International Journal of Image Processing and Vision Science

Volume 1 | Issue 4 Article 10

October 2013

MODIFICATION TO SPIHT ALGORITHM USING INCREMENTAL THRESHOLD FOR IMAGE COMPRESSION

A. S. JADHAV
ECE department, BLDEA's, Dr. P. G. Halakatti College of Engineering and Technology Bijapur, Karnataka, INDIA., asjadhavec@gmail.com

RASHMI V. PAWAR
EEE department, BLDEA's, Dr. P. G. Halakatti College of Engineering and Technology Bijapur, Karnataka, INDIA., rashmi.ajadhav@gmail.com

MODIFICATION TO SPIHT ALGORITHM USING INCREMENTAL THRESHOLD FOR IMAGE COMPRESSION

A. S. JADHAV[1] & RASHMI V. PAWAR[2]

[1] ECE department, BLDEA's Dr. P. G. Halakatti College of Engineering and Technology Bijapur, Karnataka, INDIA.
[2] EEE department, BLDEA's Dr. P. G. Halakatti College of Engineering and Technology
Bijapur, Karnataka, INDIA.
Email: asjadhavec@gmail.com, rashmi.ajadhav@gmail.com

Abstract—The Modified SPIHT represents a more efficient implementation of the SPIHT algorithm by using variable thresholds to sort the list of insignificant pixels (LIP) and the list of insignificant sets (LIS). We observe two interesting facts: (I) most of the initial subsets in LIS are not only insignificant with respect to the maximum threshold, but also insignificant with respect to the smaller threshold. And (2) Most of the pixels generating from sorting LIS are smaller than the current threshold. Based on these two observations, it represents a new image codec method, which can make the binary encoded outputs more efficient, and can work well on different image sizes and different decomposition levels.

Keywords— *Compression, SPIHT, PSNR, Sub-band coding, EZW Algorithm*

I. INTRODUCTION

Said and Pearlman developed a simple and efficient embedded image coding system based on set partitioning in hierarchical trees (SPIHT) concept. This concept uses three lists to store the significance information: list of insignificant sets (LIS), list of insignificant pixels (LIP), and list of significant pixels (LSP). At every quantization level, it partitions wavelet coefficients into these three lists. Such subset partitioning is so effective and the significance information is so compact that even binary encoded transmission achieves about the same or better performance than almost all previously existing schemes. In the case of 5 level decomposition for 512 x 512 images, the binary encoded bit stream is so efficient that using arithmetic coding can only increase peak signal-to-noise ratio (PSNR) by 0.3 - 0.6 dB for the same bit rate. But, for lower level decomposition with the same image size, it becomes not so compact any more. This is because the number of pixels in low-low band is still large compared to the whole image so that there is lot of time spent in many bits on sorting LIS and LIP at the beginning. For those images with very small size or very large size, this could not make the low-low band very small.

For small sized images, although the low-low band can be decomposed into very small size, it is still quite big as compared to the whole image. On the other hand, for large sized images, if it uses 16 bits integer to save the wavelet coefficients, then it could not do the wavelet decomposition too many levels because overflows would occur otherwise. The outputs will still be very efficient. However, arithmetic coding may not be the best solution as it will largely increase the encoding and decoding time. In fact, it is observed that most of the initial subsets in LIS, that is, the pixel sets with roots in the highest pyramid level are not only insignificant with respect to the initial threshold, but also insignificant to a smaller threshold.

Therefore, when the low-low band is not very small as compared to the whole image, this redundancy becomes one main reason that makes the binary encoded outputs not efficient. Even for normal sized images with high level decomposition, there still remains some redundancy in the binary encoded outputs. It is found that most of the new pixels generated from sorting LIS are smaller than the current sorting threshold T.

The implementation includes modified SPIHT algorithm based on using variable sorting thresholds. The new algorithm can make the binary encoded outputs more efficient on different sized images and different decomposition levels.

II. BASIC SPIHT ALGORITHM

The SPIHT algorithm [1] uses a partitioning of the spatial orientation trees in a manner that tends to keep insignificant coefficients together in larger subsets. The partitioning decisions are binary decisions that are transmitted to the decoder, providing a significance map encoding. The thresholds used for checking significance are powers of two. So in essence the SPIHT algorithm sends the binary representation of the integer value of the wavelet coefficients. The significance map encoding or set partitioning and ordering step is followed by a refinement step in which the representations of the significant coefficients are refined. The SPIHT algorithm can be applied to both grey-scale and colored images. SPIHT displays exceptional characteristics over several properties like good image quality, fast coding and decoding, a fully progressive bit stream, application in lossless

compression, error protection and ability to code for exact bit rate.

The SPIHT process represents a very effective form of coding. A straightforward consequence of the compression simplicity is the greater coding/decoding speed. The SPIHT algorithm is nearly symmetric, i.e., the time to encode is nearly equal to the time to decode. SPIHT codes the individual bits of the image wavelet transform coefficients following a bit plane sequence. It is capable of recovering the image perfectly by decoding all these bits. In practice it is possible to recover the image perfectly using rounding after recovery. Due to its embedded coding property, SPIHT is much easier to design for efficient error-resilient schemes. This is because the embedded coding has the information sorted according to its importance and the requirement for powerful error correction codes decreases from the beginning to the end of the compressed file. If an error is detected, but not corrected, the decoder can discard the data after point and still display the image obtained with the bits received before the error. Another reason is that SPIHT generates two types of data. The first is sorting information, which needs error protection. The second consists of uncompressed sign and refinement bits, which do not need special protection because they affect only one pixel.

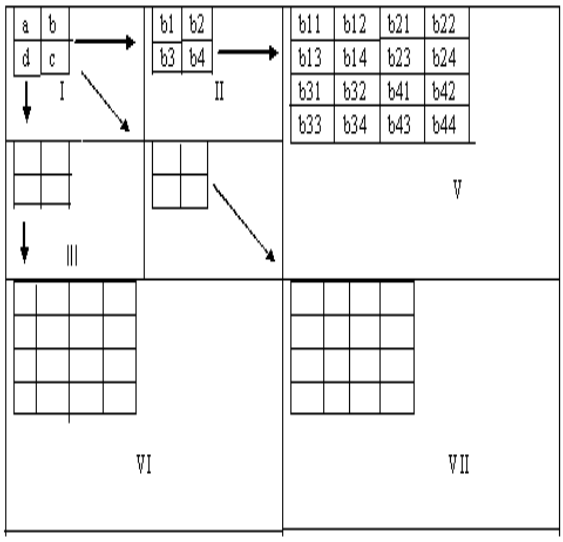

Fig. 1. Seven band Decomposition of SPIHT

In SPIHT algorithm, the wavelet coefficients [2] are divided into trees originating from the lowest resolution band. The coefficients are grouped into 2-by-2 arrays, except for the coefficients in band 1, which are offspring of a coefficient of a lower resolution band. The coefficients in the lowest resolution band are also divided into 2-by-2 arrays.The coefficient in the top-left corner of the array does not have any offspring and is known as the root node. This data structure is shown pictorially in Figure 1 for seven-band decomposition.

III. METHOD USED FOR MODIFYING THE SPIHT ALGORITHM

The following method is used to modify the SPIHT algorithm so as to get the good PSNR and MSE values.

3.1) Using Variable Thresholds to Sort the Initial Subsets

The sorting threshold in the SPIHT algorithm is always fixed within the same sorting round, which means that the sorting threshold starts from the maximum threshold Tm and then will be divided by 2 for each new sorting round. It is well known that most of an image's energy is concentrated in the low frequency components. In the meantime, because of the spatial self-similarity between subbands, a few largely valued pixels are concentrated in certain subsets; while other subsets only have small valued pixels. As a result, in several early sorting rounds, most of the initial subsets, that is, those subsets with roots in the highest pyramid level (Fig.3.1a), will be insignificant. As the sorting round continues, the sorting threshold become smaller and smaller, and some initial subsets will then become significant.

Table 3.1 shows some statistical results for the number of significant initial subsets with respect to a chosen threshold T. Here, we used 512 x 512 image Lena with 3 level decomposition.

TABLE 3.1.
NUMBER OF SIGNIFICANT INITIAL SUBSETS AT THRESHOLD T

T	512	256	128	64	32
Number	0	26	240	423	498
T	16	8	4	2	1
Number	509	714	637	25	0

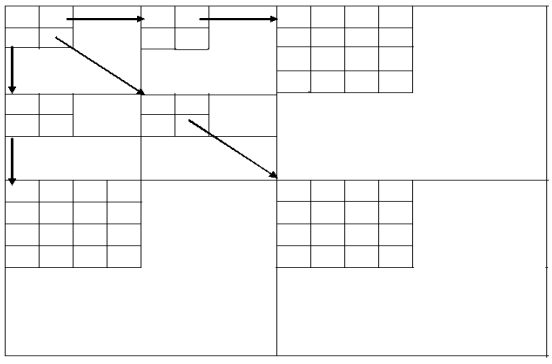

Fig. 3.1 (a) Examples of the initial subsets in LIS

Algorithm

1. Initialize T=1

2. If all the pixels are less than T record them.
3. Else if the pixels are greater than T increase the threshold by a factor of 2.
4. Continue the above procedure until all the pixels are insignificant to T

In Table 3.1**.** 'Number' represents the number of significant initial subsets when the sorting threshold is set at value T. Here, the maximum threshold Tm is chosen to be 512. It is seen that all of the initial subsets are insignificant with respect to this Maximum threshold. Actually, the significant initial subsets can spread at the entire sorting thresholds, from Tm to 1.

From these statistic results, we can see that sorting these initial subsets from Tm is not a good choice. So, it starts sorting these initial subsets from a smaller threshold T_0 directly. Using this method saves many bits and can largely increase PSNR, particularly at low bit rates.

To deal with those significant subsets, if it detects one initial subset significant to T_0, it has to increase the threshold value and check it again until it become insignificant to the latest threshold T_i. After that, it is then not necessary to test this subset because it is already known that it is significant with respect to threshold value T_i-1.

After finding these thresholds for every initial subset in LIS, it will not process them until the threshold of the current sorting round equals to their own threshold. Fig. 3.1 b shows the flow chart about how it sorts the initial subsets.

Let's make a comparison between the new sorting method and the old one. Suppose that we start sorting LIP from the maximum threshold T_m, and start sorting the initial subsets from T_0. After that, we can find directly those subsets smaller than T_0 (using 1 bit), and, at the same time, get all subsets Є [To, 2To) using 2 bits, all subsets Є [2To, 4To) using 3 bits, etc. Let T1 = 2To, T2 = 4T0, ..., Tn=Tm. Let Po, P1, ..., Pn be the probabilities of the initial subsets being only smaller than T_0, T_1, T_2, . . ., T_n, respectively. The average bits spent for sorting these initial subsets are:

$$1 \times p_0 + 2 \times p_1 + \ldots\ldots + (n+1) \times p_n = \sum_{i=0}^{n} (i+1) \times p_i \qquad 3.5$$

Where

$$n = \log_2 T_m - \log_2 T_0 \qquad 3.6$$

However, if we use the original method, the average bits are:

$$\sum_{i=0}^{n} (\log_2 T_m - \log_2 T_i + 1) \times p_i \qquad 3.7$$

Then equation (3.7) - (3.5**)** gives the average bits saved:

$$(3.7) - (3.5) = \sum_{i=0}^{n} (\log_2 T_m - \log_2 T_1 + 1 - i - 1) \times pi = \left(\frac{T_m}{T_o} - 2i \right) \times pi \qquad 3.8$$

For images with size 512 x 512 and L level decomposition, there are 3 x $(512 / 2^L)^2$/ 4 initial subsets. Thus, the total bits saved are:

$$Saved\ \ bits = \frac{3}{4} \times \left(512/2^L\right)^2 \times \sum_{i=0}^{n} \left(\log_2 \frac{T_m}{T_0} - 2i \right) \times p_i \qquad 3.9$$

There exists an optimal threshold for a particular image and it is not difficult to get it. The encoder should first compute P_1, P_2,. . ., P_n beforehand. Then, for every T_O, we can use equation (3.6) to find the optimal threshold that maximum (3.9). This procedure does not need much time. Since the saved bits for T_O being around the optimal threshold are still very large, the results will still be close to the optimal results even if we just select To smaller than Tm (not too small).

3.2) Using Variable Threshold to Sort the New Pixels

Except for the pixels in the highest pyramid level, all new pixels added to LIP and LSP are first generated from sorting LIS. These new pixels are generated in the following way: when subsets in LIS is found to be significant, it is removed from the list and partitioned, and its four offspring (the new pixels) are tested with the current threshold and added to the end of LIP or LSP, depending on whether they are insignificant or significant, respectively.

Since the encoder will spend many bits to sort these pixels, this step becomes very important to the coding efficiency. Our work shows that there is some redundancy in these pixels. Most of these pixels are insignificant with respect to the threshold that they are tested at the first time. Below is some test results for these pixels.

From Tables 2 it is clear that nearly one half of the new pixels are not only smaller than T, but also smaller than T/2. In fact, we have tested many images and they show similar statistics. So, it starts sorting from T/2 directly.

If a pixel is smaller than T/2, then it saves one bit. If not, test it again use T. If it is in the region [T/2, T), we do not use any extra bit. If it is significant to T, we have spent one more bit to sort this pixel. But, since most of the new pixels are smaller than T, we can therefore save many bits. In Tables 3.2, the 'gain' represents

TABLE 3.2
THE STATISTICS OF THE NEW PIXELS IN LENA IMAGE

T	128	64	32	16	8
∈ [T, 2T]	618	1333	2673	5046	10500
∈ [T/2, T]	431	958	1849	3627	8303
<T/2	1003	1709	3410	6507	14849
Gain	385	376	737	1461	4349

In the current sorting round, we have to spend one more bit for each pixel not smaller than T/2. At that time, the performance will decrease. But, in the next sorting round, it can save bits from those pixels

smaller than T/2. Overall, the coding efficiency will first decrease slightly, but increase subsequently. In proposed new algorithm, it uses T/2 to sort the new pixels in the beginning. However, when T is not very large, e.g. 32, we just use T to sort the new pixels generated from current sorting round. There are two more ways to improve the coding efficiency. One is that if we can first estimate which region the new pixel belongs to, that is, [T, 2T) or [T/2,T) or [O,T/2), by testing it using T or Ti, adaptively, the coding performance will then increase remarkably. The other is that it can design a filter bank that makes only a small part of the new pixels being smaller than T/2, it can also reduce many bits from sorting LIP and finally increase the performance.

IV. RESULTS AND DISCUSSIONS

4.1) Results of Lena Image using SPIHT and Modified SPIHT Algorithm

The results produced by implementation of SPIHT algorithm are shown below

(a) (b) (c)

Fig. 4.1(a) Lena Original Image (b) Decoded Lena image (SPIHT) (c) Decoded Lena image (Modified SPIHT

4.2) Comparison Results of SPIHT WITH Modified SPIHT

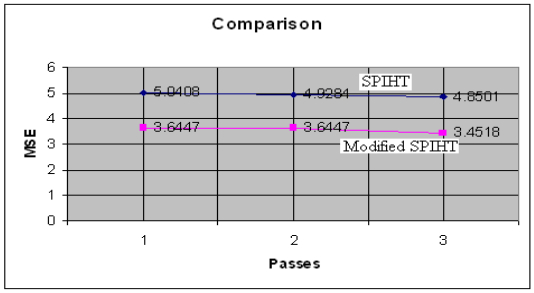

Fig. 4.2 Comparison of MSE Vs Passes (Lena)

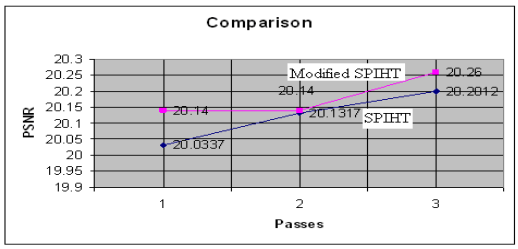

Fig. 4.3 Comparison of PSNR Vs Passes (Lena)

V. CONCLUSION

We have been presenting a new image-coding algorithm based on the well-known Set Partitioning in Hierarchical Trees (SPIHT), which has been demonstrated to be able to produce better coding performances. The unique feature of this new coding algorithm is to use different (variable) thresholds to sort pixels and initial subsets. Such new sorting procedure is fast and does not require complicated implementations. The outputs of this codec can be further entropy encoded, and therefore can achieve even higher coding performances.

REFERENCES

[1] R.ViswanathReddy, T. Sreenivasulu Reddy and Dr.Govind Sharma, "Efficient Coding of Image Sub bands using Block based Modified SPIHT" *International Journal of Recent Trends in Engineering, Vol 2, No. 5, November 2009*

[2] M. Santhi and Dr. R.S.D.Wahida Banu, "Modified SPIHT Algorithm for Coding Color Images using Inter-color Correlation" IJCSNS *International Journal of Computer Science and Network Security, VOL.10 No.3, March 2010.*

[3] W. R. Franklin and A. Said, "Lossy compression of elevation data", *Proceedings of Seventh International Symposium on Spatial Data Handling Aug. 1996. National Conference on Signal Processing and Communications-2006 (NCSPC- 2006).*

[4] Amir Said, William A. Pearlman, **"**A new, fast, and efficient Image codec based on set partitioning in hierarchical trees". *IEEE Trans. Circuits Syst. Video Technol., vol. 6, pp. 243-250, June 1996.*

[5] J. M. Shapiro, "Embedded image coding using zero trees of wavelets coefficients", *IEEE Trans. Signal Processing, vol. 41, pp. 3445-3462, Dec. 1993.*

[6] Anilkumar V. Nandi and Dr.R.M.Banakar, "Hardware modeling and implementation of modified SPIHT algorithm for compression of images". *Second International Conference on Industrial and Information Systems, ICIIS 2007, 8 – 11 August 2007, Sri Lanka.*

[7] Christopher M. Brislawn, "Preservation of Sub band Symmetry in Multirate Signal Coding", *IEEE Trans. Signal Processing, vol. 43, pp. 3046-3050 Dec. 1995.*

[8] Marc Antonini, Michel Barlaud, Pierre Mathieu and Ingrid Daubechies, "Image coding using wavelet transform", *IEEE Trans. Image Processing, vol. 1, April, 1992.*