Assignment 1 - FMAN20

Maks Epsteins

September 10, 2021

Exercise 1

First the matrix was created using the following Julia function (Julia was used since language was not specified):

```
matrix = [x*(1-y) for x = 0:0.25:1, y = 1:-0.25:0]
```

Which results in the following matrix:

$$matrix = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0625 & 0.125 & 0.1875 & 0.25 \\ 0.0 & 0.125 & 0.25 & 0.375 & 0.5 \\ 0.0 & 0.1875 & 0.375 & 0.5625 & 0.75 \\ 0.0 & 0.25 & 0.5 & 0.75 & 1.0 \end{bmatrix}$$

Then the quantization is made with the following code:

```
matrix = [x*(1-y) for x = 0:0.25:1, y = 1:-0.25:0]
graylevel = 32

##This function goes through each mapping (range of intensities f.e
##0.01 to 0.02 is mapped to a gray level f.e 2) and finds one
##where the value v is withing the range of the mapping, then returns
##the gray level that said ranges matches to.
function quantizationMapping(v, mappings)
    if (v > 1 || v < 0) 0 end
    for m in mappings
        if v <= m[1][2] && v >= m[1][1] return m[2] end
    end
    return 0
end

##This function zips two collections:
##the gray levels and the matching intensity levels
##between 0 and 1.
##The intensity levels are then mapped into arrays with start and end
##value such that the start and end values are delta away from the
##intensity level.
```

```
function quantization(matrix)
    delta = (1/(graylevel - 1))/2
    mappings = map(
        e -> ([e[1]-delta,e[1]+delta],e[2]),
        collect(zip(0:1/(graylevel - 1):1, 0:1:(graylevel - 1))))
    return map(v -> quantizationMapping(v, mappings),matrix)
end

println(quantization(matrix))
```

This code returns the following matrix:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 4 & 6 & 8 \\ 0 & 4 & 8 & 12 & 15 \\ 0 & 6 & 12 & 17 & 23 \\ 0 & 8 & 15 & 23 & 31 \end{bmatrix}$$

Exercise 2

$\mathrm{P}_r = 6r(1-r), r \in [0,1]$
$P_s = 1$
The graylevel transformation which outputs a flat histogram is the following:
$T(r) = \int_0^r P_r(t)dt = \int_0^r 6r(1-r)dt = [6rt - 6r^2t]_0^r = 6r^2(1-r)$

Exercise 3

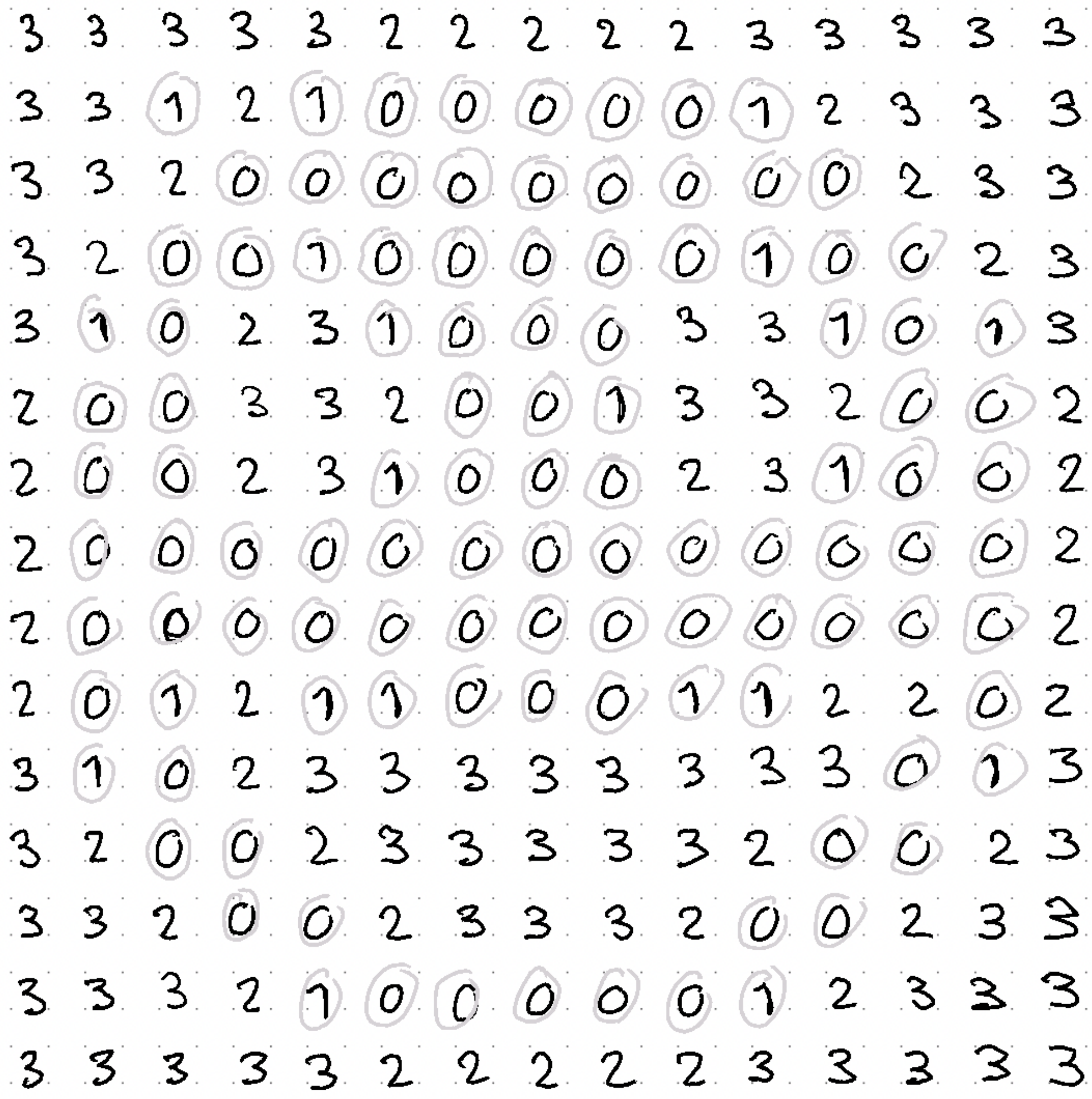

Figure 1: 3a

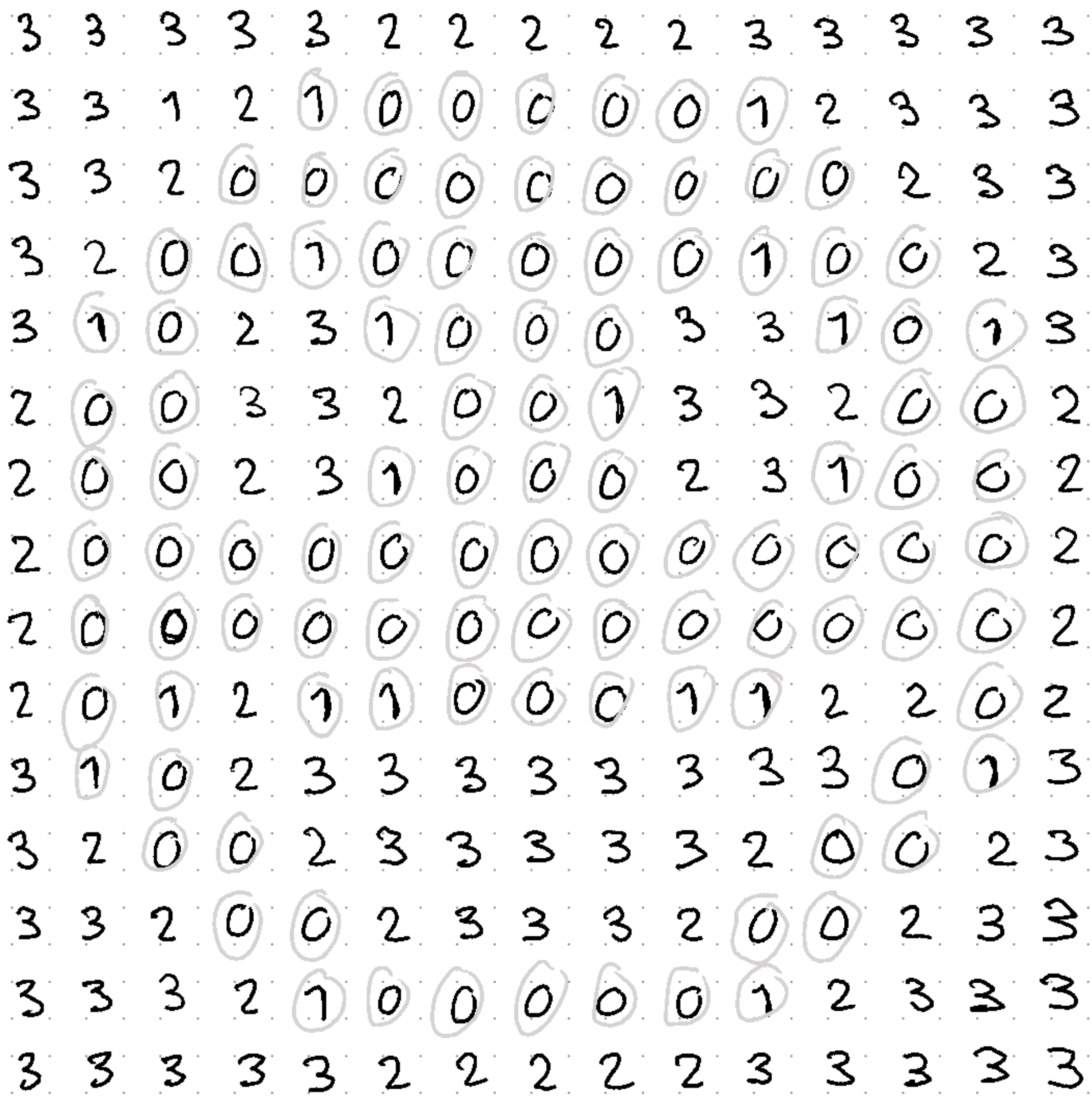
3 3 3 3 3 2 2 2 2 2 3 3 3 3 3
3 3 1 2 1 0 0 0 0 0 1 2 3 3 3
3 3 2 0 0 0 0 0 0 0 0 0 2 3 3
3 2 0 0 1 0 0 0 0 0 1 0 0 2 3
3 1 0 2 3 1 0 0 0 3 3 1 0 1 3
2 0 0 3 3 2 0 0 1 3 3 2 0 0 2
2 0 0 2 3 1 0 0 0 2 3 1 0 0 2
2 0 0 0 0 0 0 0 0 0 0 0 0 0 2
2 0 0 0 0 0 0 0 0 0 0 0 0 0 2
2 0 1 2 1 1 0 0 0 1 1 2 2 0 2
3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

Figure 2: 3b

Exercise 4

```
function S = im2segment(im)

nrofsegments = 5;
S = cell(1,nrofsegments);
m = size(im,1);
n = size(im,2);

for r = 1:m
    for c=1:n
        %Set threshold value at 27 because that value gave the best results
        if im(r, c) >= 27
            im(r, c) = 1;
        else
            im(r,c) = 0;
        end
    end
end

%I use a convolution kernel in order to calculate the amount of neighbors at each index
%to try to fix some errors from the threshholding.
neighborKernel = [1,1,1;1,0,1;1,1,1];
nbrNeighbors = conv2(im, neighborKernel, 'same');

%These loops should be pretty self explanatory
for r = 1:m
    for c=1:n
        if nbrNeighbors(r, c) <= 2
            im(r, c) = 0;
        elseif nbrNeighbors(r, c) >= 6
            im(r, c) = 1;
        end
    end
end

im = bwlabel(im);
%I use the k-means segmentation method to segment the numbers
[im,~] = imsegkmeans(uint8(im), nrofsegments + 1);
%blackZone is a variable checks which number is assigned to the background of the image.
blackZone = mode(mode(im));

for kk = 1:nrofsegments
    currentVal = 0;
    S{kk} = zeros(m,n);
    for c = 1:n
        for r = 1:m
            if im(r,c) ~= blackZone && currentVal == 0
```

```
                currentVal = im(r,c);
            end
            if im(r,c) == currentVal
                S{kk}(r,c) = 255;
                im(r,c) = blackZone;
            else
                S{kk}(r,c) = 0;
            end
        end
    end
end
```

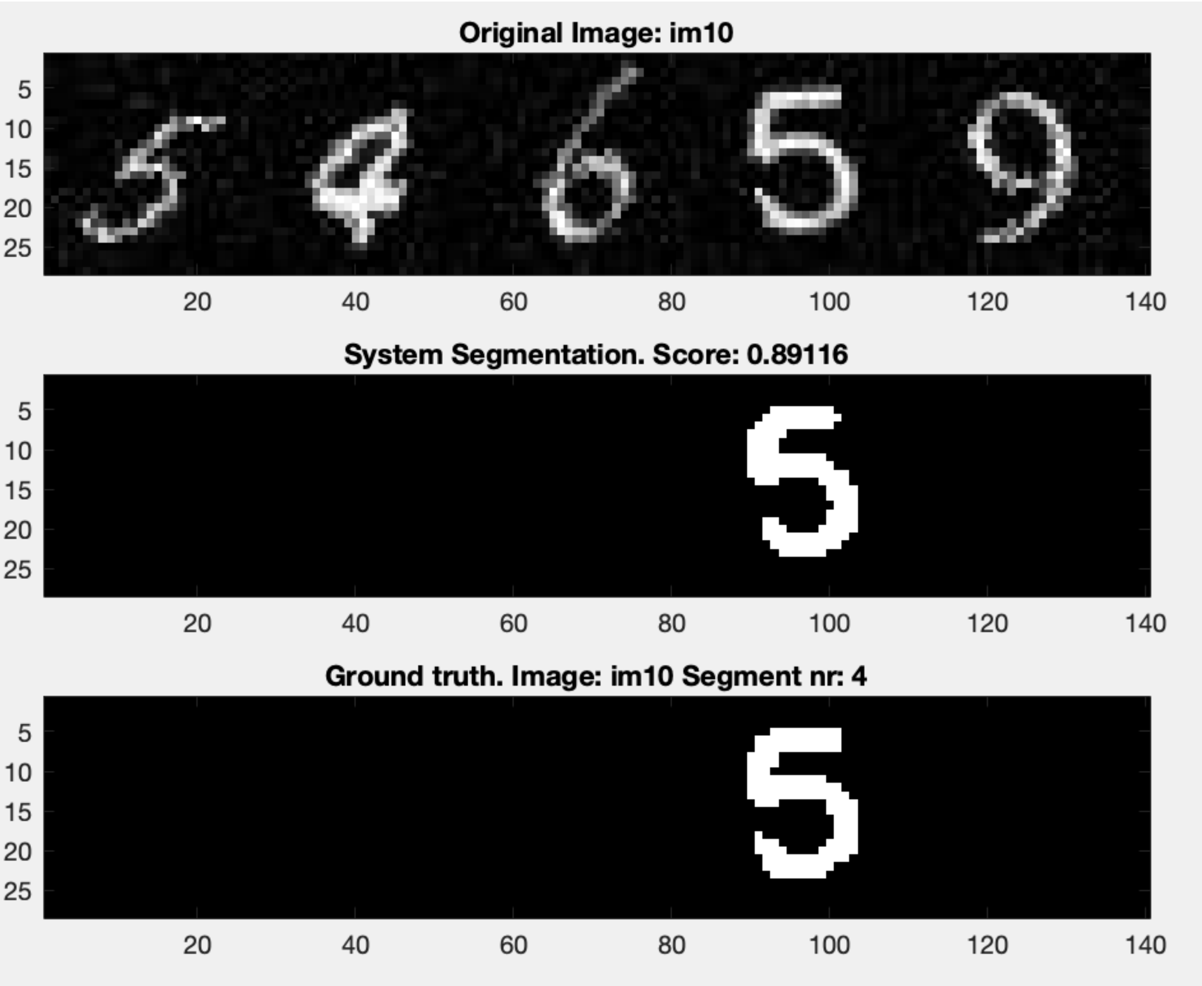

Figure 3: Result from running the segmentation algorithm

```
You tested 10 images in folder ../datasets/short1
The jaccard scores for all segments in all images were
    0.9162    0.8636    0.8989    0.8226    0.9220
    0.8544    0.9085    0.8647    0.9020    0.9624
    0.8674    0.9341    0.9015    0.9259    0.9167
    0.7619    0.9281    0.9022    0.9391    0.8765
    0.9180    0.9202    0.9137    0.8400    0.9122
    0.9078    0.9298    0.9286    0.9527    0.9156
    0.9167    0.9176    0.8824    0.9007    0.8760
    0.9294    0.9213    0.8462    0.9231    0.8592
    0.8672    0.8786    0.9371    0.9291    0.8778
    0.9107    0.8205    0.9140    0.9099    0.7900

The mean of the jaccard scores were 0.89629
You can do better
```

Figure 4: Text results from the benchmark script

Exercise 5

a)

The dimension of the A matrix is k = 6 since 2*3 = 6. An example basis for A is:

$$e_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, e_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, e_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, e_4 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, e_5 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, e_6 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

b)

The dimension of the B matrix is k = $6 \cdot 10^6$ since 2000*3000 = $6 \cdot 10^6$. The basis elements for the B matrix can be chosen by having 2000*3000 basis elements with a singular 1 at a unique index in each matrix, while the rest of the values are set to 0.

Exercise 6

a) The scalar product of two images is defined as (according to the lectures):

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

b) The norm of an image is defined as (according to the lectures):

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

c)

$$||u||_2 = \sqrt{4^2 + (-1)^2 + (-2)^2 + 5^2} \approx 6.7823$$
$$||v||_2 = \sqrt{(-1/2)^2 + (1/2)^2 + (1/2)^2 + (-1/2)^2} = 1$$
$$||w||_2 = \sqrt{(-1/2)^2 + (1/2)^2 + (1/2)^2 + (-1/2)^2} = 1$$

$$u \cdot v = 4 \cdot (-1/2) + (-1) \cdot 1 + (-2) \cdot 1/2 + 5 \cdot (-1) = -9$$
$$u \cdot w = 4 \cdot 1/2 + (-1) \cdot (-1/2) + (-2) \cdot 1/2 + 5 \cdot (-1) = -3.5$$
$$v \cdot w = 1/2(-1 \cdot 1 + (-1) \cdot 1 + 1 \cdot 1 + (-1) \cdot (-1)) = 0$$

d)

$$v \cdot w = 0, v \cdot v = 1, w \cdot w = 1$$

=> v,w is an ON-basis in $\mathbb{R}^2$. The dot products were calculated in Julia using the LinearAlgebra package.

e) The subspace spanned by v,w is denoted with Ω.

$$u_\Omega = x_1 \cdot v + x_2 \cdot w$$

$$x_1 = u \cdot v = -6, x_2 = u \cdot w = -1$$
$$u_\Omega = -6 \cdot v - 1 \cdot w = 1/2 \cdot \begin{bmatrix} 5 & -5 \\ -7 & 7 \end{bmatrix}$$

Exercise 7

a) Table of dot products calculated using Julia:

	ϕ_1	ϕ_2	ϕ_3	ϕ_4
ϕ_1	1	0	0	0
ϕ_2	-	1	0	0
ϕ_3	-	-	1	0
ϕ_4	-	-	-	1

b/c)

$$x_1 = f \cdot \phi_1 = 17, x_2 = f \cdot \phi_2 = -4, x_3 = f \cdot \phi_3 = 3/2, x_4 = f \cdot \phi_4 = 5/3$$

d)

$$f_a = x_1 \cdot \phi_1 + x_2 \cdot \phi_2 + x_3 \cdot \phi_3 + x_4 \cdot \phi_4 = \begin{bmatrix} 1.30556 & 6.22222 & -0.194444 \\ 6.97222 & 5.6667 & 5.47222 \\ 3.11111 & -0.555556 & 7.11111 \\ 3.66667 & 5.11111 & 7.66667 \end{bmatrix}$$

e) Not really, would expect the matrix to be more similar.

Exercise 8

```
load('assignment1bases.mat')

norms = {zeros(400,1), zeros(400,1), zeros(400,1)};
for i = 1:3
    %Cell array consisting of the 4 bases
    e = {bases{i}(:,:,1); bases{i}(:,:,2); bases{i}(:,:,3); bases{i}(:,:,4)};
    for j = 1:size(stacks{1},3)
        u = stacks{1}(:,:,j);
        %Calculate the dot products of the image and each basis
        x = {dot(u, e{1}), dot(u, e{2}), dot(u, e{3}), dot(u, e{4})};
        u_p = zeros(size(x{1}));
        for k = 1:4
            u_p = u_p + x{k}*e{k};
        end
        norms{i}(j) = norm(u - u_p);
    end
end

means = [mean(norms{1}) mean(norms{2}) mean(norms{3})];
disp(means)
```

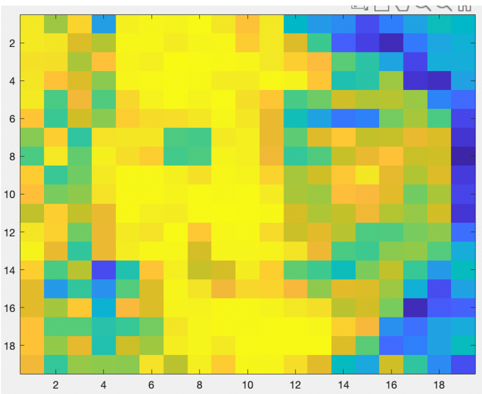

Figure 5: Image of Stack 1 which appears to consist of faces

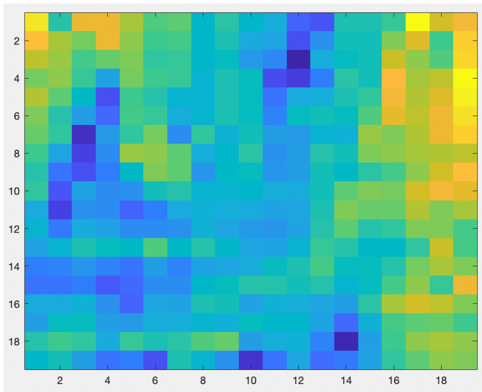

Figure 6: Image from Stack 2 which appears to consist of some sort of heat maps

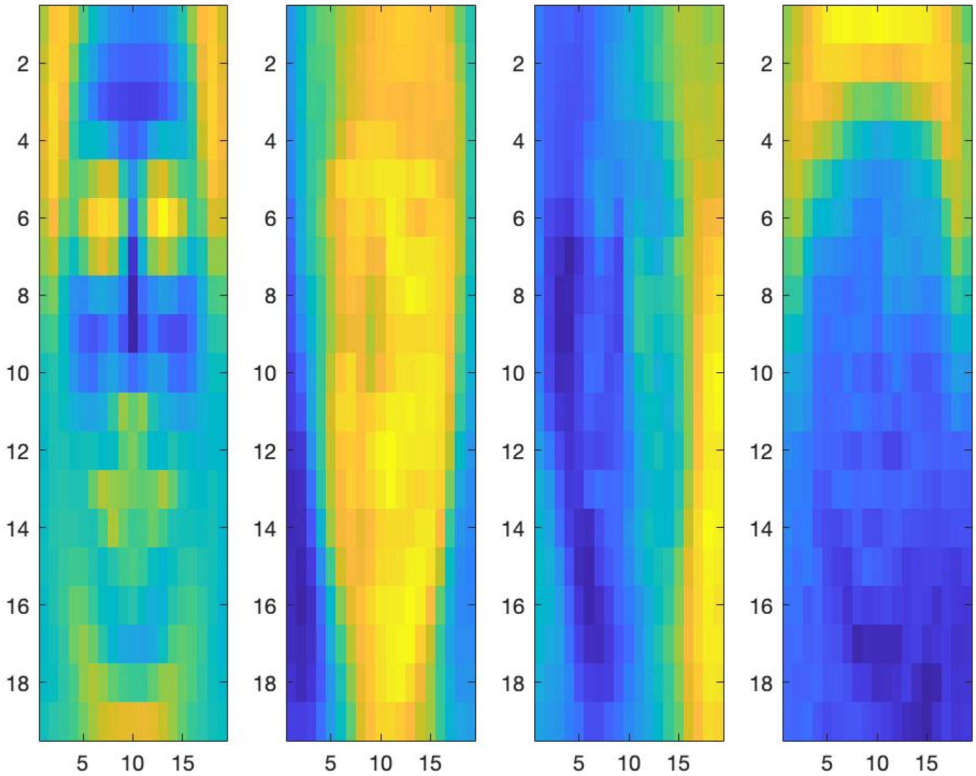

Figure 7: bases 1-4

These bases appear to resemble faces and should therefore work best for the images in stack 1 that also appeared to consist of faces.

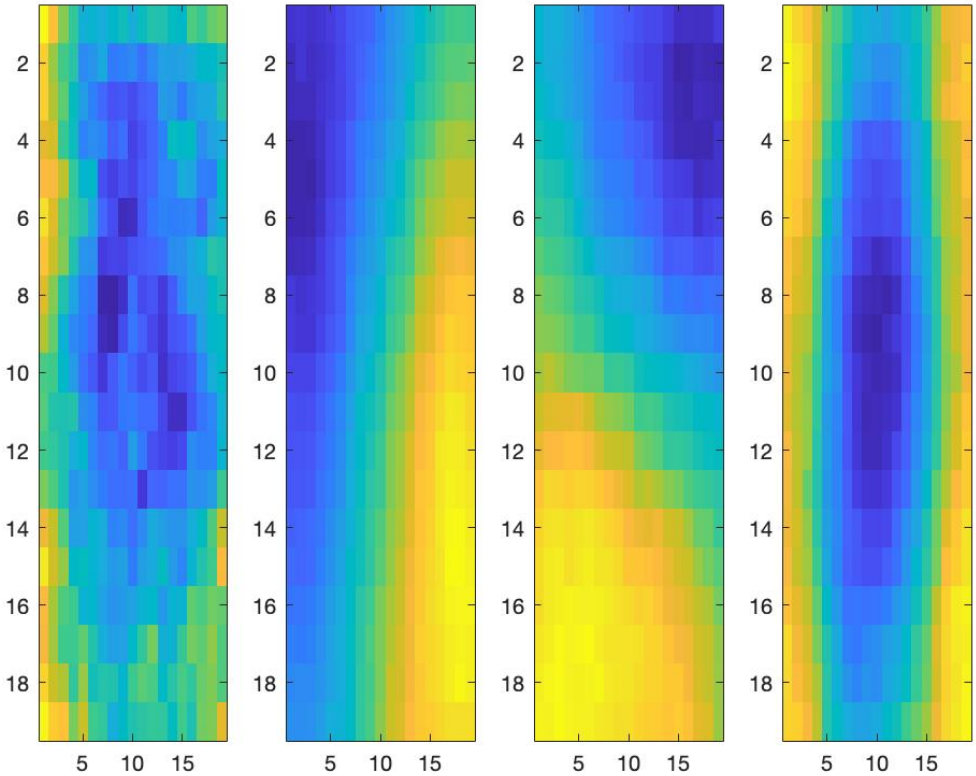

Figure 8: bases 5-8

These bases appear to resemble heat-maps just like the second stack does, so these bases should work best for the second stack.

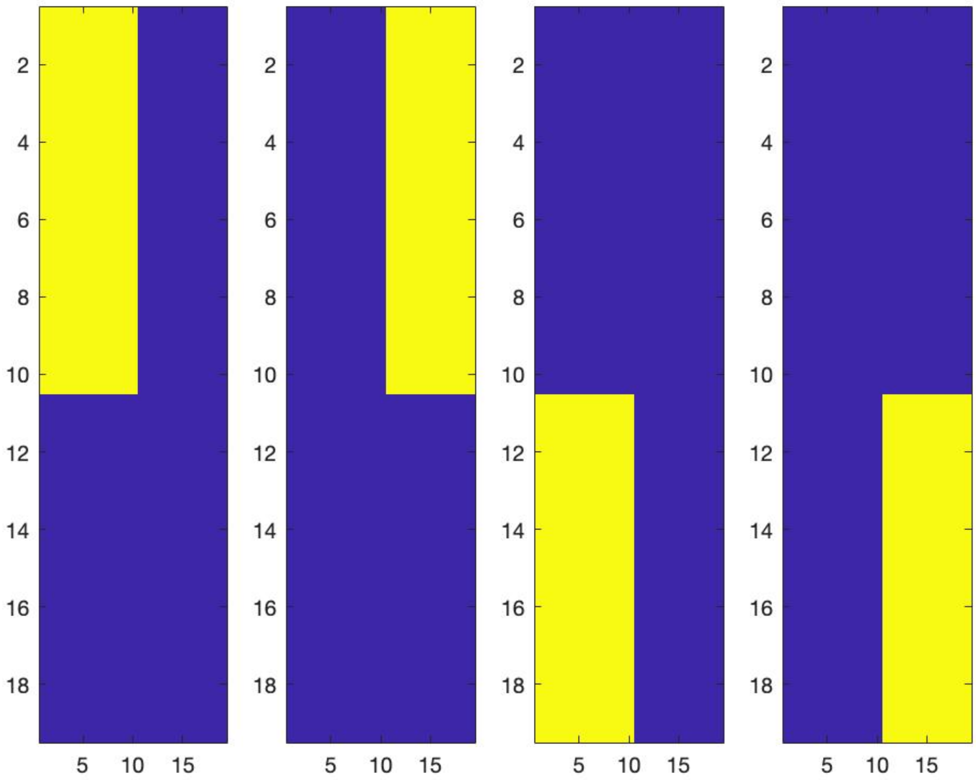

Figure 9: bases 9-12

These bases do not appear to resemble anything so probably won't work well for either set.

	norm 1	norm 2	norm3
Stack 1	712.5799	878.5292	697.0917
Stack 2	701.4410	887.8788	622.3549