

Automatic video system for computer animation using a high performance vector processor

M.M. Millares, J.J. Casares, J.M. Ramirez, J.A. Souto

*Centro de Supercomputación de Galicia, Avda, Vigo, s/n,
15706 Santiago de Compostela, Spain*

Abstract

An automatic system for producing video-animation, based in the capabilities of vector processors is presented. This system is composed by:

- Hardware utilities: A vector processor machine, Fujitsu VP2400/10; a SUN Sparcstation with Parallax graphic card; a BETACAM/SP System video recorder.
- Software utilities: A vectorized code for compressing/decompressing JPEG files; a X-window code for RS-422 video control.

The graphic files for video production (generated by any graphics application) are converted to JPEG format at the vector processor; these JPEG files are transferred to the SUN Sparcstation, and then the Parallax graphic card decompresses them (hardware uncompressing) to the video resolution; after that, the decompressed files are saved in the BETACAM video recorder automatically. All this process is transparent to the operator.

The main advantages of this system are:

- The graphic files for video production are stored in compressed format, so they can be available faster for modifications, and they need less storage space.
- All the compressing is running on the vector processor, so the video production can be faster than in a conventional workstation.

C language routines for converting several graphic formats to JPEG are available, so the system can produce movies from other computer applications.

34 Visualization and Intelligent Design in Engineering

1. Introduction

The storage of images on a digital base is a critical step in those systems generating computer-animated video; the files must be compressed to an acceptable size for existing storing devices, and at the same time it is necessary that in this process all the essential information of the image is kept in the compressed file. The JPEG (Joint Photographic Experts Group) format is widely used for the compression of digital files of images, and it is recognized as a standard for the storage of images.

The present work provides a solution to the compression of JPEG files that reduces significantly the computational time of the process. First, the standard JPEG compression process is presented, subsequently the vectorized JPEG compression method is analyzed, and finally some of the compression results of the vectorized JPEG compression are presented, comparing the response time with that of the non-vectorized compression.

2. Standard JPEG compression

The standard JPEG compression allows for two different approaches: one based on the discrete cosine transform, DCT, that provides a compression with small losses of information; another of a predictive type that obtains a compression without any loss of information. The selection of one or another depends on the quality of the image to be saved in the compression. For this work, JPEG-DCT compression is adopted as the compression is better and 95% of the original information is saved with hardly any degradation of the quality of the image.

The steps to follow when applying this method are:

- For color images, the transformation of matrix RGB to YCbCr.
- Computation of the DCT of the original matrix of the image.
- Quantization of the results obtained from the transform.
- Huffman coding of the resulting pixels matrix.

The vectorized code comprises the first three steps of the process: transformation to YCbCr, DCT, and quantization. The fourth step, Huffman coding, is hardly vectorizable, as this process treats each code as an arbitrary number of bits.

2.1. The Discrete Cosine Transform

The Discrete Cosine Transform, DCT, is a mathematical transform of the family of the Fourier transforms. The classical forms of the DCT and its inverse

are presented in equations (1) and (2). For its application to image compressing, the JPEG committee defined as the basic information block that of 8x8 size, and therefore $N=8$ in equation (1). That is, prior to the compression of the images file, it must be divided in cells of 8x8 pixels, and these will be transformed. Equation (2) is applied in the decompression stage of JPEG.

$$DCT(i, j) = \frac{1}{\sqrt{2N}} C(i) C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \text{pixel}(x, y) \cos\left[\frac{(2x+1)i\pi}{2N}\right] \cos\left[\frac{(2y+1)j\pi}{2N}\right] \quad (1)$$

$$C(x) = \frac{1}{\sqrt{2}} \quad \text{if } x \text{ is 0, else } 1 \text{ if } x > 0$$

$$\text{pixel}(x, y) = \frac{1}{\sqrt{2N}} C(i) C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} DCT(i, j) \cos\left[\frac{(2x+1)i\pi}{2N}\right] \cos\left[\frac{(2y+1)j\pi}{2N}\right] \quad (2)$$

$$C(x) = \frac{1}{\sqrt{2}} \quad \text{if } x \text{ is 0, else } 1 \text{ if } x > 0$$

2.2. The RGB-YCbCr Transform

In those cases where the images are levels of gray, the application of the DCT is straight forward. But, for RGB (Red, Green and Blue) colour images it is compulsory to do a previous conversion to colour format YCrCb (luminance, red chrominance, blue chrominance):

$$Y = 0.2999R + 0.587G + 0.114B \quad (3a)$$

$$Cb = -0.1687R - 0.3313G + 0.5B + 128 \quad (3b)$$

$$Cr = 0.5R - 0.4187G - 0.0813B + 128 \quad (3c)$$

Equation (3a) for luminance is applied to all pixels of the image, whilst equations (3b) and (3c) for the chrominances are applied to a smaller number of pixels, as human sight is less sensitive to the chrominances and the loss of information doesn't have a significant effect. This process is known as sampling.

Once the DCT is applied over the YCrCb image, most of the image information is stored in the top left hand-side corner of the compressed matrix, and more specifically when the image presents continuous tones. The value stored in the (0,0) element (DC element) is the most significant, and so it must be handled separately. The rest of the elements are known as the AC elements. Therefore, in a compressed matrix of 64 elements there will be 1 DC element and 63 AC elements.

36 Visualization and Intelligent Design in Engineering

2.3. Quantization

Quantization is a relatively simple process that consists of the reduction of the bits necessary to store the integer values that identify the properties of every pixel, by reducing them to the precision of an integer. Equation (4) corresponds to the quantization expression used,

$$\text{Quantization Value}(i,j) = \text{Integer round} \left(\frac{\text{DCT}(i,j)}{\text{Quantum}(i,j)} \right) \quad (4)$$

where $\text{Quantum}(i,j)$ is the quantization matrix, defined in this case by the specifications of the digitizing-decompressing board Parallax [5].

Once the quantization formula is applied over the whole matrix that results from the DCT, a reduction on the value of the DC element is done; this reduction is achieved subtracting from the value of the DC of the block, the DC value of the 8x8 block previously quantized. The DC of the first block is not modified.

2.4. The Huffman coding

The Huffman code acts over the 8x8 blocks, coding each pixel according to the appearance frequency of the value stored in the pixel to a new value (category); previously a statistical analysis is done on the appearance frequency of the values of the pixels that are to be coded, to define the corresponding categories. The most frequent value is assigned the shortest code used in the definition of the categories, and the least frequent the longest code, achieving a significant reduction.

The Huffman coding is done in an ordered manner all along the image, in a zig-zag sequence as shown in Figure (1). The operation starts at the top left hand-side block. As it has already been mentioned, most of the information of the image matrix, once it is compressed, is concentrated in this zone. After a certain element, the values in the pixels are null, and these values are not coded. This last operation attains the maximum compression ratio.

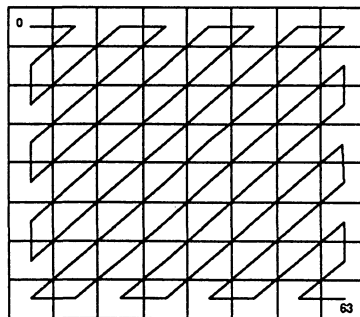


Figure 1. Zig-Zag coding.

As a summary of all explained so far, Figure (2) contains a scheme of all the steps followed for a standard JPEG compression.

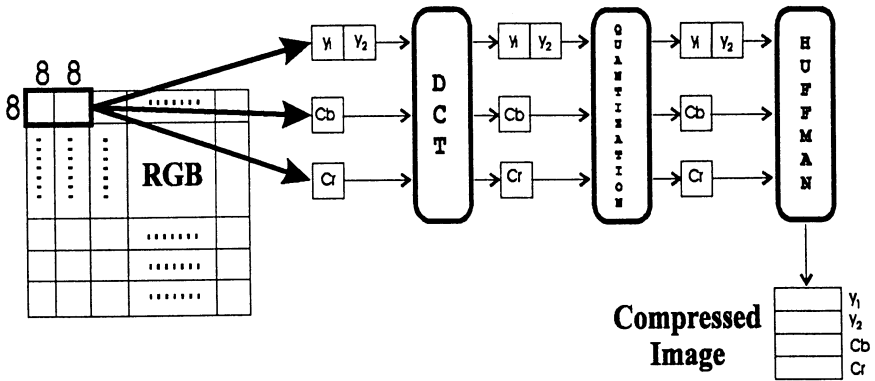


Figure 2. Standard JPEG Compression Scheme.

3. Vectorized JPEG

The vectorization of the JPEG-DCT compression algorithm has been done for each of the compression steps separately, on ANSI C code, portable to any machine and system that supports this language, but as it is a vectorized code it performs better when used in computers with a vector processor.

3.1. Vectorized RGB-YCrCb transform

For this case a YCrCb 4:2:2 sampling is applied, so the corresponding matrices have different sizes; this implies an additional complexity for the vectorizing of the code, that has been solved by storing the indexes of the corresponding elements of the three matrices on a correspondence matrix. With this sampling ratio, for every four values of the luminance obtained from the pixels of the original matrix, only two values are considered for Cr, and also two values for Cb, corresponding to only two of the four pixels.

3.2. Vectorized DCT

For the vectorization of this step it is necessary to reformulate equation (1), that defines the DCT, to change it into a product of matrices. To obtain the DCT as a product of matrices the operation manipulation shown in equation (5). This expression results in the product of matrices as shown in equation (6).

38 Visualization and Intelligent Design in Engineering

$$C_{i,j} = \begin{cases} \frac{1}{\sqrt{N}} & \text{if } i = 0 \\ \sqrt{\frac{2}{N}} \cos\left[\frac{(2j+1)i\pi}{2N}\right] & \text{if } i > 0 \end{cases} \quad (5)$$

$$\text{IDCT} = C_t * \text{Pixels} * C$$

$$\text{DCT} = C * \text{Pixels} * C_t \quad (6)$$

where C_t corresponds to the transpose of C .

However, this product refers to 8×8 matrices that allow a very inefficient vectorization due to the small number of elements. If equations (5) and (6) are analysed in detail, it is observed that the value of equation (5) is constant as it depends on the position of each pixel, not of its value and therefore this operation can be done at the beginning of the process, and not repeated for each 8×8 block.

After matrices C and C_t are calculated, the original matrix of pixels, of $W \times H$ dimension, is transformed into a new matrix that contains all the 8×8 blocks aligned; that is, a matrix known as WK1 of dimension $(W \times H/8, 8)$. At this stage, it is possible to multiply matrices WK1 and C , line by line, so that an external loop of large size, that allows a high degree of vectorization, is obtained. The basics of this procedure are presented in Figure 3.

Once the new matrix is obtained, the transpose is calculated in 8×8 blocks, and it is multiplied by matrix C . With this operation a new matrix, WK2, is obtained, and contains the transformed and transposed values in 8×8 blocks. Note that equation (6) does a different operation, but as it will be shown later the final result of the compression process doesn't change.

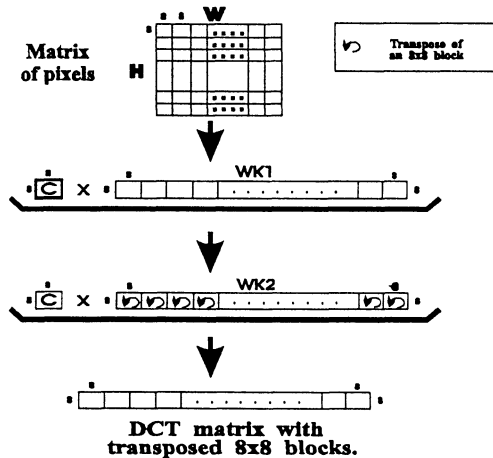


Figure 3. Vectorized DCT.

3.3. Vectorized Quantization

The quantization matrix is obtained from the original matrix, multiplied by the quantization factor. This factor is the inverse of the quotient of a value supplied by the user and 50. The value supplied must be in the range 1 to 100. Thus, the quantization factor can be less or greater than 1. If it is smaller than 1, the quantized matrix will lose less information and the image of the compressed file will be similar to the original but will take more storage space; otherwise, there will be a loss of clarity in the final image.

Then, the WK2 matrix is transformed in a one dimensional array of $W \times H$ elements (Figure 3). Hence, with the transpose matrix it can be set a loop of size $(W \times H/8)$ to do the operation between the two matrices, a loop that is highly vectorizable. Each pixel is multiplied in this fashion with its value in the quantization matrix.

Finally, it should be mentioned that the quantization matrices for the luminance and the chrominance are different and depend on the digitizing-decompressing board being used later for the visualization of the image. In this case, the quantization matrices are those that correspond to the Parallax board [5].

4. Results and conclusions

As a study case for the application of the algorithm developed here, the image shown in Figure 4 was compressed. The dimension of the figure is 768x575, and the file that contains it has SUN raster 32 bit format, so its size is 1766432 bytes.

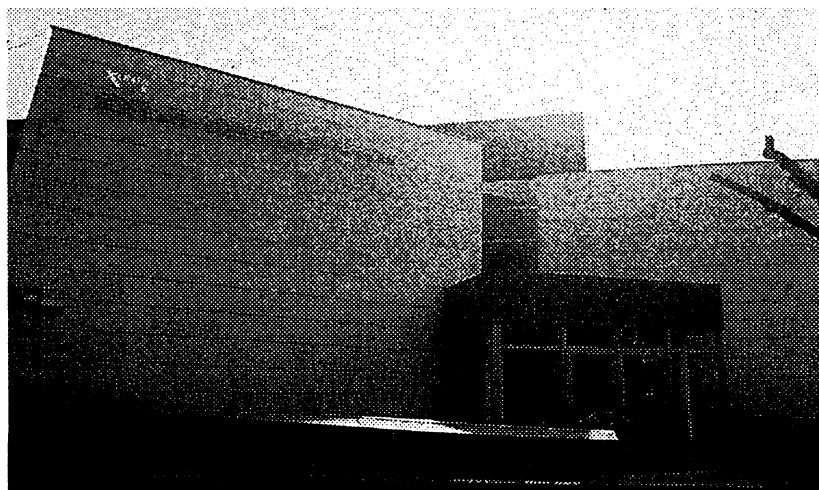


Figure 4. Main entrance to the Centro de Supercomputacion de Galicia.



40 Visualization and Intelligent Design in Engineering

The results obtained for the different quantization factors tried are shown in Table 1. The compression times necessary for each case are presented in Table 2, and correspond to the compression of one image, one second of animation (25 frames, on VHS) and 15 seconds (375 frames on VHS).

Quantization Factor	Size of compressed image	Compression (%)
25	173267 bytes	90.19
50	87421 bytes	95.05
75	52510 bytes	97.02

Table 1. Percentage of compression relative to the quantization factor.

	Non-Vectorized compression			Vectorized compression		
Time	Vectorizable Code	Non-Vectorizable Code	Total	Vectorizable Code	Non-Vectorizable Code	Total
1 frame	2.42	0.30	2.71	0.072	0.298	0.37
1 second	58.82	7.49	66.31	1.78	7.46	9.24
15 seconds	871.80	112.41	984.21	26.72	112.14	138.86

Table 2. Comparative results for vectorized and non-vectorized compression in seconds.

From the results of Table 2, the following factors can be obtained for the case that one second of video is compressed. It becomes obvious that the vectorization applied is highly efficient as it takes place over 87.95% of the total time of compression and achieves an execution time 33 times lower than the equivalent not vectorized.

$$\% \text{ Vectorization, } V = \frac{\text{vectorizable code, time}}{\text{Total time}} = \frac{58.82 \text{ s.}}{66.31 \text{ s.}} = 87.95 \%$$

$$\text{Scalar / Vector ratio, } \alpha = \frac{\text{vectorizable code, time}}{\text{vectorized code, time}} = \frac{58.82 \text{ s.}}{1.78 \text{ s.}} = 33.04$$



References

1. Eric Hamilton "JPEG File Interchange Format, v.1.02". C-Cube Microsystems, September 1992.
2. G.K. Wallance. "The JPEG still picture compresion standard". Multimedia Engineering, Digital Equipment Corporation, December 1991.
3. D.C. Kay, J.R. Lavine. "Graphics file formats". Windcrest/Mc Graw-Hill, 1992.
4. Mark Nelson . "The Data Compression Book". Prentice-Hall, 1991.
5. XVIDEO Software Developer's Guide for SBUS Systems, version 3.0. Parallax Graphic, Inc, 1991.