# Image Analysis - FMAN20
# Assignment 1

## Histogram Equalization
## Linear Algebra for Image
## Optical Recognition System

Hicham Mohamad
hsmo@kth.se

October 11, 2018

## 1 Introduction

In this assignment we study basic *histogram equalization*, *linear algebra for images* and start our work on our own *Optical Character Recognition system*. The data for the assignments is on the course homepage: `http://www.ctr.maths.lu.se/course/newimagean/2018/`

## 2 Image Sampling

In order for computers to process an image, this image has to be described as a series of numbers, each of finite precision. This calls for two kinds of discretisation:

- Sampling, and
- Quantisation

By sampling is meant that the brightness information is only stored at a discrete number of locations. Quantisation indicates the discretisation of the brightness levels at these positions.
The main task in this problem is to sample the image evenly to a discrete image with 5x5 pixels. Then we need to quantify the discrete image with 16 different gray levels from 0 to 15. Consequently, we need first to discretise the axes x and y by normalizing by 4 so we get the following set of values for x and y:

$$0, 0.25, 0.5, 0.75, 1$$

. Then we calculate the samples from the points (x,y) by using the following given function

$$f(x, y) = 4x(1 - x)$$

$$\begin{bmatrix} f(0,1) & f(0.25,1) & f(0.5,1) & f(0.75,1) & f(1,1) \\ f(0,0.75) & f(0.25,0.75) & f(0.5,0.75) & f(0.75,0.75) & f(1,0.75) \\ f(0,0.5) & f(0.25,0.5) & f(0.5,0.5) & f(0.75,0.5) & f(1,0.5) \\ f(0,0.25) & f(0.25,0.25) & f(0.5,0.25) & f(075,0.25) & f(1,0.25) \\ f(0,0) & f(0.25,0) & f(0.5,0) & f(0.75,0) & f(1,0) \end{bmatrix} = \begin{bmatrix} 0 & 0.75 & 1 & 0.75 & 0 \\ 0 & 0.75 & 1 & 0.75 & 0 \\ 0 & 0.75 & 1 & 0.75 & 0 \\ 0 & 0.75 & 1 & 0.75 & 0 \\ 0 & 0.75 & 1 & 0.75 & 0 \end{bmatrix}$$

For quantisation we need to discretize the obtained brightness levels by multiplying them by 15 and rounding the results to integers for getting 16 gray levels from 0 to 15. In this way we get the following quantified image with 5x5 pixels

$$\begin{bmatrix} 0 & 11 & 15 & 11 & 0 \\ 0 & 11 & 15 & 11 & 0 \\ 0 & 11 & 15 & 11 & 0 \\ 0 & 11 & 15 & 11 & 0 \\ 0 & 11 & 15 & 11 & 0 \end{bmatrix}$$

# 3 Histogram equalization

Here is another issue of interest, after that one of image sampling, we need to address the histogram equalization. Let $s = T(r)$ be a gray level transformation. It follows that

$$\int_0^s p_s(t)dt = \int_0^r p_r(t)dt$$

Where $p_r$ is the histogram before the transformation and $p_s$ is that one after the transformation. we take $T$ so that the resulting histogram $P_s = 1$

$$\int_0^r p_r(t)dt = \int_0^s 1dt = s \implies s = T(r) = \int_0^r p_r(t)dt = \int_0^r \frac{e^t - 1}{e - 2}dt = \frac{1}{e-2}(e^r - r - 1)$$

This transformation is what we call histogram equalization.

# 4 Neighbourhood of pixels

$$\begin{pmatrix}
3 & 3 & 2 & 2 & 2 & 3 & 3 & 3 & \textcircled{0} & 2 & 2 & 2 \\
\textcircled{0} & 3 & 2 & \textcircled{0} & \textcircled{0} & \textcircled{1} & \textcircled{0} & \textcircled{0} & \textcircled{0} & 3 & 2 & 3 \\
\textcircled{1} & 2 & \textcircled{0} & \textcircled{1} & \textcircled{1} & \textcircled{1} & \textcircled{0} & \textcircled{1} & \textcircled{1} & \textcircled{0} & 2 & 3 \\
3 & \textcircled{0} & 3 & \textcircled{1} & \textcircled{0} & 3 & 3 & \textcircled{0} & 2 & 3 & 3 & 2 \\
3 & 2 & \textcircled{1} & \textcircled{0} & \textcircled{1} & 3 & \textcircled{1} & 3 & 2 & \textcircled{0} & 3 & 2 \\
\textcircled{0} & 3 & 3 & \textcircled{1} & \textcircled{1} & \textcircled{1} & \textcircled{1} & \textcircled{0} & 2 & 3 & 3 & 3 \\
2 & \textcircled{0} & 2 & \textcircled{1} & \textcircled{1} & \textcircled{0} & \textcircled{1} & \textcircled{0} & 2 & 2 & \textcircled{0} & 2 \\
2 & \textcircled{1} & 3 & \textcircled{0} & \textcircled{0} & 3 & 2 & \textcircled{1} & 2 & \textcircled{0} & 3 & 2 \\
2 & \textcircled{0} & 3 & \textcircled{1} & \textcircled{0} & 3 & 2 & 3 & 2 & 2 & \textcircled{0} & 3 \\
2 & 2 & 2 & \textcircled{1} & \textcircled{0} & \textcircled{1} & \textcircled{0} & \textcircled{0} & \textcircled{1} & \textcircled{0} & 2 & \textcircled{1} \\
\textcircled{0} & 2 & 3 & \textcircled{0} & \textcircled{0} & \textcircled{1} & \textcircled{0} & \textcircled{0} & \textcircled{0} & 3 & 3 & 2 \\
\textcircled{0} & 2 & 3 & \textcircled{0} & 3 & 3 & 2 & \textcircled{1} & 2 & \textcircled{1} & \textcircled{0} & 2
\end{pmatrix}$$

*Figure 1: Image where elements with intensity 0 or 1 are marked with a circle*

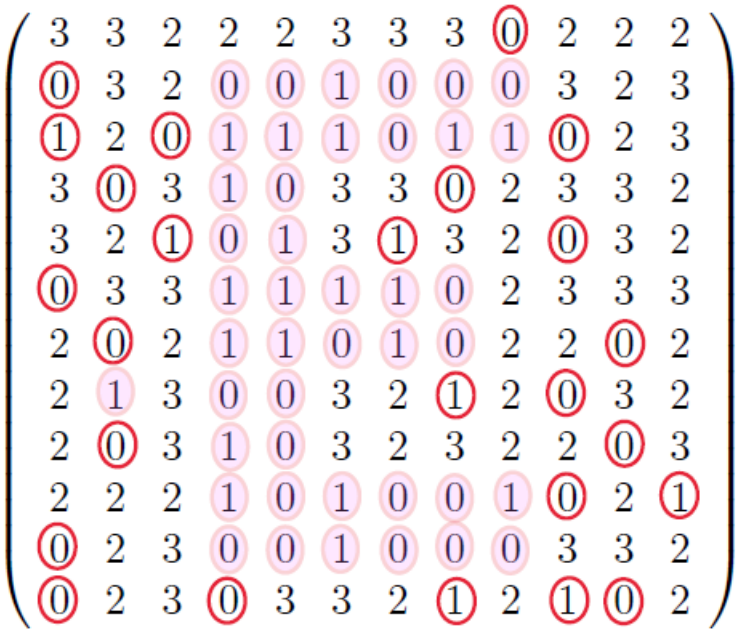


*Figure 2: Image where elements with intensity 0 or 1 and has at least two 4-neighbours, that also have intensities 0 or 1, are marked with a filled circle*

## 5 Segmentation part of OCR

For building and testing a small system for ocr (optical character recognition), we need first here in this assignment to write a function `S = im2segment(image)` in matlab. im2segment() takes such an image matrix I as input and returns a segmentation, i.e. a set of images S = (S1,...,Sn) one for each letter in the image. To solve this problem, we use an effiecient algorithm represented by the matlab function `bwlabel()` for dividing the matrix in connected components and creating a label image, where all pixels having the same value belong to the same object, i.e letter. For example

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 0 & 2 & 2 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 3 & 0 \\ 0 & 0 & 1 & 3 & 3 & 0 \\ 4 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

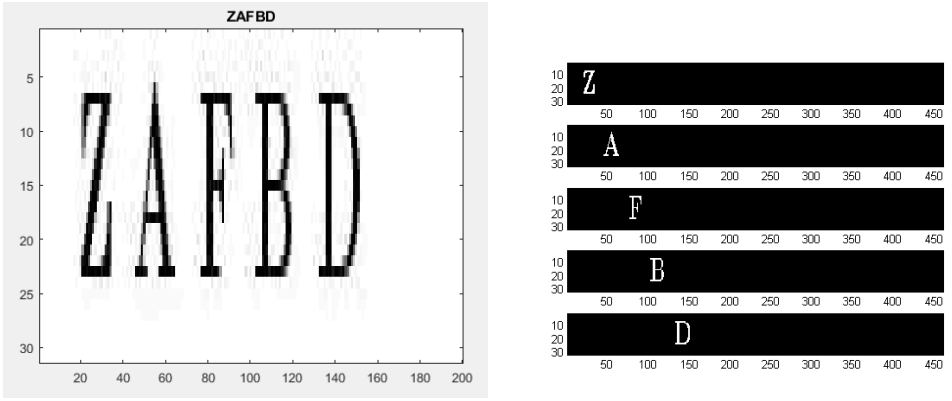
As output to `bwlabel`, we get a matrix containing labels for the connected components in the image. The elements of labels are integer values greater than or equal to 0. The pixels labeled 0 are the background. The pixels labeled 1 make up one object, the pixels labeled 2 make up a second object, and so on.

*Listing 1: Matlab function that returns a segmentation of an image.*

```
function [S] = myim2segment(im)

% returns the number of rows and columns
m = size(im,1);
n = size(im,2);

% thresholding of the intensity (brightness)to segment the foreground
im = im < 125;

% the output must be a cell array
S = cell(1,5);

% create a label image
labels = bwlabel(im,8);

```

```
for kk = 1:5;
    %S{kk}= (rand(m,n)<0.5);

    [r,c] = find(labels == kk);    % Find indices and values of nonzero
        elements
    rc = [r c];
    [sx sy] = size(rc);
    blackBg = zeros(m,n);   % each image matrix Si should be with 1 at the
                            % pixels for that letter and 0 for all other.

    for i = 1:sx
        x = rc(i,1);
        y = rc(i,2);
        blackBg(x,y) = im(x,y);
    end

    S{kk} = blackBg;

end;
```

When running my segmentation function on the input image shown in Figure 3a, the resulting output cell array of the segmented images is shown in Figure 3b.




(a) (b)

*Figure 3: a) The original input image. b) The output of the segmented images after running the segmentation function.*

Then for testing my segmentation function we use also the benchmark script, that loads each image in a folder and each ground truth segmentation and at the end measures the it works well, as following below. An output of the resulting test is illustrated in Figure 4.

*Listing 2: The results after using the benchmark script.*

```
You tested 10 images in folder ../datasets/short1
The jaccard scores for all segments in all images were
    1.0000    0.9462    0.9889    0.9683    0.9750
    0.9796    0.9615    0.9758    0.9808    1.0000
    0.9574    0.9587    0.9717    0.9271    0.9574
    0.9545    0.9888    0.9881    0.9921    0.9841
    0.9796    1.0000    0.9692    0.9674    0.9870
    0.9789    0.9681    1.0000    0.9805    0.9906
    0.9870    1.0000    0.9615    0.9921    0.9529
    0.9545    0.9851    0.9766    0.9630    0.9720
    1.0000    1.0000    0.9695    0.9688    0.9538
    0.9574    0.9841    0.9175    0.9600    0.9789

The mean of the jaccard scores were 0.97424
This is great!
```

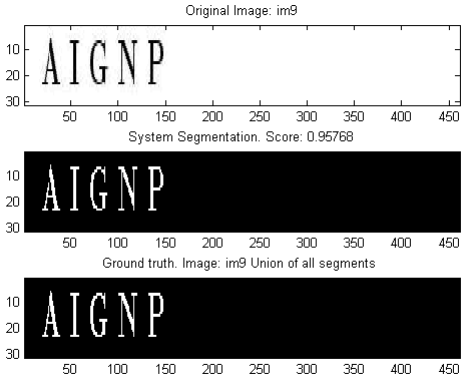



*Figure 4: The output after running the benchmark script*

# 6 Dimensionality

For finite dimensional vector spaces one may choose a basis of elements $e_1, ..., e_k$ so that every example u can be written as

$$u = x_1 e_1 + ... + x_k e_k$$

where $k$ is the dimension of the vector space. In general, $e_1, ...e_n \in R^k$ is a basis in $R^k$ if they are linearly independent and they span $R^k$.

A: The vector space formed by the set of gray-scale (monochrome) images with 2x2 pixels is of dimension 2x2 = 4. One possible basis is the following, which is called `canonical basis`

$$e_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad e_2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad e_3 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \quad e_4 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

In this way, an image u can be written as vector representation through the use of coordinates

$$u = u_1 e_1 + u_2 e_2 + u_3 e_3 + u_4 e_4$$

That is why image matrices can be seen as vectors in a linear space.

B: The vector space formed by the set of gray-scale images with 2000x3000 pixels is of dimension 6000000. One possible basis can be also canonical basis of size 6000000

$$e(i,j) = \begin{bmatrix} 0 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & 1 & \vdots & \vdots \\ 0 & \dots & 0 & \dots & 0 \end{bmatrix}$$

with 1 at the position (i,j). Using the canonical basis image u can be written as

$$u = \sum_{ij} u(i,j) e(i,j)$$

It is also possible to choose another basis that is more suitable in some sense.

# 7 Scalar products and norm on images

The scalar product is defined by

$$u \cdot v = \sum_{i}^{2} \sum_{j}^{2} u(i,j) v(i,j)$$

The norm is defined by

$$\|u\| = \sqrt{u \cdot u} = \sqrt{\sum_{i}^{2} \sum_{j}^{2} u(i,j) u(i,j)}$$

In our problem we can calculate $\|u\|$, $\|u\|$, $\|u\|$ in the same way using the Matlab function `norm(u,'fro')` which calculates the Frobenius norm

$\|u\|$ = norm(u,'fro')= 5.1962
$\|v\|$ = norm(v,'fro')= 1
$\|w\|$ = norm(w,'fro')= 1

For getting the scalar product of matrices we need to multiply the matrices `element-wise`, i.e. Frobenius product of matrices. Here also we can use Matlab by multiplying using operator ".∗" . In this way we get the following results:

$u \cdot v = sum(sum(u. * v)) = -2.5$

$u \cdot w = sum(sum(u. * w)) = 4.5$

$v \cdot w = 0$

Two vectors $e_i$ and $e_j$ are orthonormal if

$$e_i \cdot e_j = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases}$$

That is, if they are all unit vectors and orthogonal to each other. So in our case, we can say that the matrices $v$ and $w$ are orthonormal, because we have seen that $\|v\| = 1$ and $\|w\| = 1$. In addition, we have seen that $v \cdot w = 0$.

**Orthogonal projection of $u$:** According to the theorem in the text book and lecture 3, if $a_1, ..., a_k$ are orthonormal then the projection of $u$ on $\pi$ is given by

$$u_\pi = \sum_i^k x_i a_i = x_1 a_1 + .. + x_k a_k$$

where the coordinates x are given by

$$x_i = a_i^* u = a_i \cdot u$$

This orthogonal projection is characterized by

1. $u_\pi \in \pi \quad$ where $\quad \pi$ is spanned by $a_1, ..., a_k$.
2. $u - u_\pi \perp W \quad$ for every $\quad W \in \pi$

In our problem, what is the orthogonal projection of u given by

$$u = \begin{bmatrix} 1 & -3 \\ 4 & -1 \end{bmatrix}$$

onto the subspace spanned by $v, w$ where

$$v = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}, \quad w = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$$

Since $v, w$ are orthogonale, as shown before, the coordinates are

$$x_1 = u \cdot v = -2.5 \quad \text{and} \quad x_2 = u \cdot w = 4.5$$

The orthogonal projection is then

$$u_\pi = x_1 v + x_2 w = -2.5v + 4.5w = \begin{bmatrix} 1 & -3.5 \\ 3.5 & -1 \end{bmatrix}$$

# 8 Image compression

The last stage in a camera's processing pipeline is usually some form of image compression. Here in this task we are dealing with low resolution images with $3 \times 4$ pixels. Before transmitting to a computer, one

would like to compress the images consisting of 12 intensities to 4 numbers. Using `PCA, i.e. Principal Component Analysis`, it is shown that the following 4 images can represent typical images well

$$\phi_1 = 1/3\begin{bmatrix}0 & 1 & 0\\ 1 & 1 & 1\\ 1 & 0 & 1\\ 1 & 1 & 1\end{bmatrix}, \quad \phi_2 = 1/3\begin{bmatrix}1 & 1 & 1\\ 1 & 0 & 1\\ -1 & -1 & -1\\ 0 & -1 & 0\end{bmatrix}, \quad \phi_3 = 1/2\begin{bmatrix}1 & 0 & -1\\ 1 & 0 & -1\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}, \quad \phi_4 = 1/2\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 1 & 0 & -1\\ 1 & 0 & -1\end{bmatrix}.$$

The first problem we are dealing with in this part is to show that these four images are orthonormal in the scalar product

$$f \cdot g = \sum_i \sum_j \bar{f}(i,j)g(i,j)$$

Actually these four images are orthonormal since we have that for i,j = 1,..,4

$$\phi_i \cdot \phi_j = \begin{cases}1 & \text{if} \quad i = j\\ 0 & \text{if} \quad i \neq j\end{cases}$$

That is, they are all unit vectors and orthogonal to each other. To show this, the norm and the scalar product are calculated using Matlab and it can be seen that $\|\phi_i\| = 1$ and that $\phi_i \cdot \phi_j = 0$ for all i,j = 1,..,4, as the following The norm is defined by

$$\|\phi_i\| = \sqrt{\phi_i \cdot \phi_i} = \sqrt{\sum_i^3 \sum_j^4 \phi(i,j)\phi(i,j)}$$

In Matlab, this is written as

$\|\phi_i\| = norm(\phi_i, 'fro') = 1$

And using Frobenius inner product, i.e. element-wise product of two matrices, as done in the section before

$\phi_i \cdot \phi_j = sum(sum(\phi_i .* \phi_j)) = 0$

**Finding the coordinates of $f_a$:**
In the second part of this task we should determine the 4 coordinates $x_1, x_2, x_3, x_4$ such that the approximate image $f_a$ is as close to f as possible, i.e. such that the $MSE = |f - f_a|^2$, i.e. `the mean square error`, is as small as possible. where

$$f_a = \sum_i^4 x_i\phi_i = x_1\phi_1 + x_2\phi_2 + x_3\phi_3 + x_4\phi_4$$

In linear algebra, we know that if $\{a_1, ..., a_k\}$ are orthonormal then the projection of $u$ on $\pi$ is given by

$$u_\pi = \sum_i^k y_i a_i = y_1 a_1 + .. + y_k a_k$$

where the coordinates $y_i$ are given by

$$y_i = a_i^* u = a_i \cdot u$$

This can be illustrated in Figure 5. Here in our problem, we know that $\{\phi_1, \phi_2, \phi_3, \phi_4\}$ are orthonormal then the projection of f is given by

$$f_a = \sum_i^4 x_i\phi_i = x_1\phi_1 + x_2\phi_2 + x_3\phi_3 + x_4\phi_4 \implies x_i = \phi_i^* f = \phi_i \cdot f$$

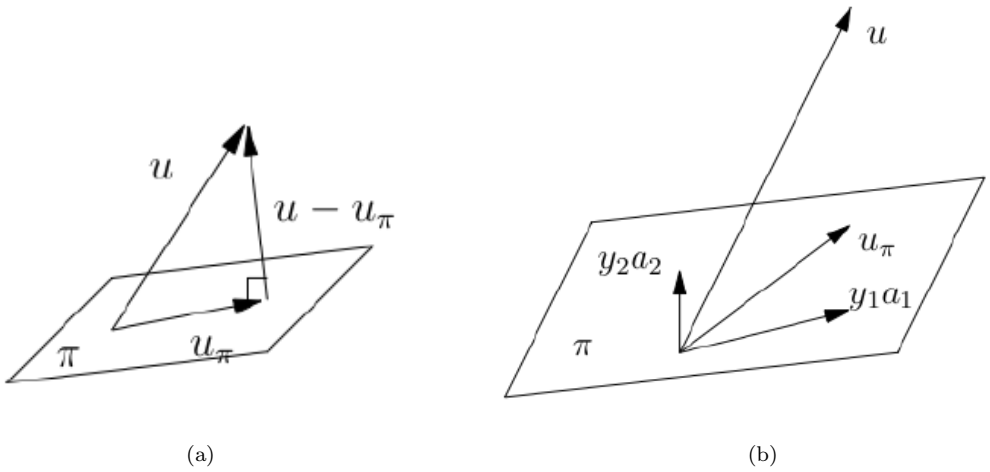
Using Matlab these coordinates can be calculated and the result is the following

$$x = \begin{bmatrix}17\\ 1.66667\\ 1.5\\ -4\end{bmatrix}$$

So the problem, in other words, is to find the coordinates of the orthogonal projection of the image f given by

$$f = \begin{bmatrix} -2 & 6 & 3 \\ 13 & 7 & 5 \\ 7 & 1 & 8 \\ -3 & 4 & 4 \end{bmatrix}$$

onto the subspace spanned by $\{\phi_1, \phi_2, \phi_3, \phi_4\}$.




(a) (b)

*Figure 5: a) Orthogonal projection of u on π, which is characterised by $u_\pi \in \pi$ and $u - u_\pi \perp W$ for every $W \in \pi$. b) Decomposition of the image u in coordinates, where the projection of u on π is given by $u_\pi = y_1 a_1 + .. + y_k a_k$.*

The approximate image, i.e. the orthogonal projection is then

$$f_a = 17\phi_1 + 1.66667\phi_2 + 1.5\phi_3 - 4\phi_4 = \begin{bmatrix} 1.3056 & 6.2222 & -0.1944 \\ 6.9722 & 5.6667 & 5.4722 \\ 3.1111 & -0.5556 & 7.1111 \\ 3.6667 & 5.1111 & 7.6667 \end{bmatrix}$$

# 9 Image bases

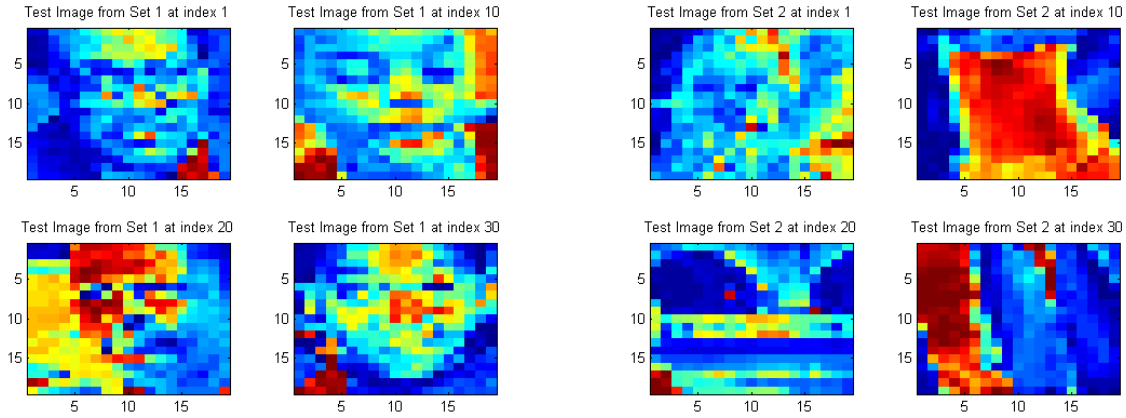
In this task we need to write a matlab function that projects an image u onto a basis $(e_1, e_2, e_3, e_4)$ and returns the projection up and error norm r, i.e. the norm of the difference $r = \|u - up\|$.

*Listing 3: Matlab function for projection that projects an image u onto a basis.*

```
1  function [up, r] = project(u, e)
2
3  % set of scalars (x1,x2,x3,x4) initialized to zeros
4  x = [0 0 0 0];
5  for i = 1 : 4
6      % The variable bases is a cell array.
7      % It contains three sets of bases for 3 different subspaces of
8      % dimension 4. The first basis is stored in a variable bases{1},
9      % which is a tensor of size 19X19X4. Thus, the 4 basis images are
10     % bases{1}(:,:,1), bases{1}(:,:,2), bases{1}(:,:,3), bases{1}(:,:,4).
11     basis = e(:,:,i);
12     x(i) = u(:)'* basis(:);
13 end
14
15 % The projection of an image u onto a basis (e1,e2,e3,e4) can be written as
16 % u = x1.e1 + x2.e2 + x3.e3 + x4.e4
17 up = x(1)*e(:,:,1) + x(2)*e(:,:,2) + x(3)*e(:,:,3) + x(4)*e(:,:,4);
18
```

```
% The error norm r = |u - up|
r = norm(u-up, 'fro');

end
```

Plots of images chosen at index 1, 10, 20, 30 from both stack 1 and stack 2 are shown in Figure 6. It seems that the images look like faces but those in stack 1 look better than images in stack 2.




(a) (b)

*Figure 6: a) Plot of images in test set 1. b) Plot of images in test set 2.*

Then plot for the four images in each of the three bases are shown in Figure 7. We can see that images from the first set of bases look like faces but the other are not clear.

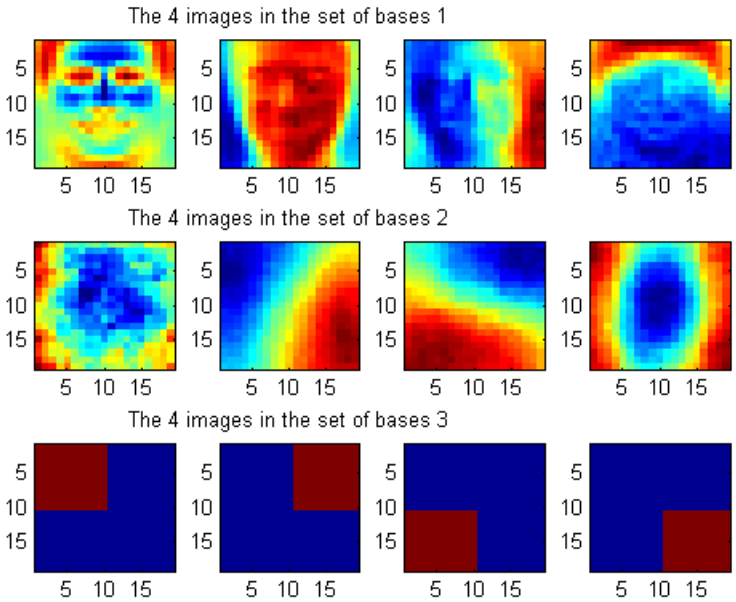



*Figure 7: Images from each of the 3 bases*

Here in the table, we can show the Result of the mean of the error norms for the 6 combinations, i..e 2 test sets against 3 bases. It seems that basis 1 works best for test set 1 whereas basis 2 works best for test set 2.

| Set | Bases 1 | Bases 2 | Bases 3 |
|---|---|---|---|
| Test set 1 | 821.0271 | 860.4754 | 944.9009 |
| Test set 2 | 795.1902 | 649.2013 | 697.3214 |

*Table 1: Mean of the error norms for the six combinations*

**Conclusion**

As we have seen in the second part of the previous task, when we want to project an image $u$ onto a subspace defined by a set of basis image, we should find a `good basis` because the appearance of these basis images affect the appearance of the projection. Thus, we should determine the coordinates $x_1, ..., x_k$ such that the approximate image $u_a$ is as close to the original $u$ as possible, i.e. such that the $MSE = |u - u_a|^2$, i.e. `the mean square error`, is as small as possible, where

$$u_a = \sum_{i}^{k} x_i \phi_i = x_1\phi_1 + ... + x_k\phi_k$$

Using `PCA`, i.e. `Principal Component Analysis`, it is shown that one can calculate good basis images that can represent typical images well.

# References


[1] Szeliski, Computer Vision - Algorithms and Applications, Springer.

[2] Forsyth and Ponce, Computer Vision - A Modern Approach, Pearson Education, ISBN 0-13-191193-7

[3] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*, volume 1 Addison-Wesley, 1991.

[4] Matlab journal demos in Image Analysis. Available: `http://www.ctr.maths.lu.se/matematiklth/personal/kalle/imageanalysis/`.

[5] Linear Algebra. Available: `http://www.immersivemath.com`

[6] Matlab, Text Extraction. Available: `http://www.academia.edu/5500626/MATLAB_TEXT_EXTRACTION_COLOR_CHANGE_GRAY_TO_RGB_and_Smliye_IMAGE_ADD_Addition_DIgital_Image_Processing`

[7] Image Segmentation Example. Available:`http://matlabtricks.com/post-35/a-simple-image-segmentation-example-in-matlab`