Lossy Image Compression with Refactoring Matrices

Osslan O. Vergara Villegas[1], Raúl Pinto Elías[1], Marcos A. Capistrán Ocampo[2]

[1]Centro Nacional de Investigación y Desarrollo Tecnológico (cenidet), Cuernavaca, Morelos México.
[2]Universidad Autónoma del Estado de Morelos (UAEM), Cuernavaca, Morelos México.
{osslan, rpinto}@cenidet.edu.mx, marcos.capistran@gmail.com

Abstract

We show a method of linear algebra known as Singular Value Decomposition (SVD) that allows making the refactoring of a matrix (digital image). The use of the resulting singular values of such refactoring allows us to represent the matrix with a smaller number of values, reason why the method is applicable to image compression. We show the preliminary results of the constructed system for compression. The primary target of the SVD compression scheme is the use of the smaller number of ranks (singular values) to approximate the original matrix.

Keywords: Image Compression, Singular Value Decomposition, Matrix, Linear Algebra.

1. Introduction

Data compression is a much studied topic during the last three decades [1], [2]; its target is to reduce the volume of necessary data to represent a certain information amount. In this paper we show an application of the refactoring matrix to compress images, we show experimental results with test images and some possible improvements to the compression method are recommended.

A digital colour image can be seen like a matrix of *m * n * 3* size, the result of such multiplication provides the size that will use the image when storing it in a hard drive. With the increase of the possibilities of electronic processing of the present time saving storage space is necessary to keep great amounts of information.

An image can be seen like a matrix, then, the operations that are applied to the matrices can be applied to the images.

In order to obtain the new representation of the matrix we use a significant topic of linear algebra called: "Singular Value Decomposition (SVD)" [3]. The method do the refactoring of matrix *A* in three new matrices *U*, *S*, and *V* in such way that $A = USV^T$. Where *U* and *V* are orthogonal matrices and *S* is a diagonal matrix.

The main reason to use the SVD in image compression is his property of energy compaction and its ability to adapt to the local statistical variations of an image [2]. The SVD exists for any arbitrary, square, reversible and non reversible matrix of *m x n* size [3]. In the following sections we show the process of SVD compress/decompress, as well as the results of the experiments made.

2. The SVD Process

SVD process is strong related to the theory of diagonalization of a symmetrical matrix and generally is applied in problems that deal with matrices of great size. Some of the applications are: design of separable filters (FIR), search of minimum squares in linear equations, in digital image processing as a method of noise reduction, image restoration and image compression.

For the SVD suppose that we have a matrix *A* with *m* lines and *n* columns, the goal is to refactoring *A* in three new matrices: *U*, *S*, and V^T [4].

$$A = (u_1 \ldots u_r \ldots u_m) \begin{pmatrix} \sigma_1 & & & \\ & \ldots & & \\ & & \sigma_r & \\ & & & \ldots \\ & & & 0 \end{pmatrix} \begin{pmatrix} v_1^T \\ \ldots. \\ v_r^T \\ \ldots. \\ v_n^T \end{pmatrix} \quad (1)$$

Where *U* is of *m x m* size, *S* of *m x n* (the number of non zero elements of the diagonal determine the rank of the original matrix) and *V* of *n x n*. *U* contains the singular vectors of the left, V the singular vectors of the right and S contains the singular values, which are allocate in the main diagonal so that:

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots. = \sigma_p = 0$$

where *r* is the rank of the matrix *A* and *p* is the smallest value of dimensions *m* or *n*.

In the literature there we found various methods to compute SVD, the more used is the traditional approach that has six steps [5]: 1. Make the

multiplication A^TA. 2. Find the eigenvalues of A^TA. 3. Build S^2 matrix (allocate the eigenvalues in the main diagonal in decreasing order). 4. Compute the square root of the S^2 matrix. 5. Find the eigenvectors of A^TA, to form the columns of v. 6. Find the eigenvectors for AA^T, to form the columns of u. The problem is that this algorithm is unstable.

A preferable method for computing SVD is described in Golub and Kahan [6], their technique brought the matrix into bidiagonal form and then the bidiagonal matrix is diagonalized.

The process to obtain the SVD with Golub-Kahan method is described in the next section.

2.1. SVD Algorithm

1. Reduce A to upper bidiagonal form by using Householder transformation.

$$U_B^T A V_B = \begin{bmatrix} B \\ 0 \end{bmatrix} \qquad B = \begin{bmatrix} d_1 & f_1 & & \cdots & 0 \\ 0 & d_2 & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & f_{n-1} \\ 0 & \cdots & & 0 & d_n \end{bmatrix}$$

2. Multiply B^TB.

3. Apply the QR algorithm to B^TB to obtain USV^T.

$$B^T B = V_B \sum {}^2 V_B{}^T = (U_B \sum V_B{}^T)^T *$$

$$U_B \sum V_B{}^T \Rightarrow B = U_B \sum V_B{}^T$$

Phase 1 involves O (mn^2) flops and phase 2 requires O (n^2) flops.

Section 3 shows an example of this algorithm.

3. SVD Example

In order to illustrate the SVD algorithm we show an example for the following matrix A:

$$A = \begin{bmatrix} 2 & 2 & 2 & 1 \\ 2 & -3 & 1 & 1 \\ 2 & 1 & 3 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix}$$

1) After the process of householder to bidiagonal form of A we obtain:

$$U = \begin{bmatrix} 0.7206 & 0.1499 & -0.4771 & -0.4803 \\ 0.0801 & 0.8736 & 0.4737 & -0,0778 \\ 0.5604 & 0.0130 & 0.0171 & 0.8279 \\ 0.4003 & -0.4629 & 0.7400 & -0.2790 \end{bmatrix}$$

$$V = \begin{bmatrix} 0.5000 & 0.7071 & -0.0277 & -0.4992 \\ 0.5000 & -0.4714 & -0.7153 & -0.1273 \\ 0.5000 & 0.2357 & 0.0461 & 0.8321 \\ 0.5000 & -0.4714 & 0.6968 & -0.2056 \end{bmatrix}$$

$$B = \begin{bmatrix} 6.2450 & 2.0381 & 0 & 0 \\ 0 & 2,7812 & 2.0406 & 0 \\ 0 & 0 & 2.2179 & 0.2508 \\ 0 & 0 & 0 & 1.4018 \end{bmatrix}$$

2) After the multiplication of B^TB we obtain:

$$B^T B = \begin{bmatrix} 39.0000 & 12.7279 & 0 & 0 \\ 12.7279 & 11.8889 & 5.6754 & 0 \\ 0 & 5.6754 & 9.0831 & 0.5561 \\ 0 & 0 & 0.5561 & 2.0280 \end{bmatrix}$$

3. After apply the QR algorithm to B^TB we obtain.

$$U = \begin{bmatrix} -0.7258 & -0.2209 & -0.5926 & 0.2706 \\ -0.2419 & 0.9658 & -0,0226 & 0.0902 \\ -05544 & -0.0606 & 0.3601 & -0.7479 \\ -0.3276 & -0.1212 & 0.7202 & 0.5994 \end{bmatrix}$$

$$S = \begin{bmatrix} 6.6458 & 0 & 0 & 0 \\ 0 & 3.6458 & 0 & 0 \\ 0 & 0 & 1.6458 & 0 \\ 0 & 0 & 0 & 1.3542 \end{bmatrix}$$

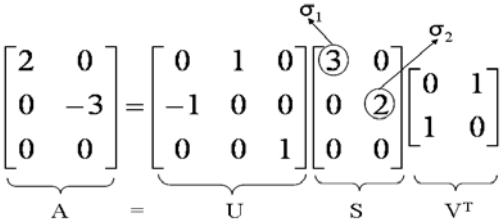
$$V = \begin{bmatrix} -0.7258 & 0.2209 & -0.5926 & 0.2706 \\ -0.2419 & -0.9658 & -0.0226 & 0.0902 \\ -0.5544 & 0.0606 & 0.3601 & -0.7479 \\ -0.3726 & 0.1212 & 0.7202 & 0.5994 \end{bmatrix}$$

With the multiplication of USV^T we can obtain A again.

4. SVD Compression

As we were seen in section 1 an image of *m x n* needs for its storage *m x n* space. In order to store A_k* (approach of SVD of the original matrix) we need *n x k* elements for *U*, *m * k* for V^T and *k* for *S,* for a total of *(n + m + 1) * k* elements.

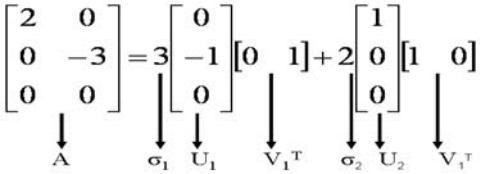
For the compression example we have the matrix *A* with its respective SVD:



From the *S* matrix observes that have two singular values σ_1 and σ_2 therefore the matrix *A* can be approximated in a rank *k* by the matrix *A** as:

$$A^* = \sum_{i=1}^{k} u_i s_i v_i^T \qquad (2)$$

When we do the respective arrangement of the matrix we obtain:



Then the matrix can be represented as:

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + ... + \sigma_r u_r v_r^T \qquad (3)$$

If we begins with a matrix of *m x n* (where *m* = *n*) we required m^2 values to represent the original image, after the SVD we need $2m^2 + m$ values for the representation of the image. This does not represent saving of space but an increase.

An approach to solve the previous problem is not to store the singular values in a separated form, which can be done is to add the singular values in the unitary vectors u_i or v_i, and then equation 2 becomes:

$$A^* = \sum_{i=1}^{k} a_i b_i^T ; a_i = \sqrt{s_i} u_i ; b_i = \sqrt{s_i} v_i \qquad (4)$$

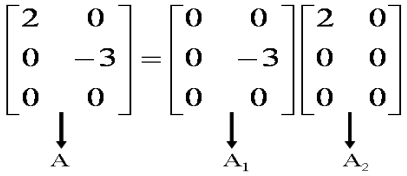
In order to store the matrices in a file for its next reconstruction (decompression) a quantization strategy is used for a compact representation of the values (regularly the values of the matrices are real numbers).

5. SVD Decompression

The original image can be reconstructed by SVD as an exact copy by means of the following equation:

$$A^* = A_1 + A_2 + ... A_k \qquad (5)$$

The sum of the different A´s will be able to reconstruct the image in an exact way:



In order to obtain the compression goal, the rank (matrices A_1, A_2,...A_k) used to reconstruct an image must be smaller than:

$$m^2 / (1 + 2m) \qquad (6)$$

The most important thing, is that when using a smaller rank than the original, the image can almost be reconstructed without differences for the observer since in the first ranks is concentrated most of the energy, but using less space of storage.

The difference between the original image and reconstructed is given by the Frobenius norm *F* and norm *2* relative error:

$$E = \frac{\left\| A - A_k^* \right\|_{F,2}}{\left\| A \right\|_{F,2}} \qquad (7)$$

Other error measures like MSE and PSNR are used.

6. Experimentation and Results

To verify the performance of the compress/decompress process, we use images in gray scale and colour, the compression factor are given by the following equation:

$$r = (n + m + 1) * k / nm \qquad (8)$$

Figure 1 shows an example of two images used for the system tests. Incise *a* and *d* show the original images (Lena and Birds), whereas incise *b* and *f* show the results of the reconstruction using 20 singular values, incise *c* and *g* show the results using 50 values

and incise *d* and *h* the results with 100 values. Table 1 shows a summary of the results obtained in terms of the storage space.

Table 1. Image compression results.

Images	**Space in kb**			
	Orig	***20 sv***	***50 sv***	***100 sv***
Lena (256x256)	65	10	25	50
Birds (512 x 512)	768	60	150	300

Table 2 shows the error measures for the tested images.

Table 2. Image compression error measures.

Error	***Lena***			***Birds***		
	20 sv	***50 sv***	***100 sv***	***20 sv***	***50 sv***	***100 sv***
MSE	$3.1e^{-2}$	$8.28e^{-4}$	$1.53e^{-4}$	$1.6e^{-2}$	$4.4e^{-4}$	$1.07e^{-4}$
PSNR	73.25	78.94	86.28	76.13	81.61	87.80
Frobenius	0.1260	0.0655	0.0281	0.0897	0.0475	0.0229
Norm 2	0.0335	0.0139	0.0056	0.0678	0.0294	0.0144

With the obtained results it is possible to observe that with the interval of the first 10 to 20 singular values one begins to observe the form of the image and with the increase of singular values we have a better approach to the original image.

7. Conclusions and Further Works

We present a method of refactoring matrices that can be used for lossy image compression. The obtained compression factor is not better than the one than provides commercial compressors like JPEG.

A disadvantage of SVD is that it is not fast from the computational point of view, but has the advantage to adapt to the statistical variations of the images.
There are portions of an image that are simple that only needs a few singular values to obtain the approximation and the complex parts needs to use more values to maintain their quality.

When analyzing an image we can conclude that the image does not require a same *k* in its totality, a future work consists of dividing the image in blocks (powers of two) and applying a single *k* for each block, the challenge is to calculate *k* for each block and the size of each block.

With those improvements some relations between the singular values and the characteristics of the image will be able to be looked for (edge, textures, etc) that trying to find some form to compress preserving this information. Another work would be to change RGB images to another system for example HSI or YIQ.

8. References

[1] J. R. Clarke, "Image and video compression: A survey", Computer department of the Heriot – Watt university, Riccarton Scotland 1999.
[2] A. K. Jain, *Fundamentals of digital image processing*. Prentice-Hall, 1989.
[3] Mande M., "Singular Value Decomposition", Department of computer science of the Technological Institute of Bombay India, august 2003.
[4] Arnold B., "An Investigation into using SVD as a method of Image Compression", The Canterbury University, UK, September 2002.
[5] Richards D. and Abrahamsen A. "Image compression using singular value decomposition", linear algebra applications, 2001.
[6] Golub G. H. and Van Loan C., *Matrix computations*, Johns Hopkins University press, 1996.

Fig. 1: Utilized images for the compress/decompress process.