IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 11, May 2010
ISSN (Online): 1694-0784
ISSN (Print): 1694-0814

Visually Improved Image Compression by Combining EZW Encoding with Texture Modeling using Huffman Encoder

Vinay U. Kale *, Shirish M. Deshmukh *

* Department Of Electronics & Telecomm. Engg.,

P. R. M. Institute Of Technology And Research, Badnera, Amravati,

Maharashtra, India, 444602.

Abstract

This paper proposes a technique for image compression which uses the Wavelet-based Image/Texture Coding Hybrid (WITCH) scheme [1] in combination with Huffman encoder. It implements a hybrid coding approach, while nevertheless preserving the features of progressive and lossless coding. The hybrid scheme was designed to encode the structural image information by Embedded Zerotree Wavelet (EZW) encoding algorithm [2] and the stochastic texture in a model-based manner & this encoded data is then compressed using Huffman encoder. The scheme proposed here achieves superior subjective quality while increasing the compression ratio by more than a factor of three or even four. With this technique, it is possible to achieve compression ratios as high as 10 to 12 but with some minor distortions in the encoded image.

Keywords: Image compression, Huffman encoder, Zero tree Wavelet

1. Introduction

The Embedded Zero tree Wavelet (EZW) algorithm proposed by Shapiro [2] is a simple, yet remarkably effective, image compression algorithm, having the property that the bits in the bit stream are generated in the order of importance, yielding a fully embedded code. An EZW encoder is an encoder specially designed to use with wavelet transforms, which explains why it has the word wavelet in its name.

Human observers are very sensitive to a loss of image texture in photo-realistic images. For example a portrait image without the fine skin texture appears unnatural. Once the image is decomposed by a wavelet transformation, this texture is represented by many wavelet coefficients of low- and medium-amplitude. The conventional encoding of all these coefficients by using EZW encoder is very bit rate expensive, because, although textured regions may exhibit high autocorrelation at larger scales (i.e., for macro-textures), they often show very little correlation at pixel-level. This has led to the idea that textured regions should be encoded using specially adapted techniques. Instead of using pixel-based encoding, texture-models could be used to describe such regions. It would then be sufficient to transmit to the decoder only the model parameters, which are usually small in number. With these parameters, the decoder can reconstruct a texture that looks very similar to the original one.

Instead of encoding the entire image using the EZW algorithm, the unstructured or stochastic texture in the image is modeled by a noise process and characterized with very few parameters. With these parameters, the decoder reconstructs a texture that looks very similar to the original one. The texture-modeling task imposes negligible computational complexity and the model parameters would require only a few bits per pixel to be encoded. Thus, by combining EZW encoding with texture modeling the coding costs are greatly reduced. Huffman encoding is an entropy-encoding algorithm which gives lossless data compression. It uses a variable-length code table for encoding the symbols generated by EZW algorithm where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the symbol. Nevertheless, this scheme delivers very good results. This paper is an attempt in that direction.

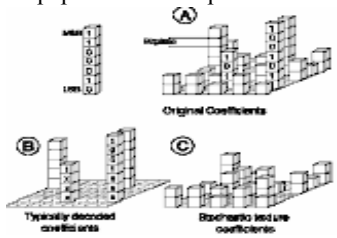

Fig.1: Wavelet coefficients are encoded bit plane by bit plane

2. Texture Modeling

Embedded Zero tree Wavelet (EZW) encoder, encode the wavelet coefficients bit plane by bit plane. First, the most significant bits of the largest wavelet coefficients are encoded. Then, progressively the large coefficients are refined and smaller coefficients are encoded as significant. Fig.1 (A) shows such a bit plane representation of wavelet coefficients. Decoded at a decent bit rate, the largest coefficients get restored [Fig.1 (B)] and the rest are quantized to zero. These remaining coefficients nevertheless carry visually important texture information. Quantizing them to zero leads to visually discernible blurring of regions with *stochastic* textures. The WITCH-scheme [1] encodes this stochastic texture information, represented by the wavelet coefficients in the last 2 or 3 bit planes [Fig. 1(C)] using a model-based approach, and combines it in a smooth manner with the conventionally decoded portions of the image.

3. EZW Encoding

The EZW algorithm is based on four key concepts: 1) a discrete wavelet transform or hierarchical sub band decomposition, 2) prediction of the absence of significant formation across scales by exploiting the self-similarity inherent in images, 3) entropy-coded successive-approximation quantization, and 4) universal lossless data compression which is achieved via adaptive Huffman encoding.

The EZW encoder was originally designed to operate on images (2D-signals) but it can also be used on other dimensional signals. The EZW encoder is based on progressive encoding to compress an image into a bit stream with increasing accuracy. This means that when more bits are added to the stream, the decoded image will contain more detail, a property similar to JPEG encoded images. Using an embedded coding algorithm, an encoder can terminate the encoding at any point thereby allowing a target rate or target accuracy to be met exactly. Also, given a bit stream, the decoder can cease decoding at any point in the bit stream and still produce exactly the same image that would have been encoded at the bit rate corresponding to the truncated bit stream. In addition to producing a fully embedded bit stream, EZW consistently produces compression results that are competitive with virtually all known compression algorithm on standard test images It is similar to the representation of a number like π (pi). Every digit we add increases the accuracy of the number, but we can stop at any accuracy we like. Progressive encoding is also known as embedded encoding, which explains the E in EZW.

3.1 The Concept of Zerotree

To improve the compression of *significant maps* (binary decisions as to whether a sample, i.e. a coefficient of a 2-D discrete wavelet transform, has a zero or nonzero quantized value) of wavelet coefficients, a new data structure called a *Zerotree* is defined.

A wavelet coefficient x is said to be *insignificant* with respect to a given threshold T
if $|x| < T$. The Zerotree is based on the hypothesis that if a wavelet coefficient at a coarse scale is insignificant with respect to a given threshold T, then all wavelet coefficients of the same orientation in the same spatial location at finer scale are likely to be insignificant with respect to T. Empirical evidence suggests that this hypothesis is often true.

More specifically, in a hierarchical sub band system, with the exception of the highest frequency sub bands, every coefficient at a given scale can be related to a set of coefficients at the next finer scale of similar orientation. The coefficient at the coarse scale is called the *parent*, and all coefficients corresponding to the same spatial location at the next finer scale of similar orientation are called *children*. For a given parent, the set of all coefficients at all finer scales of similar orientation corresponding to the same location are called *descendents*. Similarly, for a given child, the set of coefficients at all coarser scales of similar orientation corresponding to the same location are called *ancestors*. This parent-child dependency between wavelet coefficients in different sub bands is shown in Fig- 2. With the exception of the lowest frequency sub band, all parents have four children.

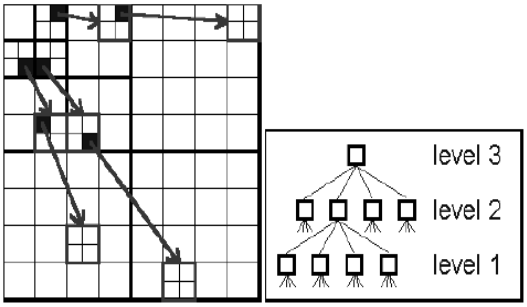

Fig 2: The relation between wavelet coefficients in sub bands as quad tree

A scanning of the coefficient is performed in such a way that no child node is scanned before its parent. For an N-scale transform, the scan begins at the lowest frequency sub band, denoted as LL_N, and scans sub bands HL_N, LH_N, and HH_N, at which point it moves on to scale N-1 etc. The two such scanning patterns for a three-scale pyramid can be seen in Fig. 3. Note that each coefficient within a given sub band is scanned before any coefficient in the next sub band.

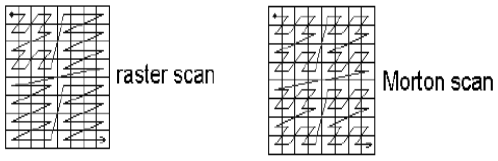

Fig 3: Different scanning patterns for scanning wavelet coefficients

Given a threshold level T to determine whether a coefficient is significant, a coefficient x is said to be an element of a *zerotree* for threshold T if itself and *all* of its descendents are insignificant with respect to T. An element of a zerotree for threshold T is a *zerotree root* if it is not the descendents of a previously found zerotree root for threshold T, i.e., it is not *predictably insignificant* from the discovery of a zerotree root at a coarser scale at the same threshold. A zerotree root is encoded with a special symbol indicating that the insignificance of the coefficient at finer scales is completely predictable. The significance map can be efficiently represented as a string of symbols from a 3-symbol alphabet which is then entropy encoded. The three symbols
are 1) zerotree root, 2) isolated zero, which means that the coefficient is insignificant but has some significant descendent and 3) significant.

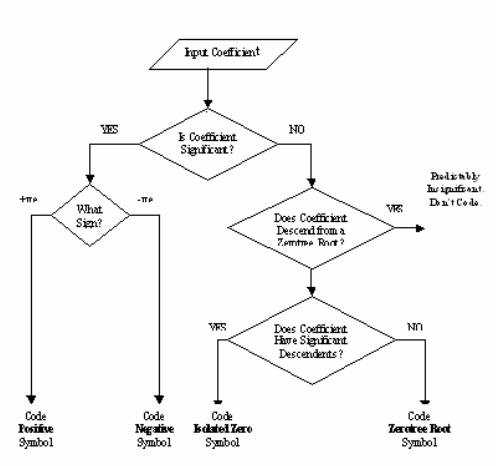

Fig 4: Flowchart for encoding a coefficient of the significance map

In addition to encoding the significance map, it is useful to encode the sign of significant coefficients along with the significance map. Thus, in practice, four symbols are used: (1) zerotree root - 't', (2) isolated zero – 'z', (3) positive significant – 'p', and (4) negative significant – 'n'. This minor addition will be useful for embedding. The flowchart for the decisions made at each coefficient is shown in Fig 4.

4. EZW Algorithm

Uses successive approximation quantization together with zerotree coding to provide embedded bit stream for image. Morton scan order is used to scan the coefficients.

Initialize:
t_o, k=0, dominant list = all coefficients, subordinate list = empty.

Maintain two separate lists:
- Dominant list: contains those coefficients not yet found to be significant
- subordinate list; magnitudes of those coefficients found to be significant

For each threshold, perform two passes: Dominant Pass followed by Subordinate Pass

4.1 Dominant Pass (Significance Pass)

Coefficients *w (m)* on the Dominant List are compared to T_k to determine significance and, if significant, their sign

If $|w(m)| \geq T_k$ [i.e. $w(m)$ is significant]
  If $w(m)$ is positive → Output symbol *'p'*
  Else [i.e., $w(m)$ is negative] → Output symbol *'n'*

Put $w(m)$ on the Subordinate List
Fill their position in Dominant List with zeros.
Calculate reconstruct value (R_v^*) using formula:

$R_v^* = \pm (T_k + T_k/2)$.
R_v^* is +ve if *w (m)* is +ve, and is -ve if *w (m)* is –ve.

Else [i.e., $|w(m)| < T_k$; insignificant]
  If $w(m)$ is a zerotree root → Output symbol *'t'*
  Else $w(m)$ is an isolated zero→Output symbol *'z'*

The resulting stream of symbols is stored in *significance map* and sent to decoder.

4.2 Subordinate Pass (Refinement Pass)

Provide additional precision to the magnitudes on the Subordinate List as follows:

Halve the quantizer width:
$[T_k: 2T_k]$ into → $[T_k: R_v^*]$ and $[R_v^*: 2T_k]$

If magnitude of *w (m)* is in upper half
i.e. in $[R_v^*: 2T_k]$: → encode *'1'*

If *w (m)* is positive: $R_v = R_v^* + T_k/4$
Else (*w (m)* is positive): $R_v = R_v^* - T_k/4$

Else magnitude of *w (m)* is in lower half
i.e. in $[T_k: R_v^* R_v]$: → encode *'0'*
If *w (m)* is positive: $R_v = R_v^* - T_k/4$
Else (*w (m)* is positive): $R_v = R_v^* + T_k/4$

R_v is the value corresponding to *w (m)* which is reconstructed by the decoder.

Send *refinement data* i.e. stream of *'1'*s and *'0'*s to the decoder.

Update: $T_{k+1} = T_k/2$; k = k+1.

Stop when the threshold reaches final threshold value.

The encoded data i.e. *significant map*, *refinement data* are then encoded using Huffman Encoder. This is where the compression takes place. The compressed image along with texture list, symbol count (no. of symbols generated during each Dominant pass), refinement code (no. of refinement codes generated during each Subordinate pass), initial threshold, image size and image mean are then stored, transmitted or passed to the decoder.

5. Analysis of Texture

The WITCH-scheme [1] operates entirely in the wavelet domain. As shown in Fig. 5, the luminance channel of the input image is first decomposed by standard wavelet decomposition, for example, using the Daubechies 9/7 filters, or the 5/3 integer filters, which permits lossless coding. The texture modeling is only applied to the first two levels of decomposition (highest frequencies and finest details, gray shaded in Fig. 5. The sub bands of the other levels are relatively small and can thus be encoded efficiently by the conventional codec.

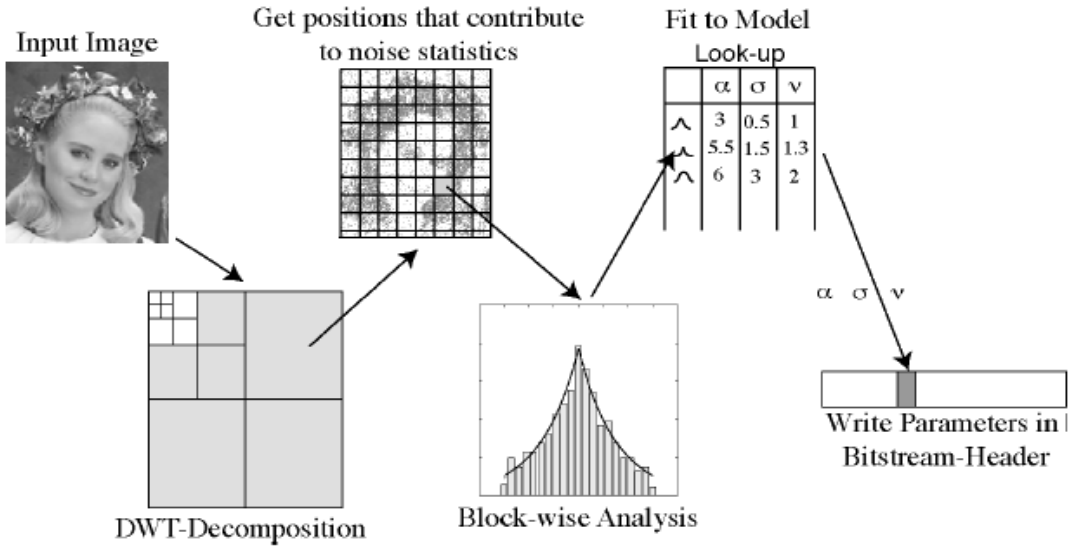

Fig 5: Texture Encoding

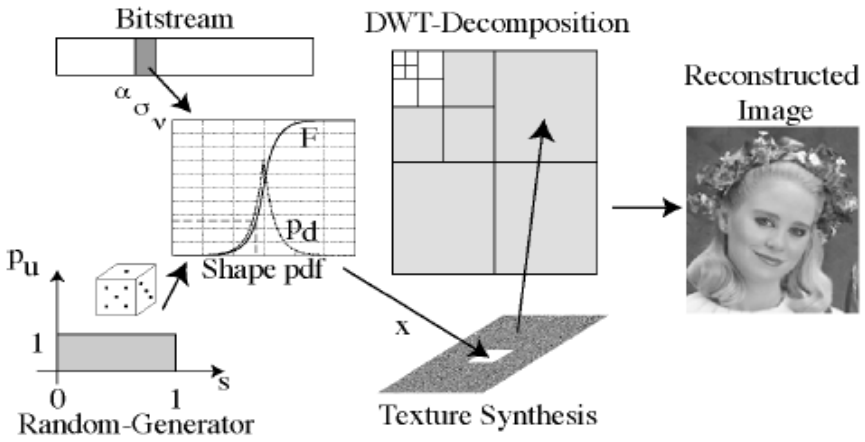

Fig 6: Texture Decoding

Then groups the coefficients of wavelet decomposition sub band into several sub blocks of coefficients and determine the corresponding noise parameters separately for each sub block. This method has the advantage of low computational complexity and less overhead information. In each sub block all wavelet coefficients with small amplitudes (i.e., coefficients represented only by the last two or three biplanes), are assumed to represent stochastic textures. The texture model is fitted separately to each sub block. The resulting texture-model parameters are stored as additional side-information using a very small portion of the bit stream.

At the decoder, the model-parameters for each sub block are read from the bit stream, as shown in Fig.6. They are used to initiate a random noise process with a Probability Density Function (PDF) that is described by the model-parameters. This synthesizes the wavelet coefficients that are inserted at every position where, at the decoding bit rate, no information about the original wavelet amplitude is available. Thus, the texture information lost from the last two or three biplanes is replaced by the synthesized texture.

The stochastic texture is assumed to be given by the set $\Phi^{(k,l,a)}$ of all wavelet coefficients $d_{i,j}$ with

$$d_{i,j}^{(k,l,a)} \varepsilon \Phi^{(k,l,a)} \quad \text{for } |d_{i,j}^{(k,l,a)}| \leq T_A^{(K)}$$

Where, i, j describes the position of the coefficient in the a^{th} sub block of the subband at level k with orientation l. Since the goal is to imitate the stochastic texture lost when quantizing the last two or three bit planes to zero, the threshold is chosen so that only the last two or three bit

planes at decomposition levels $k = 1$ and $k = 2$ and are considered.

The texture model used in the WITCH-system describes the distribution of the amplitudes of the wavelet coefficients in set $\Phi^{(k,\ l,\ a)}$. That is, the model basically characterizes the local histogram of the wavelet coefficient amplitudes. Therefore the sub block-size n used to compute the local histogram is an important parameter. The visual quality was best for subblock-sizes of $n = 16$ or $n = 32$. Therefore, to compute the local histograms for each subblock, block-size of $n = 32$ is chosen.

A good PDF approximation for the wavelet coefficient amplitudes, which describe the stochastic texture, at a particular subband produced by various types of wavelet transforms may be achieved by adaptively varying two parameters of the generalized Gaussian distribution (GGD) [4], which is defined as

$$p(x) = \left[\frac{\upsilon\, \sigma}{2\, \Gamma(1/\upsilon)} \right] \exp\left(-\left[\sigma\, |x| \right]^{\upsilon} \right)$$

Where σ models the width of PDF peak (standard deviation and υ is inversely proportional to the decreasing rate of the peak. Sometimes σ is referred to as the *scale* parameter while υ is called the *shape* parameter. The GGD model contains the Gaussian and Laplacian PDF as special cases, using $\upsilon = 2$ and $\upsilon = 1$, respectively. The gamma function is given by

$$\Gamma(x) = \int_0^{\infty} \exp(-t)\, t^{x-1}\ dt$$

As a result, with only two parameters for the GGD, we can accurately capture the marginal distribution of wavelet coefficients, which represents stochastic texture, in a subband that otherwise would require hundreds of parameters by conventional coding techniques. This significantly reduces the storage of the image features, as well as the time required for encoding and decoding of the image. Besides the two noise-parameters σ and υ, the maximum amplitude α of the generated wavelet coefficients of set $\Phi^{(k,\ l,\ a)}$ is also introduced. It is necessary to limit the maximum amplitude to values smaller than $T_A^{(K)}$, because very homogeneous regions without texture are also represented by coefficients with negligible amplitudes in the non quantized wavelet decomposition.

Since all the model-parameters have to be transmitted for each subblock, they need to be represented in a compact, quantized form. In this work the appropriate quantization levels for the parameters σ, υ and α, have been determined empirically, using a two-step method. First, the GGD was fitted, without any constraints on the parameters, to about 10000 different histograms taken from various images and sub bands. In more than 80% of the cases, the parameters were found to be within the following range:

$\alpha \in [\, 0 : 1\,] * T_A^{(K)}$
$\upsilon \in [1 : 3\,]$
$\sigma \in [\, 0.5 : 1.5\,] * \alpha$

In the second step, experiments were performed by quantizing the parameters within this range with the resolution varying between 1 and 8 bits. The visual fidelity of the generated texture improved up to a quantization precision of 3 bits, but further increases in resolution did not result in a visible difference.
Therefore each parameter is quantized with a precision of 3 bits.

This choice implies that only $2^8 = 256$ different distributions can be distinguished. Consequently the fitting process can be implemented by a lookup procedure. The smallest least-mean square error between the GGD and the real distribution indicates the best fitting entry in the look-up table (LUT). This requires only a few bits per pixel to encode. This could be reduced even further by exploiting the high correlation among the model parameters for adjacent sub blocks.

Due to the simplicity of the chosen texture model (histograms) the overhead information is kept small and the additional computational complexity is negligible. To capture high-order statistics more sophisticated models are necessary. However, only an improvement of the visual quality would justify the additional complexity and bit rate for encoding their parameters. While such an improvement can surely be observed for more structured texture, preliminary experiments did not show such an improvement for the synthesized texture represented by the small wavelet coefficients in the last two or three bit planes. Nevertheless, the issue requires further studies.

6. Huffman Encoding

Huffman coding is an entropy-encoding algorithm used for lossless data compression [6]. The term refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol. Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix-free code (that is, the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol) that expresses the most common characters using shorter strings of bits than are used for less common source symbols. Huffman coding is such a widespread method for creating prefix-free codes that the term "Huffman code" is widely used as a synonym for "prefix-free code."

For a set of symbols with a uniform probability distribution and a number of members which is a power of two, Huffman coding is equivalent to simple binary block encoding.

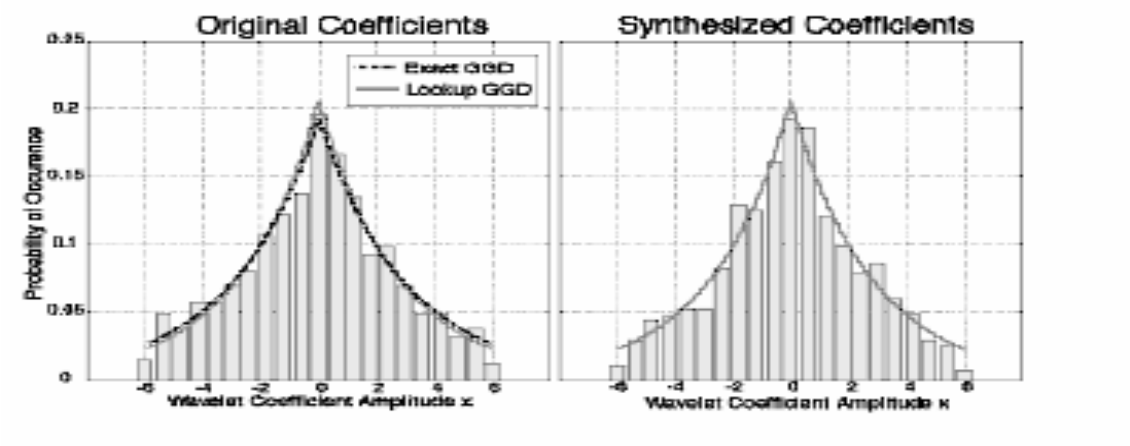

Fig 7: Fit of the GGD to the distribution of wavelet coefficient amplitudes in the analyzed subblock.

Assertions of the optimality of Huffman coding should be phrased carefully, because its optimality can sometimes accidentally be over-stated. For example, arithmetic coding ordinarily has better compression capability, because it does not require the use of an integer number of bits for encoding each source symbol. The efficiency of Huffman coding also depends heavily on having a estimate of the true probability of the value of each input symbol.

7. Proposed Work

This paper combines WITCH (Wavelet Based Image/Texture Codec Hybrid) scheme proposed by Nadenau, Reichel and Kunt (see ref. [1]), which combines the conventional wavelet based coding with model-based texture coding with Huffman encoding to achieve image compression. EZW Encoding combined with texture modeling based on Generalized Gaussian Distribution is used here. I have applied different possible combinations of these techniques briefed earlier in this paper to find out which technique gives the best results.

This technique can be applied to compress all types of images in grayscale domain. Consider the standard grayscale test image *lena.pgm* having 256 X 256 pixels and 256 grayscale levels to compare the different case studies. The different cases are compared on the bases of objective and subjective fidelity criteria and compression ratio obtained in each case.

Consider case I. The image is encoded using EZW algorithm upto the last bit plane including the texture details and then compressed using Huffman encoder. The compression ratio obtained is just above 1 (1.033). This is because while encoding the coefficients by EZW encoder in the last three bit planes, too many symbols are generated which occupy larger space. The decoded image is very similar to the original image but it is not same because of quantization error. The time required for encoding & decoding is very high owing to large number of symbols especially in the last three bit planes.

In case II, the image is compressed directly using Huffman encoder. The decoded image is exact reproduction of the original image hence this technique is lossless. But, the compression ratio is just 1.124. Hence no significant gain in compression ratio is achieved. Another important factor in this case will be the space required to store the Huffman dictionary. For 256 grayscale levels we need to store 256 symbols along with their probabilities.

Consider cases III & IV. The image is encoded using EZW algorithm upto threshold 8 (in 256 grayscale levels, this means upto the last three bit planes) and further texture data is encoded using random process. The EZW encoded data is then applied to Huffman encoder. It can be seen that the compression ratio is around 3-4 which is even better than what is proposed in the WITCH scheme [1]. This is because only four different symbols (p, n, z, and t) are generated by EZW algorithm. After adding the texture details in the image (case IV), Signal to noise ratio improves. However, very minor visual degradation can be observed (in comparison to the quality of the encoded image without using texture modeling as in case III) due to the limitation to a block-wise characterization of the texture properties. This degradation can occur when half of the texture subblock covers a highly structured region and the other half a homogeneous region.

In the WITCH scheme [1], the authors suggested that the texture data in an image is represented by the last three bit planes in wavelet domain and it can be encoded by using random noise process and remaining structural data is encoded by EZW algorithm. I have extrapolated this idea further and tried a different threshold value to check weather this technique works well for a different threshold or not. Consider case V and VI. The image is encoded using EZW algorithm upto threshold 32 (in 256 grayscale levels, this means upto the last five bit planes) and further texture data is encoded using random process. The EZW encoded data is then applied to Huffman encoder. The results achieved are spectacular. It can be seen that the compression ratio is as high as 12, but some visual degradations can be observed in the image.

Sr. No.	Details	Original Image Data	Case I	Case II	Case III	Case IV	Case V	Case VI
	Final Threshold (EZW)	NA	0.0025	NA	0.03125	0.03125	0.125	0.125
1	Image	sol.jpg	sol.jpg	sol.jpg	sol.jpg	sol.jpg	sol.jpg	sol.jpg
	Image Size	128 X 128	128 X 128	128 X 128	128 X 128	128 X 128	128 X 128	128 X 128
2	Original File size (bytes)	16384	16384	16384	16384	16384	16384	16384
	(kB)	16.38	16.38	16.38	16.38	16.38	16.38	16.38
3	Compressed File Size (bytes)	---	17014	15536	7049	7229	2719	2899
	(kB)	---	17.01	15.54	7.05	7.23	2.72	2.89
4	Compression Ratio	1	0.963	1.055	2.324	2.267	6.027	5.652
5	Bits Per Pixel (Bpp)	8	8.308	7.586	3.442	3.53	1.327	1.415
6	RMS Error	---	0	0	0.003906	0.003906	0.015624	0.015624
7	Signal to Noise Ratio (dB)	---	105.15	Infinity	66.0879	65.6338	45.296	44.9847
8	Encoding time (sec)	---	63.38	26.52	64.58	64.58	17.36	17.36
9	Decoding Time (sec)	---	233.89	258.23	200.06	200.06	46.11	46.11
10	Total Time (sec)	---	297.27	284.75	264.64	264.64	63.47	63.47

Sr. No.	Details	Original Image Data	Case I	Case II	Case III	Case IV	Case V	Case VI
	Final Threshold (EZW)	NA	0.0025	NA	0.03125	0.03125	0.125	0.125
1	Image	lena.pgm	lena.pgm	lena.pgm	lena.pgm	lena.pgm	lena.pgm	lena.pgm
	Image Size	256 X 256	256 X 256	256 X 256	256 X 256	256 X 256	256 X 256	256 X 256
2	Original File size (bytes)	65536	65536	65536	65536	65536	65536	65536
	(kB)	65.54	65.54	65.54	65.54	65.54	65.54	65.54
3	Compressed File Size (bytes)	---	63462	58324	18138	18858	5193	5913
	(kB)	---	63.46	58.32	18.14	18.86	5.19	5.91
4	Compression Ratio	1	1.033	1.124	3.613	3.475	12.62	11.08
5	Bits Per Pixel (Bpp)	8	7.747	7.12	2.214	2.302	0.6339	0.7218
6	RMS Error	---	0.000846	0	0.003383	0.003383	0.013532	0.013532
7	Signal to Noise Ratio (dB)	---	106.44	Infinity	66.203	65.7959	49.2948	48.7652
8	Encoding time (sec)	---	972.64	107.33	686.19	686.19	144.94	144.94
9	Decoding Time (sec)	---	3345.97	902.09	2333.28	2333.28	391.11	391.11
10	Total Time (sec)	---	4318.61	1009.42	3019.47	3019.47	536.05	536.05

Case I Image encoded using EZW algorithm upto the last bit plane & compressed using Huffman Encoder

Case II Image Compressed Directly using Huffman Encoder

Case III Image encoded using EZW algorithm without texture data upto threshold 8 & compressed using Huffman Encoder

Case IV Image encoded using EZW algorithm with texture data upto threshold 8 & compressed using Huffman Encoder

Case V Image encoded using EZW algorithm without texture data upto threshold 32 & compressed using Huffman Encoder

Case VI Image encoded using EZW algorithm with texture data upto threshold 32 & compressed using Huffman Encoder

8. Results

In the experiment various images are used in the program and results are obtained for each of the six cases discussed in the last section. The results of two such different test images *lena.pgm*, *sol.jpg* are shown below.

The different statistical values are summarized in the table. Thus, it can be concluded that EZW encoding supported by texture modeling combined with Huffman encoder gives excellent results. By choosing suitable threshold value compression ratio as high as 12 can be achieved.

Results: lena.pgm (256 X 256)

Fig 1: Original Grayscale Image (64 KB)

Fig 2: Reconstructed Image - Case I (63.46 KB)

Fig 3: Reconstructed Image - Case II (58.32 KB)

Fig 4: Reconstructed Image - Case III (18.14 KB)

Fig 5: Reconstructed Image - Case IV (18.86 KB)

Fig 6: Reconstructed Image - Case V (5.19 KB)

Fig 7: Reconstructed Image - Case VI (5.91 KB)

Results: sol. jpg (128 X 128)

Fig 1: Original Grayscale Image (16 KB)

Fig 2: Reconstructed Image - Case I (17.01 KB)

Fig 3: Reconstructed Image - Case II (15.54 KB)

Fig 4: Reconstructed Image - Case III (7.05 KB)

Fig 5: Reconstructed Image - Case IV (7.23 KB)

Fig 6: Reconstructed Image - Case V (2.72 KB)

Fig 7: Reconstructed Image - Case VI (2.89 KB)

9. Conclusion

A technique for image compression which uses the Wavelet-based Image/Texture Coding Hybrid (WITCH) scheme [1] in combination with Huffman encoder is proposed here. It combines EZW encoding with stochastic texture modeling using Generalized Gaussian Distribution to encode the image which is further compressed by using Huffman encoder. It is clear from above that this technique yields spectacular results with compression ratio as good as 3 to 4 times the original image with high visual quality of the reconstructed image. The Bits per pixel required is as low as 2 to 2.5. This approach utilizes zerotree structure of wavelet coefficients very effectively, which results in higher compression ratio and better signal to noise ratio. Experimental results have proved that the EZW encoding supported by texture modeling combined with Huffman encoder provides significant performance improvement.

A wavelet codec (EZW) encodes contours and structured pictorial elements efficiently, but it performs very badly on fine stochastic textures. In contrast, it is difficult to design a universal model for all kinds of textures. However, once the image is represented in the wavelet domain, the properties of stochastic texture can be captured relatively easily, and therefore modeled efficiently. The proposed scheme combines these two advantages in a very flexible manner into a hybrid system. Furthermore, the system has extremely low computational complexity. Further improvements in terms of the texture model are possible without affecting the general structure of the proposed system. Very minor visual degradation could be observed (in comparison to the quality of the encoded image without using texture modeling) due to the limitation to a block-wise characterization of the texture properties. This degradation can occur when half of the texture sub block covers a highly structured region and the other half a homogeneous region. In this case the same texture properties are synthesized for both halves. While the local activity masks the added texture in one-half, the homogeneous half might get noisier. However, only very small amplitude texture of the last two or three bit planes are added, and the described constellation is of low probability.

References

[1] Marcus J. Nadenau, Julien Reichel and Murat Kunt, “Visually Improved Image Compression by Combining a Conventional Wavelet-Codec with Texture Modeling,” IEEE Transactions on Image Processing Vol. 11, No. 11, Nov 2002, pp. 1284-1294.

[2] Jerome M. Shapiro,” Embedded Image Coding Using Zerotrees of Wavelet Coefficients,” IEEE Transactions on Signal Processing December 1993.

[3] Minh N. Do and Martin Vetterli, “Wavelet-Based Texture Retrieval using Generalized Gaussian Density and Kullback-Leibler Distance,” IEEE Transactions on Image Processing Vol.11, No. 2, 2002, pp.146-158.

[4] Creusere, C. D.,”A New Method Of Robust Image Compression Based On The Embedded Zerotree Wavelet Algorithm,” IEEE Transactions on Image Processing, Vol. 6, No. 10, 1997, pp. 1436-1442.

[5] C. Valens, EZW Encoding

[6] website: http://en.wikipedia.org/wiki/Huffman_coding

Vinay U. Kale obtained his BE (Electronics) degree in 1990 and ME (Electronics) in 1996 from Amravati University, Amravati. He is currently a Researcher and Professor in Department of Electronics and Telecommunication Engineering, Prof. Ram Meghe Institute of Technology and Research, Badnera. Amravati, Maharashtra, India. His main research interest is in the field of Image Processing, Power Electronics, etc. He has published fifteen research papers in National and International Journals & Conferences. He is a Life Member of IETE, IE, & ISTE.

Shirish M Deshmukh received his ME (Adv. Electronics) degree from Amravati University, Amravati. He is currently a Researcher and Professor in Department of Electronics and Telecommunication Engineering, Prof. Ram Meghe Institute of Technology and Research, Badnera. Amravati, Maharashtra, India. His main research interest is in the field of Control System, Communication Engg., etc. He is a Life Member of IETE, IE, & ISTE.