Performance of Lossy Image Compression Coders on Textures

Osslan Osiris Vergara Villegas[1], Raúl Pinto Elías[1] and José Ruiz Ascencio[1]
[1] Centro Nacional de Investigación y Desarrollo Tecnológico (cenidet)
Computer Science Department
Interior internado Palmira s/n Col. Palmira Cuernavaca Morelos México
{osslan, rpinto, josera}@cenidet.edu.mx
(Paper received on August 11, 2006, accepted on September 26, 2006)

Abstract. Image compression algorithms should be evaluated by at least two important criteria: the compression rate and the quality of the reconstructed images. Other factors which may be considered are compression/decompression times, the memory requirements, etc. This paper shows the results obtained in a performance comparison between six lossy compression algorithms on texture images. We measure the quality of the reconstructed image and the compression/decompression process time at moderate and low bit rate compression ratios. The results shown allow the selection of an algorithm as a function of image texture, in future we intend to measure other image components such as edges and edge-associated detail to have the possibility of recommending a compression algorithm that adapts to any image that we need to compress.

1. Introduction

Digital images are much used in several disciplines of computer science besides Digital Image Processing (DIP). A great quantity of the information that human beings use daily is in a digital form, digital images need a great amount of storage space and more bandwidth and time to transmit it for example via Internet. A possible solution to handle this data and to use less storage space and time to transmit is to use image compression.

The goal of image compression is to reduce the bit rate for signal transmission or storage while maintaining an acceptable image quality for various purposes [1]. The compression process can be made with or without loss of information. In lossless methods the decompressed data are an exact copy of the originals, whereas in the lossy methods, the decompressed data are an approximation of the originals.

Image compression methods have been evaluated on the basis of minimizing an objective distortion measure at a given level of data compression [2]. At low bit rates an image can be distorted or present artifacts in components such as edges or textures and it becomes imperative to measure the performance of a particular coder in order to know if there exists a balance between the quality of the reconstructed image and the compression factor obtained, which is a suitable criterion in lossy compression coders.

In the literature there are a few works [3], [4], [5] that attempt to measure the performance of a compression algorithms in order to try to recommend an adequate algo-

Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 249-258

rithm for the image that needs to be compressed. The difference between this paper and others reported in the literature is that we present the first and novel results obtained in a comparison made using the very important information of a texture cue in an image. We made the comparison by measuring the time used to compress/ decompress an image and the quality of the reconstructed image at moderate and low bit rate. This study was made as a base work in order to design in the future a compression coder with feature preserving (textures, edges, and edge associated detail).

In the following sections we present a brief description of the proposed model and the tests and results obtained from the compression algorithms comparison.

2. Proposed Model

All the objects in the real world have a surface that can reflect light in a way that depends on the structure of the surface. That manner of reflecting light is known as the visual texture of the objects and gives information about the material that the object is composed of (wood, water, steel, wool, etc) and some properties (roughness, regularity, brightness, homogeneity, etc) that inform about the state (wet, clean, liquid, etc) of the object.

An image gives us the sensation of texture by the repetition of similar patterns, in [6] Julez gives a definition of texture: Textures are composed by a small number of similar types of atoms called textons, that are repeated in almost regular or random position and orientation.

Several authors agree that the three most important dimensions for texture perception by human beings are: periodicity, randomness and directionality [7], [8]. A texture image can be classified in one of the three mentioned types and one can use the Fourier spectrum to help the task of classification [7].

A rough description of the Fourier spectrum for each texture class is:

a) *Periodic textures:* The spectrum consists of significant peaks scattering out regularly in some directions.
b) *Directional textures:* In the Fourier spectrum the directionality will be preserved.
c) *Random textures:* The distribution of the responses of the spectrum is not restricted to certain directions.

In this paper we measure the performance of lossy image compression algorithms on each of the three classes of textures with the goal of investigating which compressor has the best performance for each class in order to offer the possibility of recommending a compression algorithm for the particular texture information of an image.

The proposed model has six phases, first we select the database of image textures, second we compute the Fourier spectrum, third we classify the image in one of the three texture classes, fourth the image is compressed/decompressed with six lossy image coders, fifth the error between original and reconstructed image and the time used to compress/decompress are measured; and finally we make comparisons between the images. Figure 1 shows the phases for the proposed model.

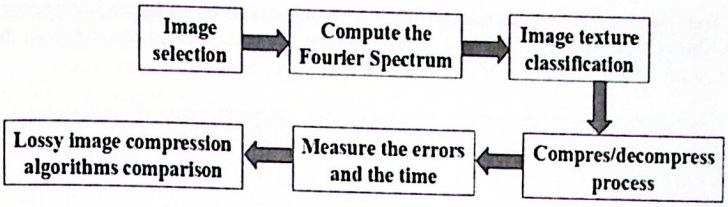

Fig. 1. The six modules of the proposed model.

2.1. Image Selection, Fourier Spectrum Computing and Texture Classification

We obtain the texture image database from the Brodatz album [9]. The album has become a standard for evaluating texture algorithms and consists of 112 texture images belonging to different classes. We select five images for each texture class for a total of 15 texture images that were used for the tests in this paper. The selection of the textures for each class was made by computing the Fast Fourier Transform (FFT) equation 1 and the Fourier Spectrum (FS), equation 2.

$$F(p,q) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m.n) e^{-J(2\pi/M)} e^{-J(2\pi/N)qm} \tag{1}$$

$$|F(u,v)| = \left[R^2(u,v) + I^2(u,v)\right]^{1/2} \tag{2}$$

By means of visual analysis of the FS we are ensured that the images selected belong to the texture class attributed to them. The selected images are greyscale images (256 x 256) which need a storage space of 66Kb (65536 bits). After the analysis five directional textures were selected showed in the first row of figure 2, the second row shows their correspondent Fourier Spectra.

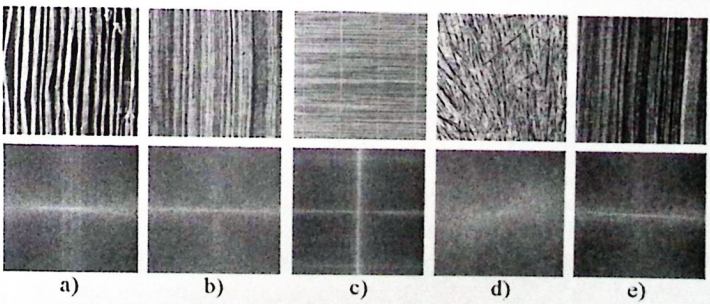

Fig. 2. Directional textures. a) D51 raffia woven with cotton threads, b) D106 cheesecloth, c) D49 Straw screening, d) D15 straw and e) D105 cheesecloth.

The five periodic textures selected are shown in the first row of figure 3, the second row shows their Fourier Spectra. The five random textures selected are shown in the first row of figure 4, the second row shows their Fourier Spectra.

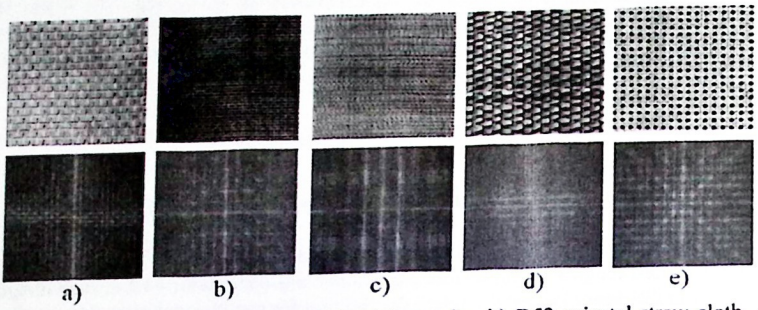

Fig. 3. Periodic textures. a) D1 woven aluminium wire, b) D52 oriental straw cloth, c) D55 straw matting, d) D56 straw matting and e) D101 cane.

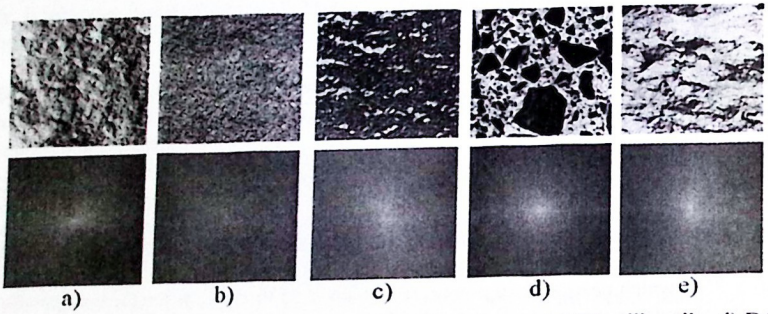

Fig. 4. Random textures. a) D2 fieldstone, b) D9 grass lawn, c) D86 ceiling tile, d) D62 european marble and e) D7 fieldstone D2.

2.2. Compress/Decompress Process

Once we have the textures classified adequately, the compress/decompress phase of the images is carried out. We select six lossy image compression algorithms: Discrete Cosine Transform (DCT) [10], Haar Wavelet Transform (HWT) [11], Singular Value Decomposition Transform (SVDT) [10], Daubechies 4 Wavelet Tranform (D4WT) [11], Embedded Zerotree Wavelet (EZW) [12] and Set Partitioning In Hierarchical Trees (SPIHT) [13]. These algorithms were programmed as proposed in the cited references and were set to obtain similar bit rates so that all coders have the same conditions and we avoid skewing the results.

Compression standards such as JPEG and JPEG2000 were not used because this work intends to serve as the basis of a future image coder with feature preservation

with respect to a target pattern recognition algorithm. A disadvantage of standards is that the only reference is human vision and they attempt to equally preserve the quality of any image. Thus they can not be easily manipulated for tests cases, and at very low bit rates the images obtained are corrupted by artifacts, and image features such as edges may be practically destroyed.

The first compress/decompress process was made with a bit rate of 0.5 with the resulting images using 33Kb (32768 bits). After the process we have a total of 90 texture images (30 in each class). The second process was made using a bit rate of 0.1, with resulting images using 6Kb (6554 bits). Its purpose was to find the algorithm with the best performance in our worst case scenario. Here we again obtain 90 texture images.

2.3. Measuring the Errors and Time

After the lossy compress/decompress process, it is important to measure the difference between the original image (I) and the decompressed image (I') in order to determine the quality of the reconstructed image [14]. For this purpose we use four well known objective measures: Mean Square Error (*MSE* eq. 3), Peak Signal to Noise Ratio (*PSNR* eq. 4) and Norm 2 and Frobenius relative errors (eq 5).

$$MSE = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}\left[I(x,y)-I'(x,y)\right]^2 \tag{3}$$

$$PSNR = 10 * \log_{10}(255 / sqrt(MSE)) \tag{4}$$

$$N_2, F = \frac{\left\| I - I' \right\|_{2,F}}{\left\| I \right\|_{2,F}} \tag{5}$$

For the measurement of time we implement a function in every algorithm to add up the time in seconds needed to complete the process of compressing/decompressing an image of a texture. To avoid skewing the results, all codecs were programmed by the same programmer, in C++ Builder, and executed in a Windows XP environment running on Intel Pentium IV processor at 2.3 GHz. with 256 MB of RAM memory.

2.4. Lossy Image Compression Algorithms Comparisons

In this stage it is important to analyze which texture of each class represents more problems for the compression algorithms; that is, which texture is better reconstructed, and which texture has poor reconstruction for each algorithm.

On the other hand, it is important to analyze the performance of the coders regarding execution time, it is desirable that the process have a balance between the quality of the reconstructed image and the time used to compress/decompress an image.

With these two measures we can compare the performance of the algorithms for each texture class and then give a recommendation of the best algorithm.

3. Tests and Results

We have two test groups, reconstructed images obtained from: a) 0.5 bit rate and b) 0.1 bit rate as pointed out in section 2.2. We have 15 texture images on which we apply six compression algorithms to obtain a total of 90 images for case *a* and 90 images for case *b*, for a total of 180 texture images.

For each image we compute the four error measures proposed in section 2.3 and the time used for the compress/decompress process of an image. As a result, we have five measures for each of the 180 images. Because of the limited space of this paper we only show the mean and the standard deviation results obtained for each texture class for each bit rate. Figure 5 shows an example of different coders reconstructed images: the first row show 0.5 bit rate images and second row 0.1 bit rate images.

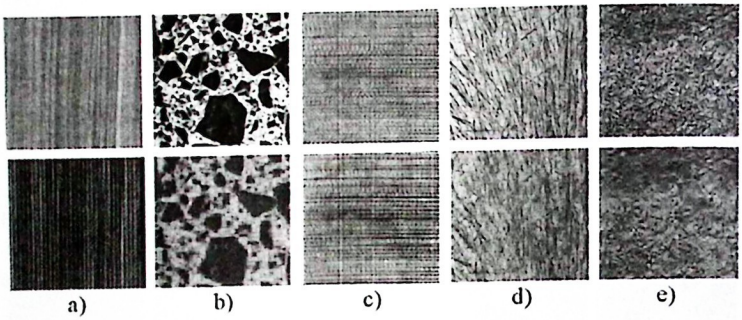

Fig. 5. Reconstructed texture images. a) HWT D105, b) EZW D62, c) DCT D55, d) D4WT D15, e) SPIHT D9.

Table 1 shows the results for the mean (upper half) and for the standard deviation (lower half) for directional textures, table 2 the results for periodic textures and table 3 the results for random textures all with bit rate of 0.5.

Table 1. Directional textures average and standard deviation results for bit rate 0.5.

Average					
Compression algorithm	*MSE*	*PSNR*	*Frobenius*	*Norm2*	*Time in seconds*
DCT	191.6071	26.6042	0.0919	0.0568	1.51
HWT	1616.7638	16.4300	0.2976	0.2096	2.102
SVDT	147.5686	28.0624	0.0789	0.0364	6.0028
D4WT	427.6316	23.0812	0.1395	0.0713	3.31
EZW	83.0009	29.5485	0.0642	0.0321	11.8
SPIHT	630.0550	21.2940	0.1693	0.0474	7.92
Standard deviation					
DCT	154.6206	3.8576	0.0279	0.0402	0.0761
HWT	595.6014	2.3081	0.0896	0.1115	0.0634
SVDT	172.6100	3.6981	0.0319	0.0127	0.2633
D4WT	303.4030	4.0181	0.0444	0.0470	0.0894
EZW	57.7073	2.3492	0.0138	0.0061	0.2121
SPIHT	505.4763	3.5089	0.0509	0.0357	0.1303

Table 2. Periodic textures average and standard deviation results for bit rate 0.5.

Compression algorithm	*MSE*	*PSNR*	*Frobenius*	*Norm2*	*Time in seconds*
			Average		
DCT	248.5560	25.6067	0.1230	0.0837	1.46
HWT	2341.2887	15.3872	0.3884	0.3312	2.186
SVDT	101.3690	28.5201	0.0822	0.0647	5.978
D4WT	478.2486	22.0153	0.1682	0.1150	3.252
EZW	87.9482	29.020	0.1282	0.0940	11.722
SPIHT	659.4961	20.5227	0.1950	0.0745	7.98
			Standard deviation		
DCT	178.8439	4.4860	0.0843	0.0798	0.1534
HWT	1674.9572	3.2333	0.2936	0.3317	0.2745
SVDT	46.3382	2.3249	0.0505	0.0453	0.2278
D4WT	315.3719	2.6695	0.0850	0.0822	0.3582
EZW	2433.3408	4.4780	0.1129	0.1154	0.4748
SPIHT	399.1214	2.4817	0.0807	0.0388	0.7049

Table 3. Random textures average and standard deviation results for bit rate 0.5.

Compression algorithm	*MSE*	*PSNR*	*Frobenius*	*Norm2*	*Time in seconds*
			Average		
DCT	221.7494	25.3617	0.1059	0.0466	1.54
HWT	1490.8119	16.6299	0.2816	0.1609	2.326
SVDT	295.1127	23.8196	0.1241	0.0378	6.24
D4WT	473.9257	21.5967	0.1579	0.0655	3.474
EZW	152.4267	27.0863	0.0879	0.0319	12.004
SPIHT	558.3568	21.0256	0.1697	0.0259	8.32
			Standard deviation		
DCT	138.6217	2.7676	0.0461	0.0228	0.0674
HWT	543.3977	1.5999	0.1047	0.1304	0.2911
SVDT	139.6775	2.0516	0.0475	0.0276	0.2302
D4WT	160.3059	1.6038	0.0481	0.0304	0.4407
EZW	81.4840	3.2836	0.0397	0.0154	0.5020
SPIHT	270.5800	1.9376	0.0586	0.0080	0.4919

For the results shown in tables 1, 2 and 3 we can give the following comments: For the three texture classes, and taking into account the quality of the reconstructed image (error measures), the EZW algorithm has the best performance. This can be seen, for example, in that MSE always has the smallest value and the PSNR is always the largest, which indicates that the images are quite close to the originals.

The next best compressor is DCT, which is demonstrated by the Frobenius and Norm2 measures, which are just above those for EZW. After DCT, we obtained the best performance with SPIHT and SVDT algorithms, in that order. The poorest performance is presented in HWT: for example in the MSE we observe that the image is reconstructed with a large difference for the three texture classes. The second worst algorithm is D4WT, it is only superior for reconstruction to the HWT algorithm.

For the time variable, different things can be said. For example the EZW algorithm which has the best performance in image reconstruction has the disadvantage of using at least twice the time to compress/decompress compared with the other algorithms. The second slowest algorithm is SVDT followed by the SPIHT algorithm.

The fastest algorithms are DCT, HWT, D4WT in that order. DCT is the fastest algorithm and if we observe that it gives a good reconstruction of the image, we can understand why DCT is a very important part of the JPEG standard.

Regarding individual textures, those that pose more problems for the compression algorithms are the directional texture D15, the periodic textures D1 and D101 and the random textures D2, D9 and D86. For the evaluation by class we can say that for the case of directional textures the best algorithm is EZW followed by SVDT. The poorest performance is presented by the HWT algorithm. For periodic textures the best algorithm is EZW followed by DCT. The poorest performance is presented by the SVDT. Finally for the case of random textures the best algorithm is the DCT followed by the SVDT. The poorest performance is presented by the HWT.

The results obtained for the case of 0.1 bit rate are shown in table 4 for directional textures, table 5 for periodic textures and table 6 for random textures.

Table 4. Directional textures average and standard deviation results for bit rate 0.1.

			Average		
Compression algorithm	*MSE*	*PSNR*	*Frobenius*	*Norm2*	*Time in seconds*
DCT	2967.4880	14.8655	0.3799	0.2496	1.172
HWT	6690	10.1215	0.5935	0.5041	1.88
SVDT	2819.9901	16.3922	0.3537	0.2215	9.8
D4WT	1760	16.2352	0.2990	0.1733	3.116
EZW	2590	14.5630	0.3813	0.2500	6.616
SPIHT	1430	17.4993	0.2596	0.1031	5.232
			Standard deviation		
DCT	2100.6320	4.4649	0.1793	0.1705	0.1285
HWT	2533.5941	1.6230	0.0577	0.0802	0.1923
SVDT	2328.7143	6.5911	0.2205	0.1986	0.2318
D4WT	899.0717	2.5749	0.0655	0.0867	0.1152
EZW	1405.9375	2.6184	0.1755	0.2067	0.5896
SPIHT	1018.8797	3.2087	0.0659	0.0802	0.2783

Table 5. Periodic textures average and standard deviation results for bit rate 0.1.

			Average		
Compression algorithm	*MSE*	*PSNR*	*Frobenius*	*Norm2*	*Time in seconds*
DCT	4950	11.3217	0.4815	0.3753	1.1808
HWT	8490	8.7890	0.6235	0.5396	1.842
SVDT	5673.289	12.68	0.4550	0.3684	9.884
D4WT	2003.4994	13.3643	0.3576	0.2779	3.072
EZW	3300	12.9024	0.4068	0.2996	7.006
SPIHT	1590	14.2571	0.2934	0.1462	5.278
			Standard deviation		
DCT	4555.6807	6.4979	0.1606	0.1382	0.2266
HWT	5536.5457	5.3933	0.0540	0.0805	0.2651
SVDT	6031.6913	8.5941	0.2298	0.1899	0.4086
D4WT	1263.4135	7.5819	0.2333	0.2585	0.3434
EZW	3556.1362	7.0534	0.2103	0.2248	0.1717
SPIHT	1103.1317	7.8059	0.0923	0.0614	0.3720

Table 6. Random textures average and standard deviation results for bit rate 0.1.

Compression algorithm	*MSE*	*PSNR*	*Frobenius*	*Norm2*	*Time in seconds*
			Average		
DCT	4345.1877	13.3438	0.4457	0.2277	1.4956
HWT	8740	9.0543	0.6464	0.5368	2.1772
SVDT	4440	12.8528	0.4538	0.2351	10.304
D4WT	1371.1854	16.8818	0.2679	0.0761	3.344
EZW	3130	14.2036	0.368	0.1826	7.1648
SPIHT	1270	17.141	0.2599	0.0509	5.613
			Standard deviation		
DCT	3221.4480	4.6856	0.2273	0.1829	0.1968
HWT	4059.253	1.8705	0.0778	0.0576	0.4515
SVDT	3109.3429	3.9110	0.1973	0.1557	0.5642
D4WT	367.7127	1.1482	0.0675	0.0319	0.2854
EZW	2818.5531	3.0621	0.1062	0.0920	0.5658
SPIHT	236.4950	0.7844	0.0661	0.0145	0.4867

For our worst test case with results in tables 4, 5 and 6 there are several comments: First the reconstructed images are obviously reconstructed with a large error. Second, the relative performance of the algorithms is different compared with the previous test case. Taking into account the quality of the reconstructed image, the poorest performance is presented by the HWT, it has the largest MSE and the reconstructed images are visually different. The next poorest performance is DCT and SVDT respectively. The best performance is given by SPITH followed by D4WT and EZW in that order.

With respect to time, the best algorithm is the DCT follow by transform algorithms HWT and D4WT. The poorest performance is presented by the SVDT algorithm. In this case we observe that the time needed to compute using less singular values is larger instead of smaller than the first test case. Next in speed are the algorithm SPIHT and EZW algorithms, in that order. Regarding individual textures, those that present problems for the compression algorithms are the directional textures D49 and D15, the periodic textures D55 and D56 and the random textures D62 and D7. For the evaluation by texture class, the best algorithm for directional textures is SVDT but it has the problem of time and the poorest is HWT. For periodic textures the best is SPIHT and the poorest is HWT. For random textures the best is DCT and the poorest is HWT.

4. Conclusions

In this paper we present the results obtained from a performance comparison of six lossy image compression algorithms. The algorithms were compared using texture images and measuring by two variables, the quality of the decompressed image and the time used for the full cycle of compression/decompression on an image. A performance ranking of the algorithms was obtained for each of three texture classes, and we found that this ranking varies with the bit rate and the image nature. These results are important because they are the first that take into account texture, one of the basic cues for image recognition.

In future work we intend to make a similar analysis measuring other components of images in order to give a final and more accurate criterion for the selection of an algorithm to compress an image and in order to build a feature preserving image coder. The goal of the future coder will not be to minimize an objective distortion measure at a given bit rate, but to preserve important features of the image. The measure of image quality will be taken with a pattern recognition process.

References

[1] C. Gonzalez Rafael and E. Woods Richard, *Digital Image Processing*, Addison Wesley / Díaz de Santos, U. S. A., 2000.

[2] Saffor A., Bin Ramli R. A., Hoong Ng K. and Dowsett D., Objective and Subjective Evaluation of Compressed Computed Tomography (CT) Images, *The Internet Journal of Radiology*, Vol. 2, No. 2, June 2002.

[3] Yang S., Zhang X. and Mitra S., Performance of Lossy Compression Algorithms from Statistical and Perceptual Metrics, *Proceedings of the 12th IEEE Symposium on Computer Based Medical Systems (CBMS)*, Stamford, CT, USA, pp. 242, June 1999.

[4] Garcia J. A., Fernandez Valdivia J., Rodríguez Sánchez R. and Fernandez Vidal X., Coder Selection for Lossy Compression of Still Images, *Pattern recognition*, Vol. 35, No. 11, pp. 2489 – 2509, November 2002.

[5] Jerabek B., Schneider P. and Uhl A., Comparison of Lossy Image Compression Methods Applied to Photorealistic and Graphical Images Using Public Domain Sources, Technical Report, Research Institute for Software technology, University of Salzburg, April 1998.

[6] Julesz B., Visual pattern discrimination, *IRE transactions on information theory*, vol. 8, pp. 84 - 92, February, 1962.

[7] Lee K. L. and Chen L. H., Unsupervised Texture Segmentation by Determining the Interior of Texture Regions Based on Wavelet Transform, *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 15, No. 8, pp.1231 - 1250, December 2001.

[8] Portilla J. and Simoncelli. E. P., A Parametric Texture Model based on Joint Statistics of Complex Wavelet Coefficients, *International Journal of Computer Vision*, Vol. 40, No. 1, pp. 49 - 71, October 2000.

[9] Brodatz P., *Textures: A photographic album for artists and designers*, New York: Dover, NY, 1996.

[10] Öktem R., Transform Domain Algorithms for Image Compression and Denoising, M. S. Dissertation, Sygnal processing labortory, Tampere University of Technology, Finland, May 2000.

[11] Davis G. and Nosratinia A, Wavelet-Based Image Coding: An Overview, *Applied and Computational Control, Signals, and Circuits*, Vol. 1, No. 1, pp. 205 – 269, 1998.

[12] Shapiro J. M., Embedded Image Coding Using Zerotrees of Wavelet Coefficients, *IEEE Transactions on Signal Processing*, Vol. 41, No. 12, pp. 3445 - 3463, December 1993.

[13] Said A. and Pearlman W., A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees, *IEEE Transactions on Circuits and Systems forVideo Technology*, Vol. 6, No. 3, pp. 243 - 250, June 1996.

[14]. Mrak M., Grgic S. and Grgic M., Picture Quality Measures in Image Compression Systems, *Proceedings of the IEEE International Conference on "Computer as a tool" (EUROCON)*, Vol. 1, Ljubljana, Slovenia, pp. 233 – 236, September 2003.