JPEG with Quadtree Adaptive Partitioning. Loss Analysis

Lic. Claudia C. Russo[i]
Lic. Hugo D. Ramón[ii]
Eng. Armando De Giusti[iii]
Analyst Raúl Flores[iv]

Laboratorio de Investigación y Desarrollo en Informática
Departamento de Informática - Facultad de Ciencias Exactas
Universidad Nacional de La Plata[v]

Abstract

In this paper two images compression techniques with loss based on the method used by the compression Standard Joint Photographic Experts Groups (JPEG) and on the Quadtree adaptive partitioning method are presented and compared, and the loss produced studied.

The baseline JPEG algorithm will be analyzed, and optimizations emphasizing quantification and not modifying the standard restrictions will be proposed.

The results obtained from the generalization in classes of images with very different characteristics will be analyzed; and blocks size effect in conflicting areas such as the borders will be considered.

Key words: Images, Compression, JPEG, Quantification, Quadtree, Adaptive, Loss

Introduction

[i] Co-Chair Professor and Practical Classes Coordinator. Department of Computer Sciences, Faculty of Exact Sciences, UNLP.
[ii] Full-time Practical Classes Coordinator, Department of Computer Sciences, Faculty of Exact Sciences, UNLP.
[iii] Director of the LIDI. Principal Researcher of the CONICET. Full-time Chair Professor, Department of Computer Sciences, Faculty of Exact Sciences, UNLP.
[iv] Advanced student of the course of studies Licenciature on Computer Sciences. Department of Computer Sciences, Faculty of Exact Sciences, UNLP.
[v] Calle 50 y 115 Primer Piso, La Plata (1900), Argentina, Pcia. de Bs. As. Telephone nr. 54-21-227707
e-mail lidi@ada.info.unlp.edu.ar

JPEG is one of the best known Standards for images compression with loss. A good compression ratio is obtained when applying it on photographs, works of art, and similar material, although it is not suitable for texts, simple drawings or lines. JPEG exploits the human visual system limitations. [Cla95] [Nels91].

JPEG defines three different codification systems:

- A baseline codification system with loss, which uses as a base the Discrete Cosine Transform (DCT)
- An extended codification system for applications requiring more accuracy, progressive reconstruction, etc.
- An independent codification system without loss for a reversible compression.

In order to be JPEG-compatible, any product should include a support for the baseline system. The standard proposes a syntax to be satisfied by any bits sequence in order to be JPEG, allowing *a large degree of freedom* for the quantification and codification stages, so that optimizations and improvements can be carried out. No images format is specified, nor spatial resolution or any particular features as regards color. A useful characteristic of the method is that loss degrees may be varied by adjusting compression parameters, which will be explained later.

Baseline JPEG algorithm description

The system recommended by JPEG is a modification of that proposed by Chen and Pratt, and it is based on the codification technique using DCT. The compression process is done in three sequential steps [Say96]:

- **Transformation**: The transform used is the DCT.
- **Quantification**: The uniform MIDTREAD quantification is used to quantify the coefficients of a transformed block.
- **Codification**: The DC and AC coefficients resulting from the quantification are codified using variable length codes.

Generalization to fixed partitioning

The first implementation is a generalization to partition the image in fixed-size blocks (4x4, 8x8, 16x16, 32x32). Different quantification and codification tables are used: for fixed partitioning with 8x8 blocks, the default quantification matrix was used, and the different loss levels were obtained by multiplying this matrix by an scalar or factor; the Huffman codification tables for 8x8 blocks are those provided by the standard. For the other block sizes the following quantification matrix was built: =1+(1+i+j)*factor; $0 <= i, j < N$

Although there are techniques to build adaptive quantification tables, the criterion used is that coefficients generally decrease in importance from the upper left corner to the lower right one. This is the reason why the matrix is built in a way so that the coefficients closest to the (0,0) position keep a greater accuracy when being dequantified.

The Huffman codes tables were built by means of a run over different images, on which different loss degrees were applied. These runs allowed the extraction of symbols probabilities, which were chosen according to the block.

Generalization Results and Observations

The VLSI.BMP image, which is a white-written formula on a black background, and from which a 64x64 pixel sub-image belonging to the black color section was extracted, was used to carry out the tests. The fixed size partitioning algorithm for the 4 types of blocks with factor 1 was applied to this sub-image, and ratios of 18, 36, 55 and 65 were obtained for N = 4, 8, 16 and 32 respectively. These were kept unchanged in spite of making the factors vary, whereas the decompressed images did not present changes in their black shade, at least visually. Then the same algorithm was applied to the original complete image and the following ratios were obtained:

N=4		*N=8*		*N=16*		*N=32*	
Factor	Ratio	Factor	Ratio	Factor	Ratio	Factor	Ratio
2	6	2	13	2	6	5	13
8	8	7	24	6	11	9	21
18	10.4	9	27			10	23
25	10.8						

The best results as regards visual quality were obtained when processing 4x4 blocks, although compression ratios increased slower when applying factors greater than 18. As the table shows, for factors 18 and 25 the difference in ratio is minimal and the quality of the decompressed images is similar. With factors 7, 6 and 10 for 8x8, 16x16 and 32x32 blocks respectively, the distortions in the area of the formula grew notorious. Even though a greater compression was obtained with greater block size, visible distortions affecting black areas away from the formula were produced.

Quadtree Partitioning

This partitioning is used for images processing, and it attempts to overcome the difficulties presented by the fixed partitioning. The basic algorithm consists on taking a square block of the image, which is then divided in four equally-sized, square sub-blocks. This is recursively repeated from the complete original image until the squares reach the desired size, which is determined according to a decision test which depends on the application. The same partitioning, but in a bottom up way, can be carried out, beginning the process with the smaller sub-blocks and joining them to form bigger ones.

Generalization using Quadtree partitioning

Quadtree is used to adapt different areas of the image to a determined block size, according to its gray shades variation. Large areas with slow changes in their levels of detail are treated with bigger blocks, and areas rich in details are treated with smaller ones.

This implementation is based on a scheme proposed by Chen (1989), where the image is divided into initial 32x32 blocks, which are in turn completely partitioned up to 4x4 blocks. A test is applied on 4 brother 4x4 sub-blocks to determine if these will be individually processed or if they can be joined to form a possible 8x8 block. The same process is done for each 4 8x8 and 16x16 brother blocks.

The test is of the following kind:

```
if abs(Mk(i)-Mk(j))>Tk    for any i=j
          Process the 4 sub-blocks individually;
Else
          Join the sub-blocks and mark them as a possible bigger one;
```

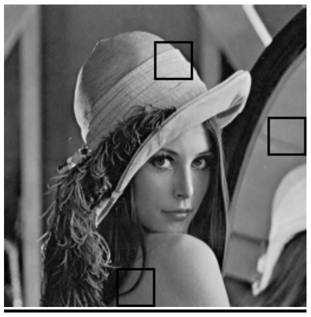
The terms Mk(i) and Mk(j) (i,j=1..4) represent averages of gray shades of brother blocks in the kth level of the tree, and Tk is the decision limit. The idea is that, in order to determine how active an NxN area is, we divide it into 4 equally-sized blocks, find their averages and see how similar they are. If there is no averages pair whose difference is greater than a pre-established limit, then we consider the NxN area as not very active (a suggested limit is 10 for 8 bits images), and we process it as a single block. In case the difference is greater than the limit, we split it.

To represent header information we only need transmit one bit for each sub-block, indicating whether it is to be partitioned or not. Thus, in the worst of cases we will codify 21 bits for each 32x32 block, which is equivalent to a cost of 0.02 bits per pixel. According to the size of each sub-block of the final partitioning (4x4, 8x8, 16x16 or 32x32), processing is similar to that applied by JPEG to each 8x8 block. The difference lies in the fact that the DC coefficient is codified as regards the same element of the previous block only if both blocks are of the same size. If the previous block is of a different size, then the same value of the quantified coefficient is codified.

Loss factors can be varied for the different block sizes according to the quality level desired for the different areas. For example, if one wishes to increase the loss in very active areas, the most convenient step to follow is to increase the loss factor for smaller blocks, since it is to be expected that these block sizes will fall into these areas. Similarly, if one wishes to decrease image quality in large, not very active areas, factors for bigger blocks should be adjusted.

In general, not very active areas belong to the background of the image, and will consequently be processed with larger blocks (32x32, 16x16).

Three examples of sequences of steps to be followed to partition a 32x32 block



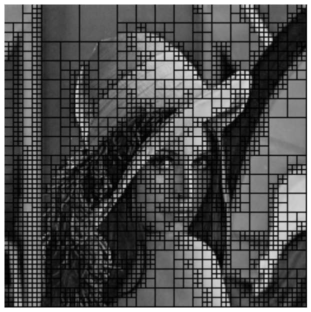
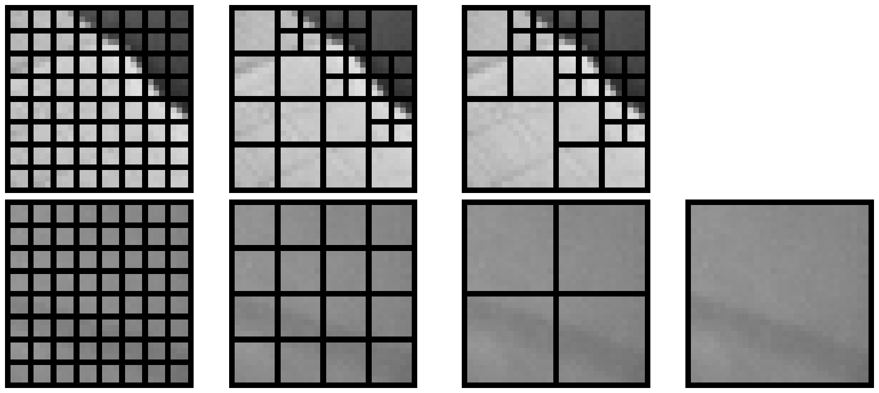
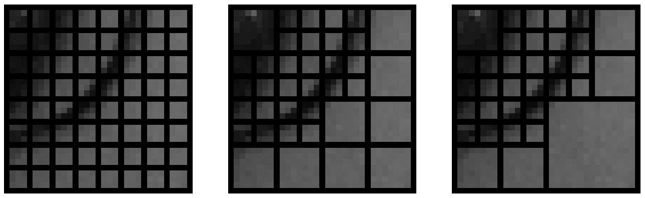


Image partitioned with limits of 20,20,20



Hat block

Mirror block



Shoulder blade block

In short, this compression technique has the following parameters which can be adjusted: Decision limit to decide whether to partition an 8x8, 16x16, 32x32 block or not; loss factor for 4x4, 8x8, 16x16 and 32x32 blocks.

Observations and some results of the adaptive method

Due to space limitations, we only analyze the results of two images with very different characteristics in this article. The results obtained from the VLSI.BMP and LENA.BMP images correspond to a partitioning of 10 and 20 respectively for each of the three limits.

For the first image partitioning, the areas with the formula were treated with smaller blocks (4x4 and 8x8), whereas most of the dark area was processed with 32x32 blocks, and with fewer 16x16 ones. With this block distribution, distortions were detected only in areas very close to the formula, and only when applying very high factors. The following tables show some of the results obtained for the images.

Even though the value of the loss factors can be increased for 8x8, 16x16 and 32x32 blocks without a visible distortion for the areas where they fall, these increments are not reflected in the size of the compressed images.

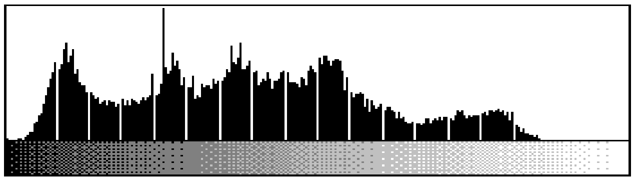
Table for VLSI.BMP

Decompressed	Ratio	Ratio Factor 4x4	Ratio Factor 8x8	Ratio Factor 16x16	Ratio Factor 32x32
pv1_1.bmp	6.4	2	2	2	2
pv1_2.bmp	9.9	9	2	2	2
pv1_3.bmp	9.98	9	9	2	2
pv1_4.bmp	11	15	9	2	2
pv1_5.bmp	13	20	9	2	2
pv1_6.bmp	14.4	25	9	2	2
pv1_7.bmp	14.5	25	9	9	9

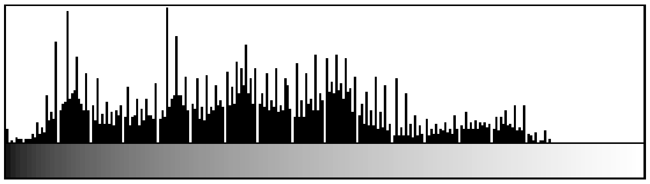
Table for LENA.BMP (with values of 20 for partitioning decision limits)

Decompressed	Ratio	Ratio Factor 4x4	Ratio Factor 8x8	Ratio Factor 16x16	Ratio Factor 32x32
plena4_1.bmp	4	2	2	2	2
plena4_2.bmp	8.3	9	2	9	9
plena4_3.bmp	6.4	5	3	5	5
plena4_4.bmp	9	10	3	10	10
plena4_5.bmp	10.8	15	2	15	15
plena4_6.bmp	11.18	15	3	15	15
plena4_7.bmp	11.4	15	4	15	15
plena4_7.bmp	13.16	20	4	15	15

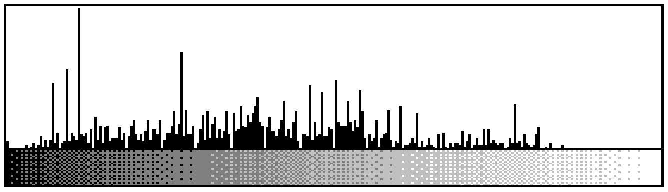
plena4_7.bmp	13.3	20	5	15	15



plena4_1.bmp

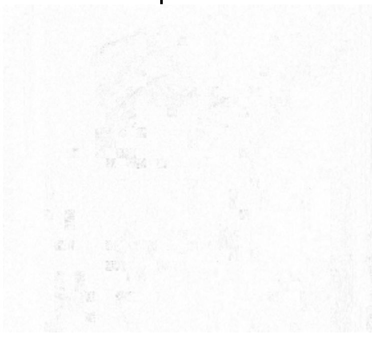
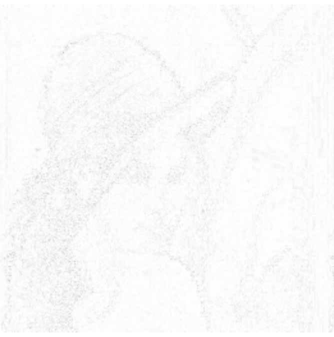


plena4_5.bmp



plena4_9.bmp

The following pictures show the loss produced as the difference between the original image and the compressed one:

Plena4_1.bmp error Plena4_5.bmp error Plena4_9.bmp error

Fidelity criteria

A natural thing to do when we are interested in the fidelity of a reconstructed sequence is to observe the differences between the original and the reconstructed values - that is, the distortion introduced during the compression process. Mean squared error and absolute difference are two well known ways of measuring the distortion or difference between the original and the reconstructed sequences. They are called *distortion difference measures*.

If {xn} is the font output and {yn} is the reconstructed sequence, then the square error measure is the following:

$$d(x,y) = (x-y)^2$$

whereas the absolute difference measure is given by:

$$d(x,y) = |x-y|$$

It is generally difficult to examine the difference on a term by term basis. Therefore, to summarize the information in the differences sequence, a set of measures of the average is used.

The most commonly used average measure is the square errors average, called *mean squared error*.

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - y_n)^2$$

If we are interested in the error size in relation to the signal, we can find the font output average square value of the ratio and the *mse*, which is called *signal-to-noise* (SNR):

$$SNR = \frac{\sigma_x^2}{\sigma_d^2}$$

The *SNR* is frequently measured in a logarithmic scale, its measurement unit being the decibel (dB):

$$SNR(dB) = 10\log_{10}\frac{\sigma_x^2}{\sigma_d^2}$$

Some times we are interested in the error size relative to the signal maximum value. This ratio is called peak signal to noise ratio (PSNR), and is given by

$$PSNR(dB) = 10\log_{10}\frac{x^2{}_{peak}}{\sigma_d^2}$$

Other widely used measure of distortion difference - although not as used as the *mse* - is the absolute difference average:

4x4 Partitioning			*8x8 Partitioning*			*16x16 Partitioning*		
image	mse	$\sqrt{mse}$	image	mse	$\sqrt{mse}$	image	mse	$\sqrt{mse}$
lenf17.bmp	11,27	3,35	lenf16.bmp	33,01	5,74	lenf4.bmp	51,28	7,16
lenf19.bmp	25,06	5,00	lenf2.bmp	56,28	7,5	lenf11.bmp	104,37	10,21
lenf21.bmp	38,53	6,20	lenf9.bmp	75,13	8,66	lenf7.bmp	153,28	12,38
lenf23.bmp	51,69	7,19	lenf12.bmp	92,73	9,62	lenf28.bmp	181,127	13,45
lenf24.bmp	64,64	8,04	lenf13.bmp	110,49	10,51			

lenf25.bmp	76,79	8,76	lenf14.bmp	129,29	11,37			
lenf26.bmp	88,18	9,39	lenf15.bmp	147,66	12,15			
lenf27.bmp	97,98	9,89						

32x32 Partitioning			***Variable Partitioning***		
image	mse	$\sqrt{mse}$	image	mse	$\sqrt{mse}$
lenf3.bmp	24,66	4,96	plen4_1.bmp	13,98	3,73
lenf10.bmp	57,81	7,6	plen4_2.bmp	33,21	5,76
lenf29.bmp	66,89	8,17	plen4_3.bmp	24,9	4,99
lenf6.bmp	92,04	9,59	plen4_4.bmp	39,07	6,25
			plen4_5.bmp	49,89	7,06
			plen4_6.bmp	53,10	7,28
			plen4_7.bmp	55,91	7,47
			plen4_8.bmp	69,46	8,33
			plen4_9.bmp	72,66	8,52

The *mse* square root is also used to know the difference in average between the original and the reconstructed values.

According to the used bibliography and to articles published on JPEG optimizations, both the SNR and the PSNR are used almost as an efficiency standard measure.

Entropy

If X represents the original image and Y the decompressed one, the definition of conditional entropy of X given Y is the amount of information of the original image that can be known by having the reconstructed one. Based on this definition - given in more detail in the bibliography - and on the sixth column of the following table, it may be concluded that the farther away from the original image, the greater the uncertainty as regards the original image in spite of knowing the reconstructed one. Despite not having a concrete measure of the loss, tables are sorted by the factor applied to quantification matrixes.

Images with 4x4 Partitioning						
image	mse	$\sqrt{mse}$	SNR(dB)	decompr. image entropy. (bits)	Conditional entropy (bits)	image entropy subtraction(bits)
lenf17.bmp	11,27	3,35	30.47	7,56	3.65	2,9
lenf19.bmp	25,06	5,00	27.00	7,43	4.08	3,38
lenf21.bmp	38,53	6,20	25.13	7,13	4.28	3,64
lenf23.bmp	51,69	7,19	23.85	6,8	4.41	3,81
lenf24.bmp	64,64	8,04	22.88	6,57	4.52	3,95
lenf25.bmp	76,79	8,76	22.14	6,35	4.60	4,05
lenf26.bmp	88,18	9,39	21.53	6,08	4.68	4,4
lenf27.bmp	97,98	9,89	21.08	5,9	4.74	4,2

Images with 8x8 Partitioning

image	mse	$\sqrt{mse}$	SNR(dB)	decompr. image entropy (bits)	conditional entropy (bits)	image entropy subtraction(bits)
lenf16.bmp	33,01	5,74	25.80	7,59	4.17	3,49
lenf2.bmp	56,28	7,5	23.48	7,54	4.48	3,84
lenf9.bmp	75,13	8,66	22.23	7,45	4.67	4,04
lenf12.bmp	92,73	9,62	21.32	7,31	4.79	4,2
lenf13.bmp	110,49	10,51	20.56	7,15	4.90	4,34
lenf14.bmp	129,29	11,37	19.87	7,04	4.98	4,46
lenf15.bmp	147,66	12,15	19.30	6,86	5.06	4,56

Images with 16x16 Partitioning

image	mse	$\sqrt{mse}$	SNR(dB)	decopmr. image entropy (bits)	conditional entropy (bits)	image entropy subtraction(bits)
lenf3.bmp	24,66	4,96	27.072221	7,59	4.083058	3,37
lenf10.bmp	57,81	7,6	23.373444	7,61	4.528039	3,88
lenf29.bmp	66,89	8,17	22.739913	7,61	4,61	3,98
lenf6.bmp	92,04	9,59	21.353333	7,61	4.7874	4,18

Images with 32x32 Partitioning

Image	mse	$\sqrt{mse}$	SNR(dB)	decompr. image entropy (bits)	conditional entropy (bits)	image entropy subtraction(bits)
lenf4.bmp	51,28	7,16	23.893925	7,61	4.505544	3,84
lenf11.bmp	104,37	10,21	20.807579	7,62	4.9126	4,29
lenf7.bmp	153,28	12,38	19.138584	7,63	5.132177	4,55
lenf28.bmp	181,127	13,45	18.413694	7,63	5.24267	4,68

Images with Variable Partitioning						
image	mse	$\sqrt{mse}$	SNR(dB)	decompr. image entropy (bits)	conditional entropy (bits)	image entropy subtraction (bits)
plen4_1.bmp	13,98	3,73	29.537773	7,57	3.62	2,9
plen4_2.bmp	33,21	5,76	25.780164	7,56	4.27	3,56
plen4_3.bmp	24,9	4,99	27.030027	7,55	4.09	3,36
plen4_4.bmp	39,07	6,25	25.07497	7,54	4.37	3,67
plen4_5.bmp	49,89	7,06	24.01287	7,48	4.48	3,8
plen4_6.bmp	53,10	7,28	23.742412	7,47	4.53	3,85
plen4_7.bmp	55,91	7,47	23.518611	7,38	4.56	3,89
plen4_8.bmp	69,46	8,33	22.575681	7,41	4.66	4,02
plen4_9.bmp	72,66	8,52	22.380217	7,33	4.69	4,06

Conclusions and Future Work

It was observed that images in gray shades with large low activity areas allowed greater compression ratios if they were processed with larger blocks. The distortions produced were not very noticeable in the low activity areas, but the losses produced in low activity areas adjacent to other highly detailed areas were significant. When reducing blocks size, these conflictive areas were restricted, but compression ratios decreased. The excellent results of the adaptive scheme when compared with the fixed one have led us to work on the parallelization of the adaptive algorithm in order to handle reasonable times for a real time processing. In addition to this, the study of adaptive quantification will be emphasized.

Bibliography

[Cla95] Roger Clarke, *Digital Compression of Still Images and Video*, Academic Press, 1995.

[Dou87] E. Dougherty, C. Giardina, *Matrix Structured Image Processing*, Prentice-Hall Inc., 1987.

[Fol90] Foley, van Dam, Feiner, Huges, *Computer Graphics*, Addison-Wesley,1990.

[Glas95-1] Andrew Glassner, *Principles of Digital Image Synthesis. Vol. 1*, Morgan Kaufmann Publishers, 1996.

[Glas95-2] Andrew Glassner, *Principles of Digital Image Synthesis. Vol. 2*, Morgan Kaufmann Publishers, 1996.

[Gon92] R. Gonzales, R. Woods, *Digital Image Processing*, Addison-Wesley, 1992.

[Jai89] Anil K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall Inc., 1989.

[Nel91] *The Data Compresión Book*. Mark Nelson, Prentice Hall, 1991.

[Say96] Khalid Sayood, *Introduction to Data Compression*, Morgan Kaufmann Publishers, 1996.