Image Analysis Assignment 1

Ruqayyah Nabage

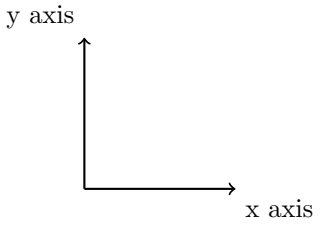
September 2023

1 Image Sampling

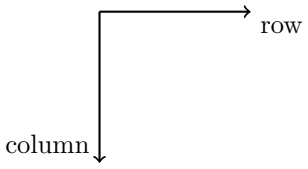
$$\mathrm{f} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 4 & 6 & 8 \\ 0 & 4 & 8 & 12 & 16 \\ 0 & 6 & 12 & 17 & 23 \\ 0 & 8 & 16 & 23 & 31 \end{pmatrix}$$

I started by creating x and y matrices with values from 0 to 1 divided into 5 since our resulting image is a 5x5 matrix. I did this using the linspace function. x and y were equal to (0, 0.25, 0.5, 0.75, 1).

Next, I created a new matrix for the values of $f(x_i, y_j)$ for each point in x and y to get the value of each pixel in the image. I used a nested for loop to achieve this. The resulting matrix was not on the right axes as matrices are populated from top to bottom and the image is on a cartesian plane so the values should be from the bottom to the top. To amend this I flipped the rows of the matrix from top down.



Cartesian Plane



Matrix

Next, I quantized the discrete image with 32 levels of gray scale; i.e I mapped the levels of the image which were from 0 - 1 to 0 - 31. To achieve this I multiplied the image matrix with 31 and used a rounding off function to round them off into integers to complete the quantization.

2 Histogram Equalization

The transformation required for a uniform histogram,

$$s = T(r) = \int_0^r p(r)\,dr$$

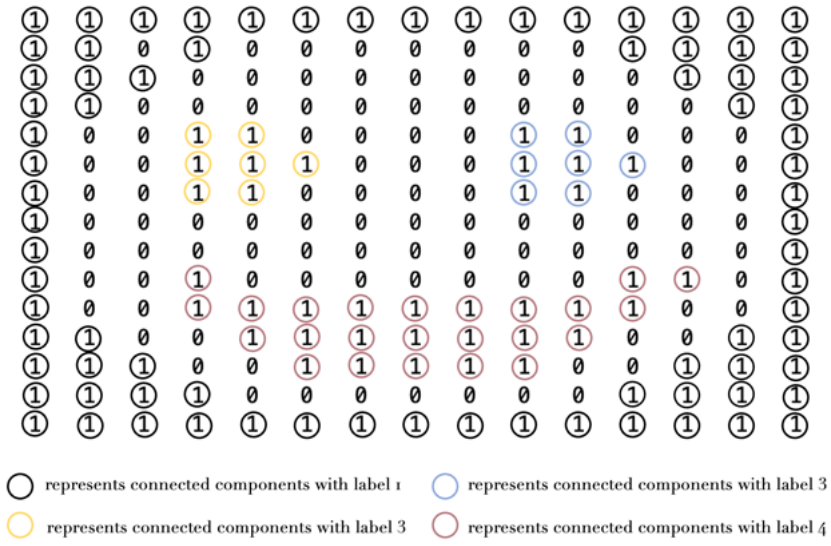
$$and \quad p(r) = 6r(1-r)$$

$$T(r) = \int_0^r 6r(1-r)\,dr = 6\int_0^r (r - r^2)\,dr$$

$$T(r) = 6\left[\frac{r^2}{2} - \frac{r^3}{3}\right]_0^r = 3r^2 - 2r^3$$

$$s = T(r) = 3r^2 - 2r^3$$

3 Neighbourhood of Pixels



4 Segmentation part of OCR

```
function S = im2segment(im)

im = uint8(im); %converting the image into 8 bit

im_f = imgaussfilt(im); %filtering the image for noise

bnd = 40; % set some arbitrary bound on thresholding
```

```
im_t = im_f>bnd; % threshold the image

L = bwlabel(im_t); % label connected components in image
nrseg = max(L(:)); % find out number of segments
S = cell(1,nrseg); %creating the cell array S

for kk = 1:nrseg    %created a for loop to store the different segments in
        S{kk} = (L==kk); %different cells of the array when the index is the same as segment number
end
```

Listing 1: Code

```
>> inl1_test_and_benchmark
You tested 10 images in folder ../datasets/short1
The jaccard scores for all segments in all images were
    0.9512    0.8868    0.9302    0.7951    0.9444
    0.9010    0.9424    0.8692    0.9658    0.9545
    0.9419    0.9454    0.9528    0.9474    0.9333
    0.7451    0.9137    0.9438    0.9310    0.9172
    0.9262    0.9170    0.9493    0.8958    0.9448
    0.9624    0.9298    0.9732    0.9527    0.9931
    0.9461    0.8824    0.9115    0.9565    0.9328
    0.9298    0.9598    0.7745         0         0
    0.9268    0.9203    0.9432    0.9449    0.8901
    0.9633    0.2348         0         0         0

The mean of the jaccard scores were 0.81747
This is good!
>>
```

Listing 2: Text Results of Benchmark Script

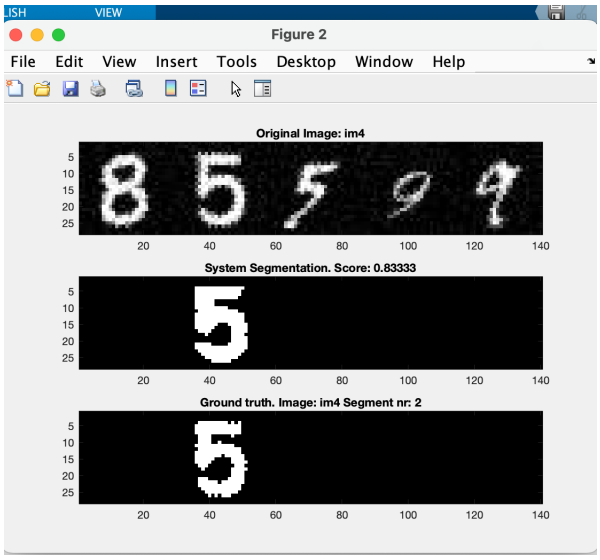


Figure 1: Example of input image and resulting segmentation

5 Dimensionality

- In A the dimension k is $3 \times 2 = 6$.
- An example basis for A is:

$$e_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} e_2 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} e_3 = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{pmatrix}$$

$$e_4 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} e_5 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} e_6 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

- In B the dimension k is $1500 \times 2000 = 3 \times 10^6$
- The basis elements in B can be chosen such that 3×10^6 zero matrices are created. Using a for loop, each matrix $e_i$is only populated by a 1 at the ith position using the MATLAB notation of single subscript (linear indexing) which goes down each column in order.

6 Scalar Product and Norm on Images

- The scalar product for images is defined as a summation of the product of each corresponding element in the image matrices. Mathematically:

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

- The norm of an image is defined as the length of an image. Mathematically:

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

- $\|u\| = \sqrt{(3)^2 + (-7)^2 + (-1)^2 + (4)^2} = 8.6603$

  $\|v\| = \sqrt{(\frac{1}{2})^2 + (-\frac{1}{2})^2 + (-\frac{1}{2})^2 + (\frac{1}{2})^2} = 1$

  $\|w\| = \sqrt{(-\frac{1}{2})^2 + (\frac{1}{2})^2 + (-\frac{1}{2})^2 + (\frac{1}{2})^2} = 1$

  $u \cdot v = (3 \times \frac{1}{2}) + (-7 \times -\frac{1}{2}) + (-1 \times -\frac{1}{2}) + (4 \times \frac{1}{2}) = 7.5$

  $u \cdot w = (3 \times -\frac{1}{2}) + (-7 \times \frac{1}{2}) + (-1 \times -\frac{1}{2}) + (4 \times \frac{1}{2}) = -2.5$

  $v \cdot w = (\frac{1}{2} \times -\frac{1}{2}) + (-\frac{1}{2} \times \frac{1}{2}) + (-\frac{1}{2} \times -\frac{1}{2}) + (\frac{1}{2} \times \frac{1}{2}) = 0$

- Yes, u, w are orthonormal because they are orthogonal to each other ($v \cdot w = 0$) and they both have a norm of 1.
- The orthogonal projection of u onto the subspace spanned by v, w is:

$$u_p = (u \cdot v)v + (u \cdot w)w$$

$$u_p = \frac{15}{2} \times \frac{1}{2}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} - \frac{5}{2} \times \frac{1}{2}\begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 5 & -5 \\ -2.5 & 2.5 \end{pmatrix}$$

I think the resulting projection is a good approximation because the error norm $\|u - u_p\|$ isn't too high at 3.5.

7 Image Compression

For a set of images to be orthonormal, $\phi_i \cdot \phi_j = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$. For this to be true, $\phi_1 \cdot \phi_2$, $\phi_1 \cdot \phi_3$, $\phi_1 \cdot \phi_4$, $\phi_2 \cdot \phi_3$, $\phi_2 \cdot \phi_4$, $\phi_3 \cdot \phi_4$, should be equal to 0 and $\phi_1 \cdot \phi_1$, $\phi_2 \cdot \phi_2$, $\phi_3 \cdot \phi_3$, $\phi_4 \cdot \phi_4$, should be equal to 1. So those are the combinations we need to test.

$$\phi_1.\phi_2 = 0, \quad \phi_1 \cdot \phi_3 = 0$$
$$\phi_1 \cdot \phi_4 = 0, \quad \phi_2 \cdot \phi_3 = 0$$
$$\phi_2 \cdot \phi_4 = 0, \quad \phi_3 \cdot \phi_4 = 0$$
$$\phi_1 \cdot \phi_1 = 1, \quad \phi_2 \cdot \phi_2 = 1$$
$$\phi_3 \cdot \phi_3 = 1, \quad \phi_4 \cdot \phi_4 = 1$$

- To determine coordinates x_1, x_2, x_3, x_4 such that f_a is as close to f as possible, we project f orthogonally onto a subspace spanned by $\phi_1, \phi_2, \phi_3, \phi_4$. This translates to x_i being the scalar dot product of f and ϕ_i.

$$x_i = f \cdot \phi_i$$

- $x_1 = f \cdot \phi_1 = 1.5 \quad x_2 = f \cdot \phi_2 = 1.67 \quad x_3 = f \cdot \phi_3 = 17 \quad x_4 = f \cdot \phi_4 = -4$
- $f_a = x_1\phi_1 + x_2\phi_2 + x_3\phi_3 + x_4\phi_4$

$$f_a = \begin{pmatrix} 1.3056 & 6.2222 & -0.19144 \\ 6.9722 & 5.6667 & 5.4722 \\ 3.1111 & -0.5556 & 7.1111 \\ 3.6667 & 5.1111 & 7.6667 \end{pmatrix}$$

- I think f is close to f_a because the error $\|f - f_a\| = 11.7$ which is almost half the norm of f. I think the approximations in number 6 are better because the error norm in that example was 3.5 and also I think the larger the images become the larger the difference between them becomes.

8 Image Bases

I created two separate functions, one for calculating the scalar product of images; returning a scalar as the result. The second function I created calculates a projection of an image and the error norm. It takes in an input matrix, and four basis elements It returns the projected image and the error norm. Then the formula for orthogonal projection is entered with the scalar dot product function I created first. The function calculates the error norm using the norm function and taking the norm of the difference between the original image and the projected image.

$$\hat{f} = (f \cdot \phi_1)\phi_1 + (f \cdot \phi_2)\phi_2 + (f \cdot \phi_3)\phi_3 + (f \cdot \phi_4)\phi_4$$

```
function [I] = imgdot(A, B)

I = sum((A.*B),"all");
```

Listing 3: Code

```
function [up,r] = projection(u,e1,e2,e3,e4)
up = (imgdot(u,e1)*e1)+(imgdot(u,e2)*e2)+(imgdot(u,e3)*e3)+(imgdot(
    u,e4)*e4);
r = norm((u-up), 'fro');
```

Listing 4: Code

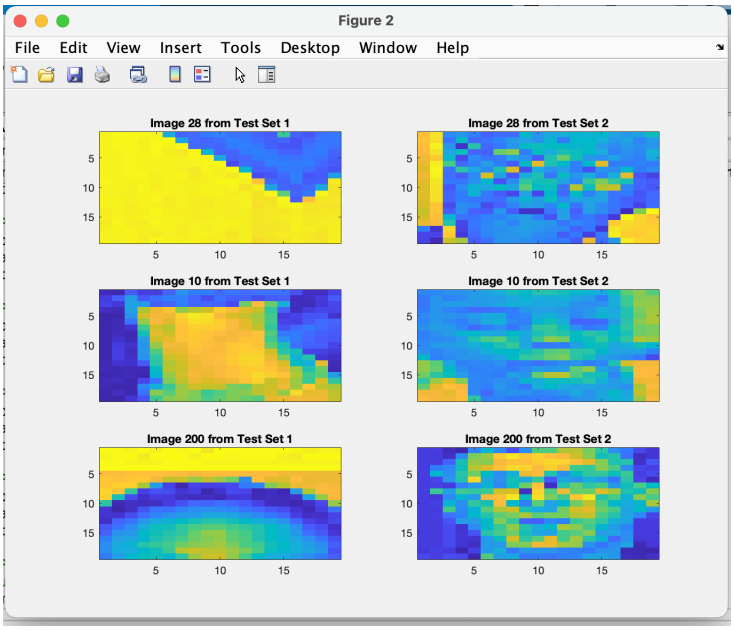


Figure 2: Example plot of a few images from both test sets.

- The images in test set 1 look very abstract and look like they may be closeups of body parts, while the images in test set 2 are very clearly faces.

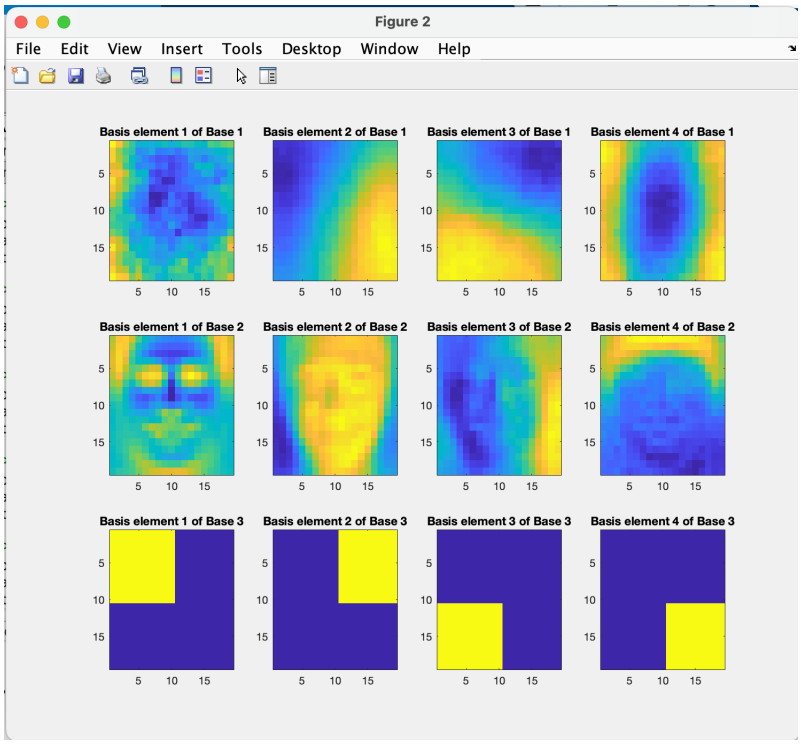


Figure 3: Plot of the four basis elements in each base.

- The basis elements in base 1 also look quite abstract such that even when images from test set 2 (which as we saw are quite obviously faces) are projected onto base 1 they turn into abstract images too. Meanwhile, the basis elements in base 2 look like faces, so when images from test set 1 (abstract looking images) are projected onto it, they look like faces. The basis elements of base 3 on the other hand look like parts of cross or grid so it has the effect of turning the images into a grid looking image. To illustrate the effect further I have included a plot of the same image I am displaying the image at the same index in both test sets and also these same 2 images onto the different bases.

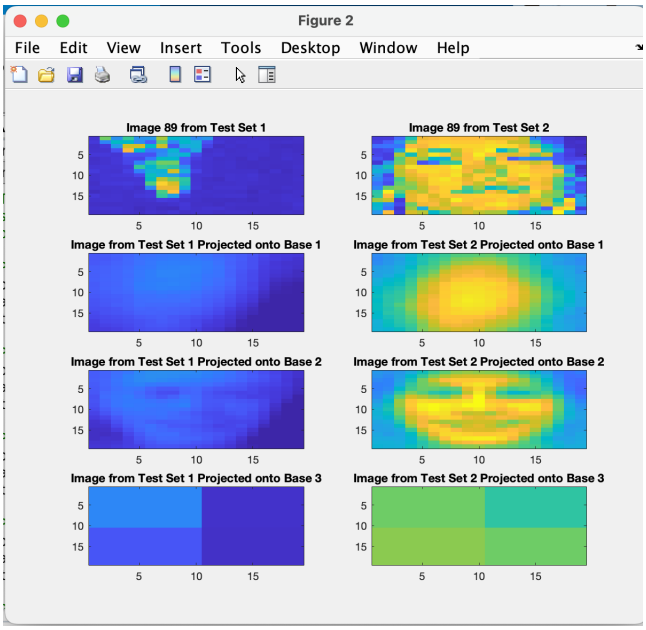


Figure 4: Plot of image at index 89 from both test sets and their projections.

Table 1: Error Norms for the Six Combinations

Test Set	**Base**		
	Base 1	**Base 2**	**Base 3**
Test Set 1	649.20	795.12	697.32
Test Set 2	860.48	821.03	944.90

- I think the basis elements in Base 1 work best for the images in test set 1. I chose Base 1 because it has the lowest error norm, but also intuitively it makes sense that it has the lowest error norm because they are the most similar visually.
- I think the basis elements in Base 2 work best for the images in test set 2 for the same reasons I gave for choosing Base 1 for test set 1.