} are evaluated according to Eqn. 1 and 3 respectively for c=18 and are plotted in this figure. It can be seen that the MESH algorithm has considerably reduced working memory requirement in comparison to that of SPIHT.

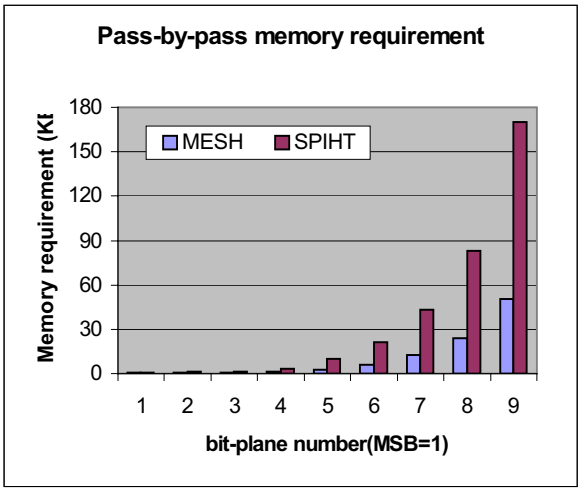

Fig. 1: Pass-wise memory requirement of MESH and SPIHT

5.2 Rate-distortion Analysis:

The results in terms of PSNR of decoded image at various bit-rates (up to 1.0 bpp) are compared with that of SPIHT. These results are shown in Figs. 2 & 3 for *Lena* and *Barbara* images respectively. It should be noted that these results are without the use of any arithmetic coding. As it can be seen that MESH has comparable performance to that of SPIHT for Lena image, however has slightly inferior coding efficiency for Barbara image at higher bit-rates. Despite this negligible degradation of coding efficiency, still the MESH is profitable over SPIHT due to significant memory saving at higher bit-rates.

6. CONCLUSIONS

In this paper, we have presented a novel memory efficient variant of SPIHT for wavelet image coding. The proposed coder generates fully embedded bit-stream similar to SPIHT, but performs only one pass in each bit-plane, as against two passes of SPIHT. The most attractive feature of the coder is its high memory efficiency while rate-distortion performance comparable to that of SPIHT. It uses only one list which is initialized at the beginning of each bit-plane, updated and exhausted within the same bit-plane. This makes the generated sub-bitstream of each bit-plane independent from the other bit-planes. Thus the proposed coder is inherently error resilient. This is in contrast to SPIHT, in which each bit-plane uses the same list updated from the previous bit-planes & hence has the poor error resiliency. In future, we aim to extend this algorithm for color image and video coding.

7. REFERENCES

[1] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchal trees. *IEEE Trans. on Circuits and Systems for Video Technology*, 6(3): pp. 243–250, June 1996.

[2] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. on Signal Processing*, 41(12): pp. 3445–3462, December 1993.

[3] D. Taubman, "High performance scalable image compression with EBCOT", *IEEE Transaction on Image Processing*, Vol. 9, pp. 1158-1170, July 2000.

[4] F. W. Wheeler and W. A. Pearlman, "SPIHT image compression without lists", *IEEE International Conference on Acoustics, Speech and Signal Processing, ICAASP 2000*, Vol. 6, pp 2047-2050, September 2000.

[5] F. W. Wheeler and W. A. Pearlman, "Low memory packetized SPIHT image compression", *33rd Asilomar Conference on Signals, Systems and Computers*, Vol. 2, pp 1193-1197, October 1999.

[6] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Trans. on Image Processing*, Vol. 1, No. 2, April 1992.

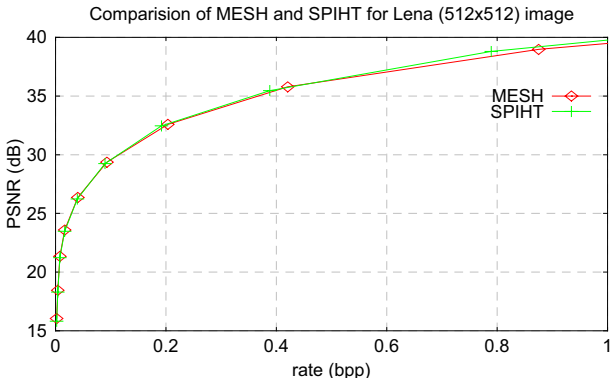

Fig. 2: Rate-distortion performance comparison of MESH and SPIHT for Lena image

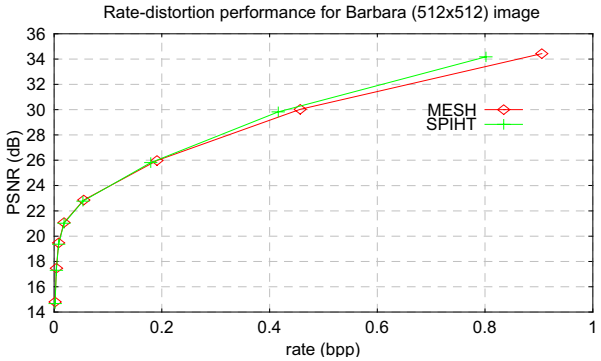

Fig. 3: Rate-distortion performance comparison of MESH and SPIHT for Barbara image