FMAN20 - Assignment 1

Eric Rostedt
er4057ro-s

September 10, 2020

Introduction

In this assignment we shall look at some of the basics in image analysis. We shall look at image sampling, histogram equalisation, 4-neighbours connected components, image segmentation, dimensionality of vector spaces, scalar products and norms, image compression and image bases. The assignment is split up into eight tasks where each task covers one of the topics listed above.

Task one

In this first task we shall look at image sampling. Let's consider the function:

$$f(x,y) = x(1-y) + y(1-x) \quad 0 \le x \le 1,\ 0 \le y \le 1$$

We could consider this function as a continuous monochrome image where the function value at some point (x, y) is the intensity. We shall now sample this image into a 5×5 discrete image by taking equidistant sample points. The discrete image will have the appearance:

$$f = \begin{bmatrix} f(0/4,\ 4/4) & f(1/4,\ 4/4) & f(2/4,\ 4/4) & f(3/4,\ 4/4) & f(4/4,\ 4/4) \\ f(0/4,\ 3/4) & f(1/4,\ 3/4) & f(2/4,\ 3/4) & f(3/4,\ 3/4) & f(4/4,\ 3/4) \\ f(0/4,\ 2/4) & f(1/4,\ 2/4) & f(2/4,\ 2/4) & f(3/4,\ 2/4) & f(4/4,\ 2/4) \\ f(0/4,\ 1/4) & f(1/4,\ 1/4) & f(2/4,\ 1/4) & f(3/4,\ 1/4) & f(4/4,\ 1/4) \\ f(0/4,\ 0/4) & f(1/4,\ 0/4) & f(2/4,\ 0/4) & f(3/4,\ 0/4) & f(4/4,\ 0/4) \end{bmatrix}$$

This is easily computed in MatLab using f as a anonymous (vector) function and inserting the linspace vectors $l_x = \text{linspace}(0, 1, 5)$ and $l_y = \text{linspace}(0, 1, 5)^\top$. Thereby we can calculate the discrete image very fast using vectorised operations. Lastly we shall also quantify the discrete image with 16 different gray scale levels. There are different ways to do so, but we shall opt to simply multiply all elements by 15 and then round all elements to its closest integer value (which works since all elements started with values in the interval [0, 1]). When doing so we get the following image:

$$f = \begin{bmatrix} 0 & 4 & 8 & 11 & 15 \\ 4 & 6 & 8 & 9 & 11 \\ 8 & 8 & 8 & 8 & 8 \\ 11 & 9 & 8 & 6 & 4 \\ 15 & 11 & 8 & 4 & 0 \end{bmatrix}$$

Task two

In the second task we shall look at histogram equalisation. Let p_r be the gray level histogram of the image before any transformation and p_s be the gray level histogram of the same image after some transformation. We recall from the lectures that the following histogram relation holds:

$$\int_0^r p_r(t)dt = \int_0^s p_s(t)dt \tag{1}$$

We also recall that the transformation $T(r)$ which maps the histogram p_r to the histogram p_s is found as $T(r) = s$ where s is solved for using equation (1). Our task is to find the transformation $s = T(r)$ which maps $p_r(r) = \frac{3}{2}\sqrt{r}, \quad r \in [0,\ 1]$ to the uniform histogram $p_s = 1, \quad s \in [0,\ 1]$. We simply integrate using the equation (1) to get:

$$\begin{aligned} \int_0^r \frac{3}{2}\sqrt{t}\, dt &= \int_0^s dt \iff \\ \left[t^{3/2}\right]_0^r &= [t]_0^s \iff \\ s = T(r) &= r^{3/2}. \end{aligned}$$

Which is what we wanted to find.

Task three

In this third task we wish to look at neighbourhood of pixels. The 4-neighbours of a pixel at some point (m, n) is defined as the four points $(m \pm 1,\ n \pm 1)$. Let's consider a discrete image f with the following intensities:

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

We are asked to mark all elements with intensity 0 or 1 with a circle, and then colourise all elements with intensity 0 or 1 and at least two 4-neighbours with intensity 0 or 1 as well. We illustrate this in the following Latex table (where red denotes the colourised cells):

Table 1: Elements with 0 or 1 intensity marked with circles and elements with 0 or 1 intensity and at least two 4-neighbours with also 0 or 1 intensity coloured red.

3	3	2	2	①	3	3	3	⓪	2	2	2
🟥⓪	⓪	2	🟥⓪	🟥①	🟥①	🟥⓪	🟥⓪	🟥⓪	3	2	3
①	2	🟥⓪	🟥①	🟥①	🟥⓪	🟥⓪	2	🟥①	⓪	2	3
3	⓪	🟥①	🟥①	🟥⓪	3	3	①	2	3	3	2
3	2	🟥①	🟥⓪	🟥①	2	①	2	3	⓪	3	2
⓪	3	3	🟥①	🟥①	🟥①	🟥⓪	🟥⓪	①	3	3	3
2	⓪	2	🟥①	🟥①	🟥⓪	🟥①	🟥⓪	3	2	⓪	2
2	🟥①	3	🟥⓪	2	3	2	🟥①	2	⓪	3	2
3	🟥⓪	🟥⓪	🟥①	3	3	2	🟥⓪	2	2	2	3
2	2	2	2	🟥①	🟥①	2	🟥⓪	🟥①	⓪	2	①
⓪	2	3	🟥⓪	🟥⓪	🟥①	🟥⓪	🟥⓪	🟥①	3	3	2
⓪	2	3	⓪	3	3	3	⓪	2	①	2	2

Task four

In this fourth task we shall look at image segmentation. We are asked to implement a MatLab function which takes an image and returns segmentation of the image. The input images are binary images of five numbers, an example is show in figure 1.

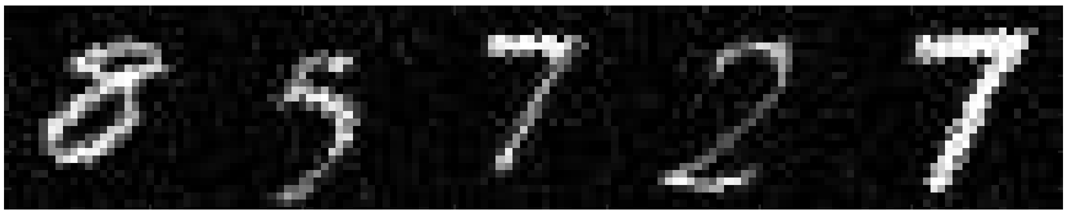
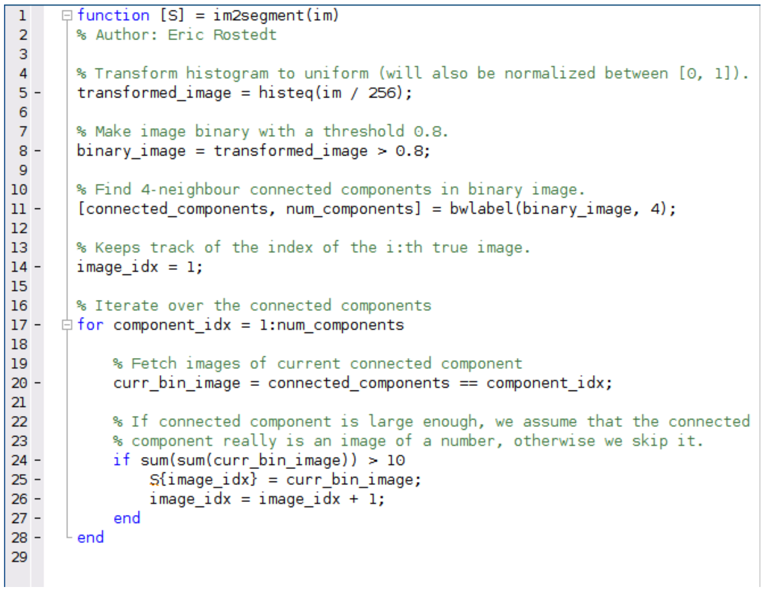


Figure 1: Example of an input image.

To segment the image, we shall use the *bwlabel* function in MatLab which finds 4 (or 8)-neighbours connected components. The full segmentation algorithm does the following:

- Transform the intensities into (normalised) uniform using a histogram transformation (*histeq* in MatLab).
- Makes image binary using some threshold value.
- finds 4-neighbour connected compnents (*bwlabel* in MatLab).
- Loops over all found connected components and saves only the ones that are larger than some threshold number.

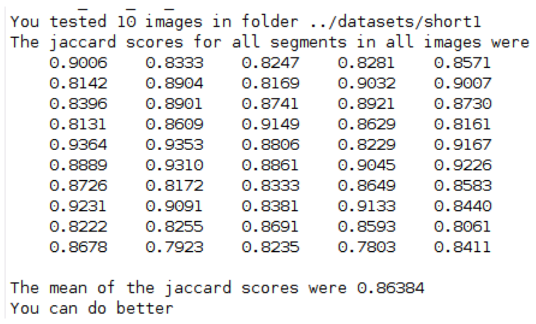
The code is displayed in figure 2 below:



```
function [S] = im2segment(im)
% Author: Eric Rostedt

% Transform histogram to uniform (will also be normalized between [0, 1]).
transformed_image = histeq(im / 256);

% Make image binary with a threshold 0.8.
binary_image = transformed_image > 0.8;

% Find 4-neighbour connected components in binary image.
[connected_components, num_components] = bwlabel(binary_image, 4);

% Keeps track of the index of the i:th true image.
image_idx = 1;

% Iterate over the connected components
for component_idx = 1:num_components

    % Fetch images of current connected component
    curr_bin_image = connected_components == component_idx;

    % If connected component is large enough, we assume that the connected
    % component really is an image of a number, otherwise we skip it.
    if sum(sum(curr_bin_image)) > 10
        S{image_idx} = curr_bin_image;
        image_idx = image_idx + 1;
    end
end
```

Figure 2: Segmentation code.

When running this implementation on the prescribed benchmark code we get the following result:



```
You tested 10 images in folder ../datasets/short1
The jaccard scores for all segments in all images were
    0.9006    0.8333    0.8247    0.8281    0.8571
    0.8142    0.8904    0.8169    0.9032    0.9007
    0.8396    0.8901    0.8741    0.8921    0.8730
    0.8131    0.8609    0.9149    0.8629    0.8161
    0.9364    0.9353    0.8806    0.8229    0.9167
    0.8889    0.9310    0.8861    0.9045    0.9226
    0.8726    0.8172    0.8333    0.8649    0.8583
    0.9231    0.9091    0.8381    0.9133    0.8440
    0.8222    0.8255    0.8691    0.8593    0.8061
    0.8678    0.7923    0.8235    0.7803    0.8411

The mean of the jaccard scores were 0.86384
You can do better
```

Figure 3: Benchmark of algorithm.

This score could be improved by tuning parameters and/or by utilising a more sophisticated algorithm. However we'll be content with this result. Before proceeding to task five, we'll display an example of the segmentation images in figure 4 below:

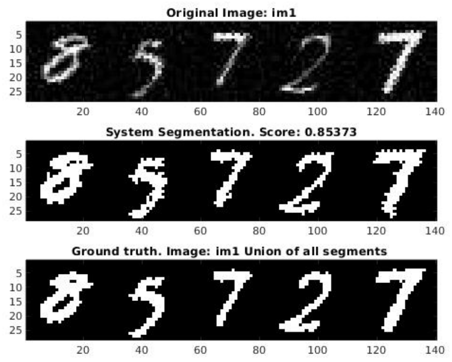

Original Image: im1
System Segmentation. Score: 0.85373
Ground truth. Image: im1 Union of all segments

Figure 4: Example segmented image. Upper part is the input, the middle is our segmentation and the lower is a segmentation for us to compare with.

We observe that our segmentation contains a bit more noise than the comparison segmentation.

Task five

In the fifth task we shall take a brief look at dimensionality. Let's define two vector spaces A and B in the following way:

A The set of gray-scale images with 2×2 pixels.

B The set of gray-scale images with 2000×3000 pixels.

Now we shall answer what the dimension of each vector space and also construct a suitable basis for each vector space. Beginning with A, we could see the image as a vector if we simply vertically concatenate the column vectors in the image matrix. We thereby get the dimension $2 \times 2 = 4$. We could use the standard basis:

$$\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

However this basis could be represented in a more natural way using the standard basis matrices:

$$e_{1,1} = \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}, e_{2,1} = \begin{bmatrix}0 & 0\\1 & 0\end{bmatrix}, e_{1,2} = \begin{bmatrix}0 & 1\\0 & 0\end{bmatrix}, e_{2,2} = \begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}$$

Another common notation is the following:

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

Where the basis with 1 on the $i:th$ row and $j:th$ column for a matrix with dimension $m \times n$ can be accessed as $i + (j-1)m$. We shall now do very similarly for the set B. The dimension is 2000 $\times$ 3000 and the standard basis can be generated as:

$$e_{i,\,j} = e_{i+(j-1)m} = \begin{array}{c@{}c} \begin{bmatrix} 0 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & \cdots & 0 & 0 \end{bmatrix} & i. \\ \begin{array}{ccccccc} & & & j & & & \end{array} & \end{array}$$

Task six

In the sixth task we shall look at scalar products and norms on images. We define the scalar product for two images f and g as a natural extension from the vector scalar product, namely:

$$f \cdot g = \sum_{i=1}^{M} \sum_{j=1}^{N} \bar{f}(i,\, j) g(i,\, j)$$

With a scalar product, we can naturally define a norm for an image f as:

$$||f|| = \sqrt{f \cdot f} = \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} \bar{f}(i,\, j) f(i,\, j)}$$

Now consider the three following images:

$$u = \begin{bmatrix} 6 & -2 \\ 1 & -3 \end{bmatrix}, \quad v = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}, \quad w = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$$

We wish to calculate the following values:

$$||u||, \quad ||v||, \quad ||w||, \quad u \cdot v, \quad u \cdot w, \quad v \cdot w$$

Inserted into the respective definitions we get the following:

$$||u|| = \sqrt{50}, \quad ||v|| = 1, \quad ||w|| = 1$$
$$u \cdot v = 3, \quad u \cdot w = -6, \quad v \cdot w = 0$$

We note that v and w are both unit vectors and that they are orthogonal to each other, hence they are orthonormal. Lastly we shall find the orthogonal projection of u onto the subspace spanned by v and w. We recall the projection formula for a vector u onto a subspace V with orthogonal basis vectors v_i from linear algebra as:

$$\text{proj}_V(u) = \sum_{i=1}^{N} \frac{u \cdot v_i}{v_i \cdot v_i} v_i \tag{2}$$

Since both $\{v,\, w\}$ forms an orthonormal basis we simply get the following projection:

$$\text{proj}_{\{v,\ w\}}\, u = (u \cdot v)v + (u \cdot w)w = 3v - 6w = \frac{3}{2}\begin{bmatrix} 3 & -1 \\ 1 & -3 \end{bmatrix}$$

We shall finally quantify how good the approximation of u is. This will be done by simply looking at the Euclidean distance, i.e:

$$||u - \text{proj}_{\{v,\ w\}}\, u|| = \sqrt{5}$$

Task seven

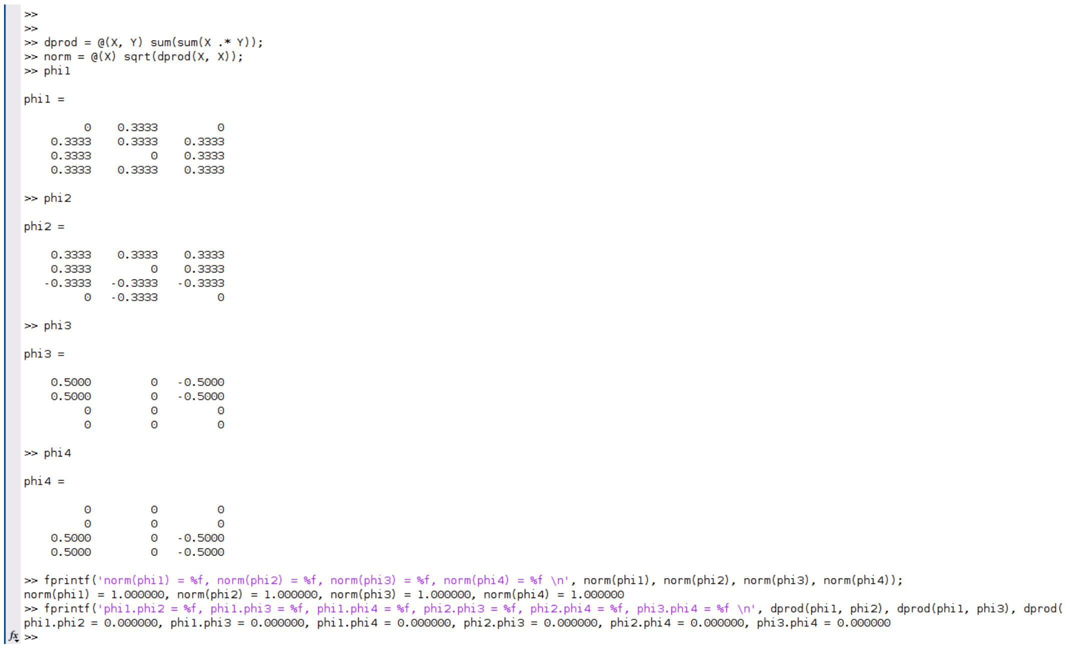
In the seventh task we shall look at image compression. We shall assume that we have a small camera that delivers low resolution 3×4 pixels images. We would like to compress this image by projecting it to the subspace spanned by the following basis matrices:

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

We know that our image has the following intensities:

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

First we shall check so that the basis matrices are orthonormal. We wish to show that all matrices have unit length, i.e $||\phi_i|| = 1,\ \forall i \in [1,\ 2,\ 3,\ 4]$ and that the basis matrices are orthogonal, i.e $\phi_i \cdot \phi_j = 0,\ i \neq j$. We can check this very fast i MatLab, we summarise the results in figure 5



```
>>
>>
>> dprod = @(X, Y) sum(sum(X .* Y));
>> norm = @(X) sqrt(dprod(X, X));
>> phi1

phi1 =

         0    0.3333         0
    0.3333    0.3333    0.3333
    0.3333         0    0.3333
    0.3333    0.3333    0.3333

>> phi2

phi2 =

    0.3333    0.3333    0.3333
    0.3333         0    0.3333
   -0.3333   -0.3333   -0.3333
         0   -0.3333         0

>> phi3

phi3 =

    0.5000         0   -0.5000
    0.5000         0   -0.5000
         0         0         0
         0         0         0

>> phi4

phi4 =

         0         0         0
         0         0         0
    0.5000         0   -0.5000
    0.5000         0   -0.5000

>> fprintf('norm(phi1) = %f, norm(phi2) = %f, norm(phi3) = %f, norm(phi4) = %f \n', norm(phi1), norm(phi2), norm(phi3), norm(phi4));
norm(phi1) = 1.000000, norm(phi2) = 1.000000, norm(phi3) = 1.000000, norm(phi4) = 1.000000
>> fprintf('phi1.phi2 = %f, phi1.phi3 = %f, phi1.phi4 = %f, phi2.phi3 = %f, phi2.phi4 = %f, phi3.phi4 = %f \n', dprod(phi1, phi2), dprod(phi1, phi3), dprod(
phi1.phi2 = 0.000000, phi1.phi3 = 0.000000, phi1.phi4 = 0.000000, phi2.phi3 = 0.000000, phi2.phi4 = 0.000000, phi3.phi4 = 0.000000
>>
```

Figure 5: MatLab print of the norm calculations and scalar products.

We see in figure 5 that all basis matrices have unit length and that they are pairwise orthogonal, and are thereby orthonormal. We have now shown that the basis matrices are orthonormal. Now we wish to find the numbers x_1, x_2, x_3, x_4 such that:

$$||f - f_a|| = ||f - (x_1\phi_1 + x_2\phi_2 + x_3\phi_3 + x_4\phi_4)||$$

is minimized. We can easily find this best approximate image f_a to f by simply orthogonally projecting the image onto the subspace spanned by $\{\phi_1, \phi_2, \phi_3, \phi_4\}$. We simply insert our image and basis matrices into the projection formula (2) to get:

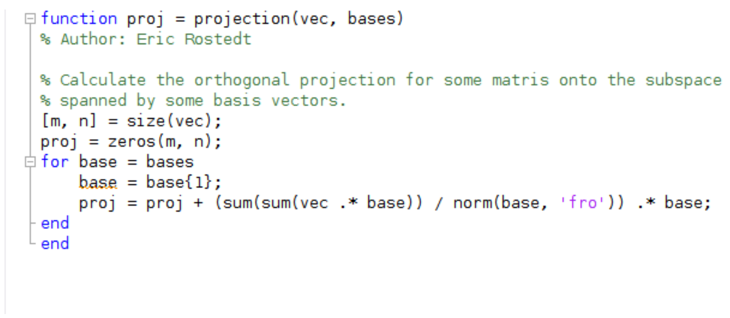
$$f_a = \text{proj}_{\{\phi_1, \phi_2, \phi_3, \phi_4\}} f = \underbrace{(f \cdot \phi_1)}_{x_1} \phi_1 + \underbrace{(f \cdot \phi_2)}_{x_2} \phi_2 + \underbrace{(f \cdot \phi_3)}_{x_3} \phi_3 + \underbrace{(f \cdot \phi_4)}_{x_4} \phi_4 =$$

$$\underbrace{\frac{50}{3}}_{x_1} \phi_1 + \underbrace{2}_{x_2} \phi_2 + \underbrace{\frac{3}{2}}_{x_3} \phi_3 + \underbrace{-4}_{x_4} \phi_4 = \begin{bmatrix} 1.4167 & 6.2222 & -0.0833 \\ 6.9722 & 5.5556 & 5.4722 \\ 2.8889 & -0.6667 & 6.8889 \\ 3.5556 & 4.8889 & 7.5556 \end{bmatrix}$$

So we have now found the optimal parameters x_1, x_2, x_3, x_4 and the best approximate image f_a. We can find a quantitative measure of how good the approximation is by taking the Euclidean distance between the true image and the approximate image, i.e $||f - f_a||$ which turns out to be 11.8310.

Task eight

In this last task we shall look at image bases. We are granted two data set of images and three different bases. The objective is to evaluate how well the different bases are at approximating the images from respective data set. So firstly we write a projection function which takes an image matrix and a cell array of basis matrices. We implement the method according to the formula (2):



```
function proj = projection(vec, bases)
% Author: Eric Rostedt

% Calculate the orthogonal projection for some matris onto the subspace
% spanned by some basis vectors.
[m, n] = size(vec);
proj = zeros(m, n);
for base = bases
    base = base{1};
    proj = proj + (sum(sum(vec .* base)) / norm(base, 'fro')) .* base;
end
end
```

Figure 6: Code for projection.

We shall now take a quick look at both of the data set and the three given bases. Firstly we look at four images from the first data set displayed in figure 7:

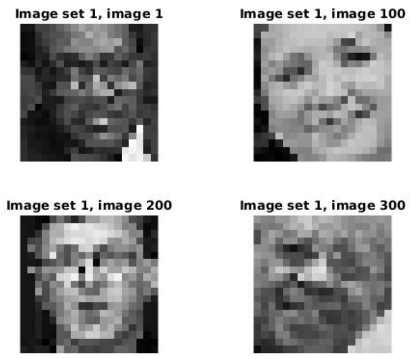


Figure 7: Four images from the first data set

The images in the first data set looks like faces. Then we shall display the second data set in figure 8:

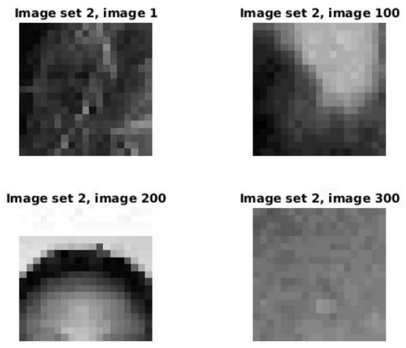


Figure 8: Four images from the second data set

The images in the second data set is a lot harder to distinguish what they are, but looking at image 200 it appears that the images might be zoomed in images from the first data set. We continue along and shall now look at the different bases, starting with base 1 in figure 9:

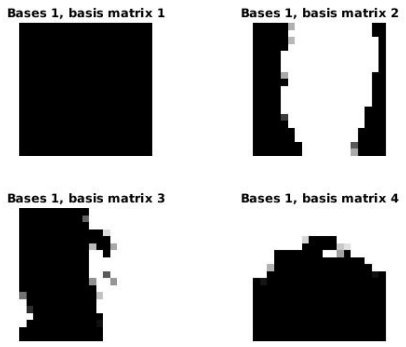


Figure 9: First base

From figure 9 it appears that the first basis holds facial features. We display the second base in figure 10:

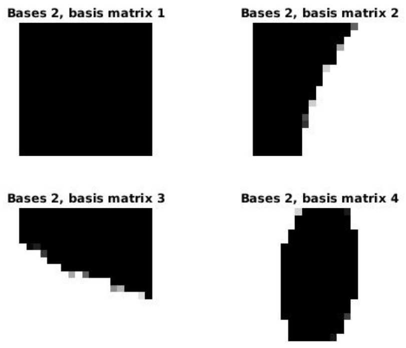


Figure 10: Second base

The second base holds features that are not apparent what they are. However basis matrix 2 and 3 could be edges of faces while the fourth is the background of a face. We display the last image in figure 11

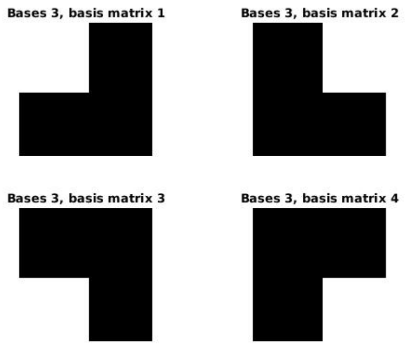
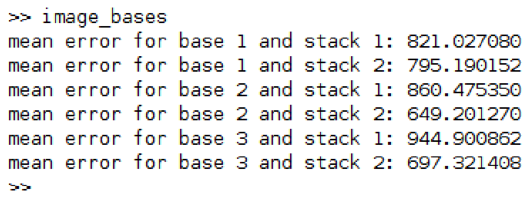


Figure 11: Third base

The third base is split up so that the basis matrices holds one of the four quadrants each. We iterate over all bases and both data set and calculate the mean error for the images projected onto the different bases. We display the mean errors in figure 12



```
>> image_bases
mean error for base 1 and stack 1: 821.027080
mean error for base 1 and stack 2: 795.190152
mean error for base 2 and stack 1: 860.475350
mean error for base 2 and stack 2: 649.201270
mean error for base 3 and stack 1: 944.900862
mean error for base 3 and stack 2: 697.321408
>>
```

Figure 12: Mean errors for each data set on each base.

We note that the best basis for the first data set appears to be the first one. Which is logical since the first basis appears to have facial features and the first data set consists of images of faces. The best basis for the second image set is the second basis, which would make sense if our guess that the second image set is zoomed images of faces and that the second basis holds face edge and background features.