Singular Value Decomposition Applied To Digital Image Processing

Lijie Cao

Division of Computing Studies
Arizona State University Polytechnic Campus
Mesa, Arizona 85212
Email Lijie.cao@asu.edu

ABSTRACT

This project has applied theory of linear algebra called "singular value decomposition (SVD)" to digital image processing. Two specific areas of digital image processing are investigated and tested. One is digital image compression, and other is face recognition. SVD method can transform matrix A into product USV^T, which allows us to refactoring a digital image in three matrices. The using of singular values of such refactoring allows us to represent the image with a smaller set of values, which can preserve useful features of the original image, but use less storage space in the memory, and achieve the image compression process. The experiments with different singular value are performed, and the compression result was evaluated by compression ratio and quality measurement. To perform face recognition with SVD, we treated the set of known faces as vectors in a subspace, called "face space", spanned by a small group of "base-faces". The projection of a new image onto the base-face is then compared to the set of known faces to identify the face. All tests and experiments are carried out by using MATLAB as computing environment and programming language.

KEY WORDS:
Image processing, Image Compression, Face recognition, Singular value decomposition.

1. INTRODUCTION

Image processing is any form of information processing, in which the input is an image. Image processing studies how to transform, store, retrieval the image. Digital image processing is the use of computer algorithms to perform image processing on digital images.

Many of the techniques of image processing were developed with application to satellite imagery, medical imaging, object recognition, and photo enhancement. With the fast computers and signal processors available in the the 2000s, digital image processing has become the most common form of image processing, and is generally used because it is not only the most versatile method, but also the cheapest [6].

1.1 Digital Image Processing

An image can be defined as a two-dimension function f (x, y) (2-D image), where x and y are spatial coordinates, and the amplitude of f at any pair of (x, y) is gray level of the image at that point. For example, a grey level image can be represented as:

f_{ij} Where $f_{ij} \equiv f(x_i, y_j)$

When x, y and the amplitude value of f are finite, discrete quantities, the image is called "a digital image". The finite set of digital values is called *picture elements* or pixels. Typically, the pixels are stored in computer memory as a two-dimensional array or matrix of real number.

Color images are formed by a combination of individual 2-D images. Many of the image processing techniques for monochrome images can be extend to color image (3-D) by processing the three components image individually [2].

Digital Image Processing (DIP) refers to processing a digital image by mean of a digital computer, and the study of algorithms for their transformation. Since the data of digital image is in the matrix form, the DIP can utilize a number of mathematical techniques. The essential subject areas are computational linear algebra, integral transforms, statistics and other techniques of numerical analysis. Many DIP algorithms can be written in term of matrix equation, hence, computational method in linear algebra become an important aspect of the subject [3].

Digital Image processing encompasses a wide and varied field of application, such as area of image operation and compression, computer vision, and image analysis (also called image understanding). There is the consideration of three types of computerized processing: low-level processing is characterized by that both its inputs and outputs are images; mid-level processing on images is characterized by the fact that its inputs are images, but outputs are attributes extracted from those images, while higher-level processing involves "making sense" of an ensemble of recognized objects as in image analysis, and performing the cognitive function associated with human vision [3].

In particular, digital image processing is the practical technology for area of:

- Image compression
- Classification
- Feature extraction
- Pattern recognition
- Projection
- Multi-scale signal analysis

1.2 Objective of the Project

The objective of this project is to apply linear algebra "Singular Value Decomposition (SVD) "to mid-level image processing, especially to area of image compression and recognition. The method is factoring a matrix A into 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 experiments are conducted under different term k of singular value, and the outer product expansion of image matrix A for image compression; this project also demonstrates how to use SVD approach for image processing in area of Face Recognition (FR).

In this project, we assume a matrix A with m lines and n columns, $m \geq n$, this assumption is made for convenience only, all the result will also hold if $n \geq m$ [8].

MATLAB is used as a platform of programming and experiments in this project, since MATLAB is a high-performance in integrating computation, visualization and programming.

The reminder of this project is organized as follows: section2 describes the theory of Singular Value Decomposition; the section3 is methodology for applying SVD to image processing, section4 shows the experimentations and results obtained. Section5 explains my own contribution to this project. Finally, section6 presents the conclusion and the further work proposed.

2 THEORY OF SINGULAR VALUE DECOMPOSITION

2.1 Process of Singular Value Decomposition

Singular Value Decomposition (SVD) is said to be a significant topic in linear algebra by many renowned mathematicians. SVD has many practical and theoretical values; special feature of SVD is that it can be performed on any real (m, n) matrix. Let's say we have a matrix A with m rows and n columns, with rank r and $r \leq n \leq m$. Then the A can be factorized into three matrices:

$$A = USV^T \tag{1}$$

(See the figure1 for illustration)

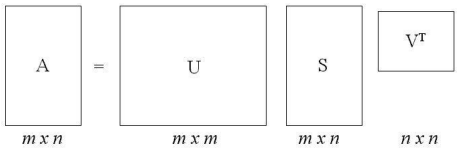


Figure1. Illustration of Factoring A to USV^T

Where Matrix U is an $m \times m$ orthogonal matrix

$$U = [\mathrm{u}_1, \mathrm{u}_2, ...\mathrm{u}_r, \mathrm{u}_{r+1}, ..., \mathrm{u}_m] \tag{2}$$

column vectors $\mathbf{u}_i$, for i = 1, 2, …, m, form an orthonormal set:

$$\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij} = \begin{cases} 1,,, i = j \\ 0,,, i \neq j \end{cases} \tag{3}$$

And matrix V is an $n \times n$ orthogonal matrix

$$V = [\mathrm{v}_1, \mathrm{v}_2, ...\mathrm{v}_r, \mathrm{v}_{r+1}, ..., \mathrm{v}_n] \tag{4}$$

column vectors $\mathbf{v}_i$ for i = 1, 2, …, n, form an orthogormal set:

$$\mathrm{v}_i^T \mathrm{v}_j = \delta_{ij} = \begin{cases} 1,,, i = j \\ 0,,, i \neq j \end{cases} \tag{5}$$

Here, S is an $m \times n$ diagonal matrix with singular values (SV) on the diagonal. The matrix S can be showed in following

$$S = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & \sigma_{r+1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & \sigma_n \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{6}$$

For i = 1, 2, …, n, σ_i are called Singular Values (SV) of matrix A. It can be proved that

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0, \text{ and}$$
$$\sigma_{r+1} = \sigma_{r+2} = \cdots = \sigma_N = 0. \tag{7}$$

For i = 1, 2, …, n, σ_i are called Singular Values (SVs) of matrix A. The $\mathbf{v}_i$'s and $\mathbf{u}_i$'s are called right and left singular-vectors of A [1].

2.2 Properties of the SVD

There are many properties and attributes of SVD, here we just present parts of the properties that we used in this project.

1. The singular value $\sigma_1, \sigma_2, \cdots, \sigma_n$ are unique, however, the matrices U *and* V are not unique;
2. Since $A^T A = VS^T SV^T$, so V diagonalizes $A^T A$, it follows that the $\mathbf{v}_j$ s are the eigenvector of $A^T A$.

3. Since $AA^T = USS^T U^T$, so it follows that U diagonalizes AA^T and that the u_i 's are the eigenvectors of AA^T.
4. If A has rank of r then $\mathbf{v}_j$, $\mathbf{v}_j$, …, $\mathbf{v}_r$ form an orthonormal basis for range space of A^T, $R(A^T)$, and $\mathbf{u}_j$, $\mathbf{u}_j$, …, $\mathbf{u}_r$ form an orthonormal basis for .range space A, $R(A)$.
5. The rank of matrix A is equal to the number of its nonzero singular values [4].

3. METHODOLOGY OF SVD APPLIED TO IMAGE PROCESSING

3.1 SVD Approach for Image Compression

Image compression deals with the problem of reducing the amount of data required to represent a digital image. Compression is achieved by the removal of three basic data redundancies: 1) coding redundancy, which is present when less than optimal; 2) interpixl redundancy, which results from correlations between the pixels; 3) psychovisual redundancies, which is due to data that is ignored by the human visual [2].

The property 5 of SVD in section 2 tells us "the rank of matrix A is equal to the number of its nonzero singular values". In many applications, the singular values of a matrix decrease quickly with increasing rank. This propriety allows us to reduce the noise or compress the matrix data by eliminating the small singular values or the higher ranks.

When an image is SVD transformed, it is not compressed, but the data take a form in which the first singular value has a great amount of the image information. With this, we can use only a few singular values to represent the image with little differences from the original.
To illustrate the SVD image compression process, we show detail procedures:

$$A = USV^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T \qquad (11)$$

That is A can be represented by the outer product expansion:

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T \qquad (12)$$

When compressing the image, the sum is not performed to the very last SVs, the SVs with small enough values are dropped. (Remember that the SVs are ordered on the diagonal.)
The closet matrix of rank k is obtained by truncating those sums after the first k terms:

$$A_k = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_k u_k v_k^T \qquad (13)$$

The total storage for A_k will be

$$k\,(m + n + 1) \qquad (14)$$

The integer k can be chosen confidently less then n, and the digital image corresponding to A_k still have very close the original image. However, the chose the different k will have a different corresponding image and storage for it. For typical choices of the k, the storage required for A_k will be less the 20 percentage. In this project, experiment and testing for different k are carried out and the result will show in section 4.

3.2 Image Compression Measures

To measure the performance of the SVD image compression method, we can computer the compression factor and the quality of the compressed image. Image compression factor can be computed using the Compression ratio:

$$C_R = m*n/(k\,(m + n + 1)) \qquad (15)$$

To measure the quality between original image A and the compressed image A_k, the measurement of Mean Square Error (MSE) [10] can be computed:

$$MSE = \frac{1}{mn}\sum_{y=1}^{m}\sum_{x=1}^{n}(f_A(x,y) - f_{A_k}(x,y)) \quad (16)$$

3.3 SVD Approach for Face Recognition

Over the past decades, face image compression, representation and recognition has drawn wide attention from researchers in arrears of computer vision, neural network, pattern recognition, machine learning, and so on. The application of face recognition includes:
Access Control based on the face recognition, Computer -human interaction, Information Security, Law enforcement, Smart Car etc. [11]

Several approaches to face recognition have been proposed for the 2-dimensional facial recognition. Much of the work has focused on detecting individual features such as eyes, nose, mouth, and head outline, and defining a face model by the position, size, and relationships among these features [12][13].

SVD approach treats a set of known faces as vectors in a subspace, called "face space", spanned by a small group of "base-faces"[1]. It likes Principal Component Analysis (PCA) [14], recognition is performed by projecting a new image onto the face space, and then classifying the face by comparing its coordinates (position) in face space with the coordinates (positions) of known faces. However, the SVD approach has better numerical properties than PCA.

In this case, we redefined the matrix A as set of the training face. Assume each face image has $m \times n = M$ pixels, and is represented as an $M \times 1$ column vector $\mathbf{f}_i$, a 'training set' S with N number of face images of known individuals forms an $M \times N$ matrix:

$$S = [\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_N] \quad (17)$$

The mean image $\bar{\mathbf{f}}$ of set S, is given by

$$\bar{\mathbf{f}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{f}_i \quad (18)$$

Subtracting $\bar{\mathbf{f}}$ from the original faces gives

$$\mathbf{a}_i = \mathbf{f}_i - \bar{\mathbf{f}},,, i = 1,2,...N \quad (19)$$

This gives another $M \times N$ matrix A:

$$A = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_N] \quad (20)$$

Since $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_r\}$ form an orthonormal basis for $R(A)$, the range (column) subspace of matrix A. Since matrix A is formed from a training set S with N face images, $R(A)$ is called a 'face subspace' in the 'image space' of $m \times n$ pixels, and each $\mathbf{u}_i$, $i = 1,2,...,r$, can be called a 'base-face'.

Let $\mathbf{x}$ $(= [x_1, x_2, ..., x_r]^T)$ be the coordinates (position) of any $m \times n$ face image f in the face subspace. Then it is the scalar projection of $\mathbf{f} - \bar{\mathbf{f}}$ onto the base-faces:

$$\mathbf{x} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r]^T(\mathbf{f} - \bar{\mathbf{f}}) \quad (21)$$

This coordinate vector x is used to find which of the training faces best describes the face f. That is to find some training face $\mathbf{f}_i$, $i = 1,2,...,N$, that minimizes the distance:

$$\varepsilon_i = \|\mathbf{x} - \mathbf{x}_i\|_2 = [(\mathbf{x} - \mathbf{x}_i)^T(\mathbf{x} - \mathbf{x}_i)]^{1/2} \quad (22)$$

where $\mathbf{x}_i$ is the coordinate vector of $\mathbf{f}_i$, which is the scalar projection of $\mathbf{f}_i - \bar{\mathbf{f}}$ onto the base-faces:

$$\mathbf{x}_i = [\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_r}]^T(\mathbf{f}_i - \bar{\mathbf{f}}) \quad (23)$$

A face $\mathbf{f}$ is classified as face $\mathbf{f}_i$ when the minimum ε_i is less than some predefined

threshold ε_0. Otherwise the face f is classified as "unknown face".
If **f** is <u>not</u> a face, its distance to the face subspace will be greater than 0. Since the vector projection of $\mathbf{f}-\bar{\mathbf{f}}$ onto the face space is given by

$$\mathbf{f}_p = [\mathbf{u}_1, \mathbf{u}_2, ...\mathbf{u}_r]\mathbf{x} \qquad (24)$$

where **x** is given in (21).

The distance of **f** to the face space is the distance between $\mathbf{f}-\bar{\mathbf{f}}$ and the projection $\mathbf{f}_p$ onto the face space:

$$\varepsilon_f = \left\|(\mathbf{f}-\bar{\mathbf{f}})-\mathbf{f}_p\right\|_2 = [(\mathbf{f}-\bar{\mathbf{f}}-\mathbf{f}_p)^T(\mathbf{f}-\bar{\mathbf{f}}-\mathbf{f}_p)]^{1/2} \qquad (25)$$

If ε_f is greater than some predefined threshold ε_1, then **f** is not a face image.

3.4 Steps to Conduct FR with SVD

The flowchart for face recognition with SVD is showed in the Figure2. The explanations of each step as following:

1. Obtain a training set *S* with *N* face images of known individuals.
2. Compute the mean face $\bar{\mathbf{f}}$ of *S* by (18)
3. Forms a matrix *A* in (20) with the computed $\bar{\mathbf{f}}$.
4. Calculate the SVD of *A* as shown in (1)
5. For each known individual, compute the coordinate vector $\mathbf{x}_i$ from (23). Choose a threshold ε_1 that defines the maximum allowable distance from face space. Determine a threshold ε_0 that defines the maximum allowable distance from any known face in the training set *S*.

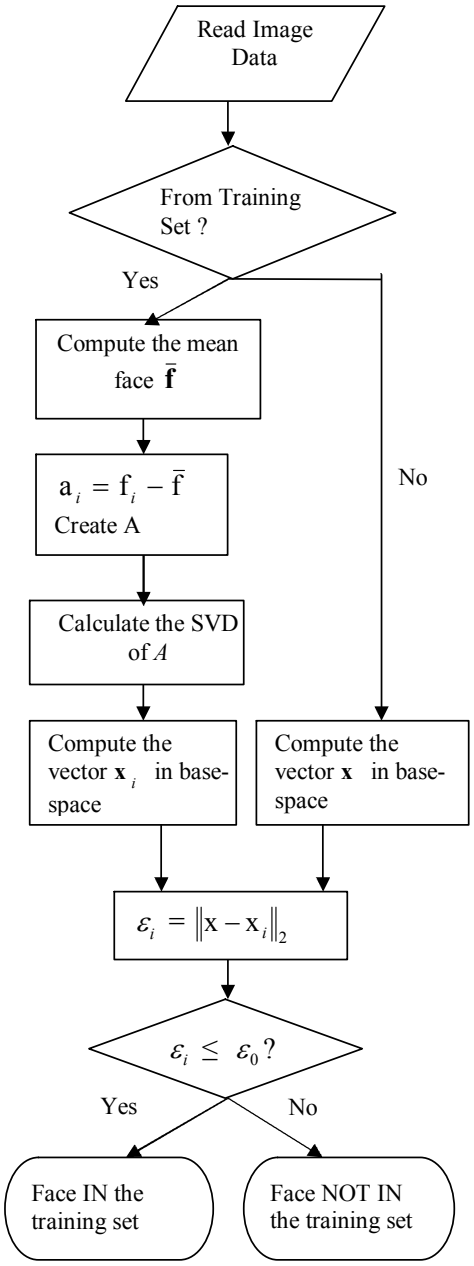


Figure2. Flow chart of Face Recognition with SVD

6. For a new input image f to be identified, calculate its coordinate vector **x** from (21), the vector projection $\mathbf{f}_p$, the

distance ε_f to the face space from (25). If $\varepsilon_f > \varepsilon_1$ the input image is not a face.

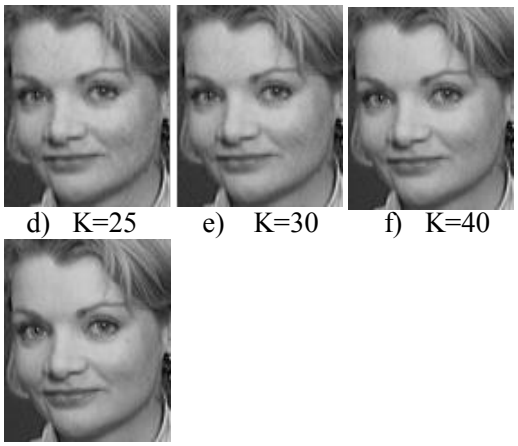
7. If $\varepsilon_f < \varepsilon_1$, compute the distance ε_i to each known individual. If all $\varepsilon_i > \varepsilon_0$, the input image may be classified as unknown face, and optionally used to begin a new individual face. If $\varepsilon_f < \varepsilon_1$, and some $\varepsilon_i < \varepsilon_0$, classify the input image as the known individual associated with the minimum ε_i ($\mathbf{x}_i$), and this image may optionally added to the original training set. Steps 1-5 may be repeated. This can update the system with more instances of known faces [1].

4. EXPERIMENTATIONS AND RESULTS

4.1 Result of Experimentations for Image Compression

Figure3 shows examples of images used for the system tests under different K terms. i) shows the original images whereas a) shows the results of the reconstruction image using 10 singular values, and c) shows the results using 20 values and so on. The observation on those examples, we found when k1 ≤ 20, the images are blurry and with the increase of singular values we have a better approach to the original image. Table1 shows a summary of the results obtained with the measure of the storage space, and the error measures for the tested images.

a) K=10 b) K=15 c) k=20



d) K=25 e) K=30 f) K=40

g) Original

Figure3. Examples of a Image Used to Test Image Compressing

Table1. Summery of the Result for Image Compression

K	Storage Space (bytes)	C_R Comp	MSE (Quality)
10	2050	5.03	108.11
15	3075	3.35	63.15
20	4100	2.51	40.39
25	5125	2.01	27.22
30	6150	1.68	15.64
40	8200	1.26	9.07
original	10304	1	

With the results from table1, we have couple of the observations:

1. Using less singular value (smaller K), the. better compression ratio is achieved
2. However, the more singular value is used (larger K), quality measurement *MSE* is smaller (better image quality), and the reconstructed images are more equal to the original, but using more storage space.
3. For the testing image, the acceptable image quality is about with k=25, and compression ratio is $C_R = 2.01$.
4. The image close to original image when k = 40. At this point $CR = 1.26$, and $MSE = 9.07$.

4.2 Results for Face Recognition with SVD

The test is under the training set with image Size: M = 92 x 112 = 10,304, the number of known individuals: N = 20, Different Conditions: All frontal and slight tilt of the head, different facial expressions.

Essentially, a face image is of M (say 10,000) dimension. But the rank r of matrix A is less than or equals N. For most applications, a smaller number of base-faces than r are sufficient for identification. In this way, the amount of computation is greatly reduced. The following figures show the base face image, the average of training set image, and the training set image we used for this experiment.

Figure4. Training Set Images

Figure5. A Computed Mean Face Image of Training Set Images

5 MY CONTRIBUTION TO THIS PROJECT

This part is a special section which emphasizes my contribution to this project. A summery of my contribution is list following:

1. tested and evaluated the digital image compression under different k terms by using SVD theory;
2. Recoded the program of face recognition by applying matrices operation in the MATLAB and reduced the lines of the code, tested it with 20 face images
3. Invested the characteristics of singular values and singular vectors in the image processing.

More detail explanations will be given below

5.1 Experiments on Characteristics of SVD

In digital image processing, image features are divided into four groups: visual features, statistical pixel features, transform coefficient features, and algebraic features. SVD technique can be considered as an algebraic feature [16]. The algebraic features usually represent intrinsic properties.

Refer to section2.2; the first property of SVD is that: the singular values $\sigma_1, \sigma_2, \cdots, \sigma_n$ are unique, but the matrices *U and V* are not unique

Dr.Zeng was interested in that the singular values are more important since its uniqueness; it is naturally thinking that SVs is the most important attribute of image matrix. It could be used for recognition.

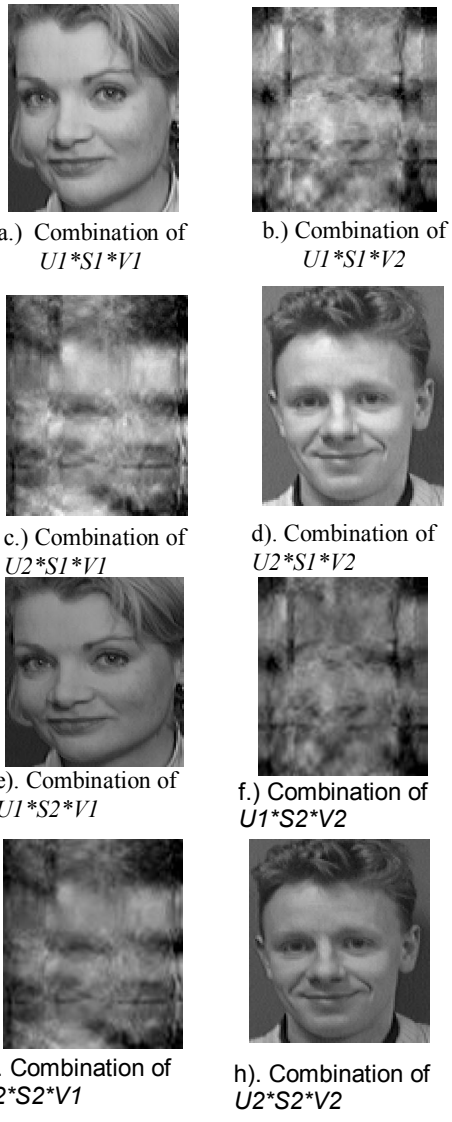
However, with experiments on exchange the SVs of two images, the result is very interesting and it shows that the singular-vectors (left and right) are more important for reconstruction of the original image.

The first experiment was designed to use two person's face images, on which we performed

SVD decomposition. For example, the face images of Janet (*A1*) and Andy (*A2*) was decomposed into *U,S,V* so that:

$$A1 = U_1 * S_1 * V_1^T$$
$$A2 = U_2 * S_2 * V_2^T$$

Then we did the combination of the singular values and singular-vectors, the result shows in the Figure 7.



a.) Combination of *U1*S1*V1*

b.) Combination of *U1*S1*V2*

c.) Combination of *U2*S1*V1*

d). Combination of *U2*S1*V2*

e). Combination of *U1*S2*V1*

f.) Combination of *U1*S2*V2*

g). Combination of *U2*S2*V1*

h). Combination of *U2*S2*V2*

Figure6. Result of Exchanged Singular Value with Singular Vectors

In figure6, a) shows the combination of $U_1 * S_1 * V_1^T$, which is an original image of Janet. However, when we combined Janet's SVs with Andy's singular-vector, it shows Andy's face image (see figure6 d)). The image has different brightness with Andy's original image. e) shows Janet's face, but it is a combination of Andy's SVs and Janet's singular-vectors.

When we combined two pair of singular vectors *U* and *V*, which are from two images respectively, the outcome images look like a "ghost". The result show in b), c), f) and g),

We also tested the two images, one is a face image and other is not a face image (e.g. flower). The experimentation showed the same result as two face images.

From the result we see that, though the singular values are unique in SVD decomposition, but the singular victors are more important for image recognition. This fact indicates that deeply research and further investigation on characteristics of SVD in image processing are necessary.

5.2 Coding With MATLAB

MATLAB is a numerical computing environment and allows easy matrix manipulation. Many built in functions are for image processing. It is used to test new image processing techniques and algorithms. Almost everything in MATLAB is done through programming and manipulation of raw image data [6]

MATLAB does include standard "for" and "while" loops, but using MATLAB's vectorized notation often produces code that is easier to read and faster to execute.

In this project, face recognition using SVD were coded with MATLAB. The programs utilize matrices operation to manipulate data to reduce "for" loops, which reduce lines of the

code. The source code is in Appendix, for comparison purpose, the old program for face recognition also is attached in Appendix.

Following is partial demonstration of using matrices operation in my program:

```
function [ef, d] = svdRecognition0(newName,
r, N, A, U, S, V, fbar, e0, e1)
%newName = 'janet1.tiff'; r = # of sv
chosen;

Ur = U(:, 1:r);
X = Ur'*A;
fnew = imread(newName);
fnew = imresize(fnew, [112, 92]);
f = reshape(fnew, 10304, 1);
f0 = double(f) - fbar;
x = Ur'*f0; fp = Ur*x;
ef = norm(f0 - fp);
if ef < e1
    D = X - x*ones(1, N);
    d = sqrt(diag(D'*D));
    [dmin, indx] = min(d);
    if dmin < e0
        fprintf(['This image is face
                #', num2str(indx)]);
    else
        fprintf('The input image is an
                unknown face');
    end
else
    fprintf('The input image is not a
            face');
end
```

The consideration for using matrix operations is that the inner matrix dimensions must agree, In the program above, U is a *MxM* right matrix of singular value decomposition of A *(M x N)*, Ur is *M x r* matrix that are form from *U* (r is the number of singular values we chosen). X=Ur'*A is the coordinates (position) matrix that is the *r x N* dimensions for training set A. **x** is a coordinates vector in the *r x N* subspace for testing image.

Here:

X= Ur'*A $= [\text{Ur'}*a_1, \text{Ur'}*a_2, \ldots, \text{Ur'}*a_N]$

$= [\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}]$

Where

$$\mathbf{x_1} = \begin{bmatrix} x_{11} \\ x_{12} \\ \ldots \\ x_{1r} \end{bmatrix}, \mathbf{x_2} = \begin{bmatrix} x_{21} \\ x_{22} \\ \ldots \\ x_{2r} \end{bmatrix}, \quad \mathbf{x_N} = \begin{bmatrix} x_{N1} \\ x_{N2} \\ \ldots \\ x_{Nr} \end{bmatrix}$$

and $$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_r \end{bmatrix}$$

When computing the distance of each points between the training set and testing image**,**

D = X - **x***ones(1, N)

In order to have agreed dimensions to operate the matrix X and vector **x** , we need to transfer the vector **x** to the matrix form by operation x*ones(1, N),

$$\mathbf{x}*\text{ones}(1, N) = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_r \end{bmatrix} [1,1, \ldots 1]$$

$$= \begin{bmatrix} x_1 & x_1 & \ldots & x_1 \\ x_2 & x_2 & \ldots & x_2 \\ \ldots & \ldots & \ldots & \ldots \\ x_r & x_r & \ldots & x_r \end{bmatrix}$$

(r x N)

So that the coordinates distance matrix between the testing image and each image in the training set

$$D = [\mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_N}]$$

Where

$$\mathbf{D_1} = \begin{bmatrix} x_{11} - x1 \\ x_{12} - x_2 \\ \ldots \\ x_{1r} - x_r \end{bmatrix} \quad \ldots \quad \mathbf{D_N} = \begin{bmatrix} x_{N1} - x1 \\ x_{N2} - x_2 \\ \ldots \\ x_{Nr} - x_r \end{bmatrix}$$

The minimized distance between training face and testing image:

$$d = \text{sqrt}(\text{diag}(D'*D))$$

Where

$$D'*D = \begin{bmatrix} D_1' \\ D_2' \\ \dots \\ D_N' \end{bmatrix} [D_1, \quad D_2, \quad \dots, \quad D_N]$$

$$= \begin{bmatrix} D_1'D_1 & & & D_1'D_N \\ \dots & D_2'D_2 & & \\ \dots & & \dots & \\ D_N'D_1 & & & D_N'D_N \end{bmatrix}$$

$$\text{diag}(D'*D) = [D_1'D_1, \quad D_2'D_2, \quad \dots \quad D_N'D_N]$$

Therefore

$$d = \text{sqrt}(\text{diag}(D'*D))$$

is a vector of the minimized distance between training face images and testing image.

The code for image compression and computing the MSE for image compression are also in the Appendix.

6. CONC LUSION AND FUTURE WORK

This project has applied technique of linear algebra "singular value decomposition (SVD)" to digital image processing. Two specific areas of image processing are investigated and tested.

Basis on the theory and result of experiments, we found that SVD is a stable and effective method to split the system into a set of linearly independent components, each of them is carrying own data (information) to contribute to the system, Thus, both rank of the problem and subspace orientation can be determined.

SVD has the advantage of providing a good compression ratio, and that can be well adapted to the statistical variation of the image; but it has the disadvantage that it is not fast from the computational point of view, and the problem of which its application is strongly conditional due to the excessive work of associate calculations.

The result obtained for image compressing has satisfactory of image compression ratio compare with image quality; the results for face recognition with a small error percentage compare recognition using the original image dimensions. The face recognition test performed using the image that project into face–base show that it is necessary to improve the algorithm to work with complex objects.

Some of images are simple so that only needs a few singular values to obtain the approximation, and the complex parts needs to use more values to maintain their quality. We can conclude that the image does not require a same k in its totality.

Overall, The SVD approach is robust, simple, easy and fast to implement. It works well in a constrained environment. It provides a practical solution to image compression and recognition problem. Instead of searching a large database of faces, by using base-faces, this small set of likely matches for given images can be easily obtained.

Future work consists three aspects, one is to work on more complex image such as vary large size 2-D image or 3-D images with SVD technique for image compression and recognition; second, deeply to study and investigate the roles of singular values and singular vectors in image processing. Third, this application is completed on the MATLAB, in the future; the application can be performed with programming of Java or C++, so that the real-time image processing can be achieved.

REFERENCES

[1] Guoliang Zeng, "Face Recognition with Singular Value Decomposition.", CISSE *Proceeding*, 2006

[2] Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins, "Digital Image Processing Using MatLab", Prentice Hall, 2006

[3] Bernd Jahne, "Digital Image Procession", Springer, 2002

[4] Steve J. Leon; "Linear Algebra with Applications", Macmillan Publishing Company, New York; 1996

[5] S.G. Kong, J. Heo, B.R. Abidi, J. Paik, and M. A. Abidi. Recent advances in visual and infrared face recognition-a review. *Computer Vision and Image Understanding*, 97(1):103{135, 2005.

[6] Wickpidia,. Digital Image processing, available at http://en.wikipedia.org/wiki/Digital_image_processing

[7] L. Wiskott, J. Fellous, N. Kruger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 19(7):775{779, 1997.

[8] Mande M., Singular Value Decomposition, Department of computer science of the Technological Institute of Bombay India, august 2003.

[9] A. M. Martinez and A.C. Kak. Pca versus lda. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(2):228{233, 2001

[10] Mrak M., Grgic S. and Grgic M., Picture Quality Measures in image compression systems, IEEE EUROCON, Ljubljana, Eslovenia, September 2003.

[11] S.G. Kong, J. Heo, B.R. Abidi, J. Paik, and M. A. Abidi. Recent advances in visual and infrared face recognition-a review. *Computer Vision and Image Understanding*, 97(1):103{135, 2005.

[12] T. Kanade, "Picture Processing System by Computer Complex and Recognition of Human Faces," Department of Information Science, Kyoto University, Nov.1973.

[13] A. L. Yuille, D. S. Cohen, and P. W. Hallinan, "Feature Extraction from Faces Using Deformable Templates," *Proc. CVPR*, San Diego, CA June 1989.

[14] Matthew A. Turk, Pentland P. Alex(1991), "Face Recognition using Eigenface method", *IEEE Conference on Computer Vision and Pattern Recognition,* pp.586-591, 1991.

Appendix

Partial Source code:

1. code for face recognition

```
function [ef, d] =
svdRecognition0(newName, r, N, A, U, S,
V, fbar, e0, e1)
%newName = 'janet1.tiff'; r = # of sv
choosed;

Ur = U(:, 1:r); X = Ur'*A;
fnew = imread(newName);
fnew = imresize(fnew, [112, 92]);
f = reshape(fnew, 10304, 1);
f0 = double(f) - fbar;
x = Ur'*f0;
fp = Ur*x;
ef = norm(f0 - fp);
if ef < e1
    D = X - x*ones(1, N); d =
sqrt(diag(D'*D));[dmin, indx] = min(d);
    if dmin < e0
        fprintf(['This image is face
#', num2str(indx)]);
    else
```

```
        fprintf('The input image is an
unknown face');
    end
else
    fprintf('The input image is not a
face');
end

function [A, U, S, V, fbar,basefi] =
svdDecomp(fileName, N)
%fileName = 'imageset.txt';
fid = fopen(fileName);
S = zeros(10304, N);
    for i=1:N
        face = fgetl(fid);
        fi = imread(face);

subplot(ceil(sqrt(N)),ceil(sqrt(N)),i);
        fprintf(1,'%s.\n',face);
        figure(1); imshow(fi);

        fi =
double(reshape(fi,10304,1));
        S(:,i) = fi;
    end
fbar = (mean(S'))';
figure(2); imshow(reshape(uint8(fbar),
112, 92));
A = S - fbar*ones(1, N);
[U, S, V] = svd(A, 0);
```

2. Code for Exchange Singular Value and Singular Vectors

```
function
[A1,A2]=SVDExchange(Image1,Image2)
A1=imresize(imread(Image1),[112,92]);
A2=imresize(imread(Image2),[112,92]);;

   s1 = size(A1); % find out the size
of the image
   s2 = size(A2);
   ss1 =size(s1);
   ss2 = size(s2);
         if ss1(:,2)== 3; % if the
image is a color image in jpeg or jog
formatn it will covert to the grey
scale
                A1 = rgb2gray(A1);
          end
          if ss2(:,2)== 3; % if the
image is a color image in jpeg or jog
formatn it will covert to the grey
scale
                A2 = rgb2gray(A2);
         end
```

```
[u1,s1,v1]=svd(double(A1));
[u2,s2,v2]=svd(double(A2));
combinf1=uint8(u1*s1*v1');
figure(1); imshow(combinf1);
 title('combination of u1*s1*v1');
combinf2=uint8(u1*s1*v2');
figure(2); imshow(combinf2);
title('combination of u1*s1*v2');
combinf3=uint8(u2*s1*v1');
figure(3); imshow(combinf3);
title('combination of u2*s1*v1');
combinf4=uint8(u2*s1*v2');
figure(4); imshow(combinf4);
title('combination of u2*s1*v2');
combinf5=uint8(u1*s2*v1');
figure(5); imshow(combinf5);
title('combination of u1*s2*v1');
combinf6=uint8(u1*s2*v2');
figure(6); imshow(combinf6);
title('combination of u1*s2*v2');
combinf7=uint8(u2*s2*v1');
figure(7); imshow(combinf7);
title('combination of u2*s2*v1');
combinf8=uint8(u2*s2*v2');
figure(8); imshow(combinf8);
title('combination of u2*s2*v2');
```

3. Code for Image Compression and MSE

```
function AK = svdPartSum(A,K)
A=double(A);
[u,s,v]=svd(A);
AK=u(:,1:K)*s(1:K,1:K)*v(:,1:K)';
AK=uint8(AK);
imshow(AK);

function [MSE,MSEA,m,n] =ComputMse (A,
AK)
 m=size(A,1);
 n=size(A,2);
 e=0.0;
 MSEA = 0.0;
 A=double(A);
 AK=double(AK);
for  i= 1:m
     for j=1:n
         e = (A(i,j)-AK(i,j))^2;
         MSEA =MSEA +e;
     end
end
MSE=MSEA/(m*n);
```

4 Old Program for Face Recognition

```
clear;
clc;
```

```
Filename = 'trainingset.txt';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%vv
%this function getrs the training
faces which is stored in a text file
% and store them in an array 'face'%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

num =25;
fid = fopen(Filename); %open a file
images = zeros(10304, num);
    for i = 1:num
        face = fgetl(fid);
        readface = imread(face); %
read image from graphic file and
return image data in array
        s = size(readface); % find out
the size of the image arrey
        ss =size(s);
          if ss(:,2)== 3; % if the
image is a color image in jpeg or jog
formatn it will covert to the grey
scale
                readface =
rgb2gray(readface);
          end

         readface = imresize(readface,
[112,92]);
         readface
=(reshape(readface,10304,1)); %
reshape the 92*112 vector as a single
column of 10304
         images(:,i) = readface; %
faces contain "num"   number of images
    end
         faces = images;
         S = size(faces,2); %Determin
the number of faces selected,2 is dem
of the faces , returns the size of the
dimension of face in dim2
       %psi = (mean(faces'))'; %
deturmine the avg face of all the
training faces
        psi = mean(faces,2);
        for i=1:S;
           phi(:,i) = faces(:,i)-psi;
% substruct each of  the training
faces from avg face
        end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
       %[U,S,V] = svd(X,'econ') also
produces the "economy size"
decomposition.
       %If X is m-by-n with m >= n, it
is equivalent to svd(X,0).
       %For m < n, only the first m
columns of V are computed and S is m-
by-m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
       [u sig vt] = svd(phi,'econ');
       %[u sig vt] = svd(phi);
        si = svd(sig);
       tol =
max(size(sig))*eps(max(si))
       r = sum(si > tol);
      %  r = rank(sig);
     %    tol =
max(size(A))*eps(max(s));
     %      r = sum(s > tol);
       [row col] = size (sig);
       new_u =u(:,1:r);
     fclose(fid);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
       % project of image onto the
face space

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
      fid = fopen(Filename);
       images =zeros(10304,num);
     for i=1:num
              face =fgetl(fid);
              readface = imread(face);
              s = size(readface);
              ss =size(s);
        if ss(:,2)== 3; % if the image
is a color image in jpeg or jog
formatn it will covert to the grey
scale
                 readface =
rgb2gray(readface);
        end

readface=imresize(readface,[112,92]);
            readface =
double(reshape(readface,10304,1));
            proj = new_u'*(readface-
psi);
           proj_matrix(:,i) = proj';
     end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
%Test the image
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%

          test = imread('test3.tif');
          TestFace =
imresize(test,[112 92]);
          TestFace =
double(reshape(TestFace,10304,1));
          ProjTestFace = new_u'
*(TestFace-psi);

      for i = 1:num
              TestMatrix(:,i) =
proj_matrix(:,i) -ProjTestFace;

              DistanceMatrix(:,i) =
sqrtm(TestMatrix(:,i)' *
TestMatrix(:,i));
      end
             [row col] =
size(DistanceMatrix);
      for   i = 1:row
              evalMatrix(1,i) =
DistanceMatrix(i,i);
      end

              [mi index] =
min(evalMatrix) %Minimum elements of
an array
              fid =fopen(Filename);
      for i = 1:index
             face =fgetl(fid);
      end

      figure(3)
       subplot(2,2,1)
       imshow(test)
       title('Input Image', 'fontsize',
10);
       subplot (2,2,2)
       imshow(face);
       title('Retrived
Image','fontsize',10);
       i = 1: size(evalMatrix,2);
       subplot(2,2,3);
       stem(i,evalMatrix);

       function faces =
getfaces(Filename, num)
       fid = fopen(Filename);
       images = zeros(10304, num);
       for i = 1:num
        face = fgetl(fid);
        readface = imread(face); %
read image from graphic file and
return image data in array

subplot(ceil(sqrt(num)),ceil(sqrt(num)
),1)
         fprintf(1,'%s.\n',face);

       readface =
(reshape(readface,10304,1));
       figure =readface;
       end
       face =images;

          readface =
double(reshape(readface,10304,1));
```