University of New Brunswick

Image Processing Course
(EE 6553)

Assignment 4

Professor:
Julian Meng
Electrical and Computer
Engineering

Author:
Saeed Kazemi
Student Number:
3713280

December 22, 2020

Contents

1 Abstract **2**

2 Technical discussion **2**
2.1 The First Question 2
2.2 The Second Question 7

3 Discussion of results **9**

4 Results Images **11**

5 Appendix (codes) **19**
5.1 Code of the first question 19
5.2 Code of the second question 21

1 Abstract

In this assignment, I worked on a MATLAB program for implementing a wavelet transformation as well as Principal Components for an image dataset. This project contains two m-files for solving examples. In section next section, the techniques used for this assignment will be discussed. In section 3, the output results are checked based on images that are shown in part 4, and finally, in the appendix, my codes are added.

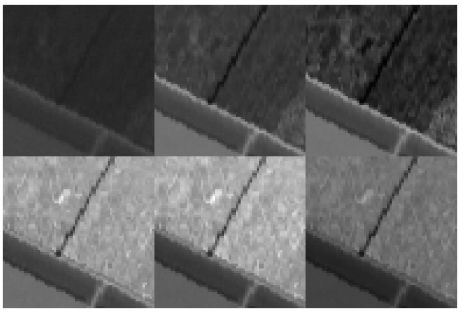
2 Technical discussion

2.1 The First Question

In this section, the techniques were used in this assignment are discussed. For the first question, we implemented Principle Components on a face dataset. This first dataset had 165 face images, and the dimension of each image was 32×32, all grayscale. The second dataset was a multi-spectral image. This dataset is a collection of 224 images that were token on different band. Figure 1 illustrates some images in two datasets. I should mention that since my Matlab is out of date, I could not use the mentioned app for this part, and I work on a part of multi-spectral image.

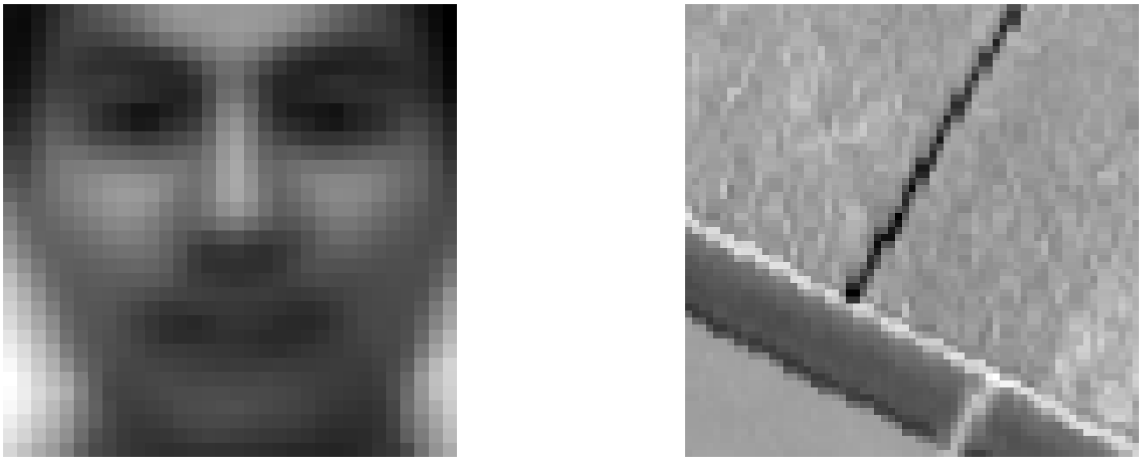
The first step for Principle Component is that all images reshape to a vector. we used `reshape()` in Matlab. In this function, the size of the new shape, as well as the image (or matrix), were passed to the function as input arguments. As a result, each pixel could be considered as a feature. So, our feature space for a $32 * 32$ image is about $32^2 = 1024$. The below matrix indicates our new feature space.

(a) Dataset 1.



(b) Dataset 2.

Figure 1: The sample images form two datasets.



(a) Dataset 1. (b) Dataset 2.

Figure 2: The mean image of all images in two datasets.

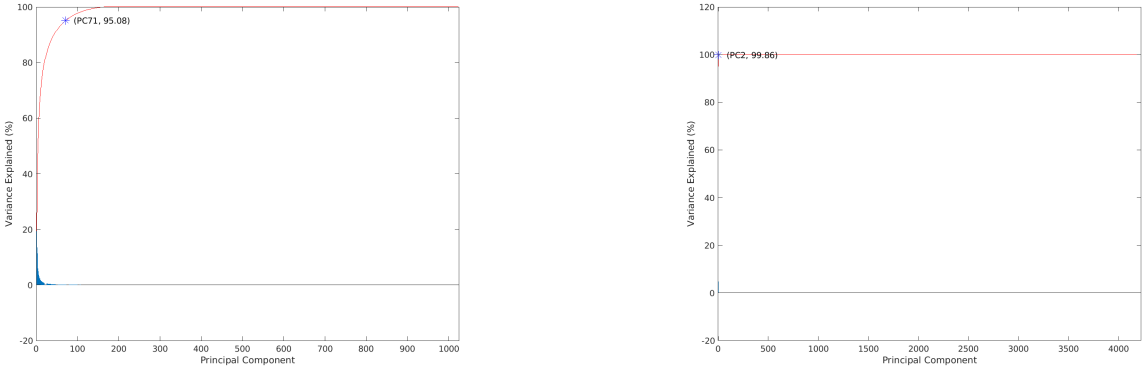
$$new\ feature\ space\ = A = \begin{bmatrix} P_{1,1} & P_{1,2} & \dots & P_{1,N^2} \\ P_{2,1} & P_{2,2} & \dots & P_{2,N^2} \\ P_{3,1} & P_{3,2} & \dots & P_{3,N^2} \\ \vdots & \vdots & \ddots & \vdots \\ P_{M,1} & P_{M,2} & \dots & P_{M,N^2} \end{bmatrix}_{(M \times N^2)}$$

which N and M are 32 and 165 for the first dataset and 64 and 224 for the second one, respectively. But before computing the covariance matrix, we need to normalize matrix A because PCA is so sensitive to the scale of data. Therefore, we have found the vector of mean values by using the below formula. Figure 2 indicates the mean face for our dataset.

$$\vec{mean}(A) = \frac{1}{M} \sum_{i=1}^{M} P_i$$

$$normalized\_A = A - \vec{mean}(A)$$

In this stage, the covariance matrix was needed for our normalized matrix. The covariance matrix was computed by $C = normalized\_A^T * normalized\_A$, and due to the size of A that is $(M \times N^2)$, the size of covariance became $(N^2 \times N^2)$. This matrix has a large size and it has N^2 eigenvalues (λ_i) and eigenvectors (e_i) .



(a) Dataset 1. (b) Dataset 2.

Figure 3: The scree plot of eigenvectors for two datasets.

For computing eigenvalues (λ_i) and eigenvectors (e_i), we used a `eig()` function. This function returned two matrixes, eigenvalues, and eigenvectors. Eigenvector related to the highest eigenvalue contains the highest variance and the one associated with the lowest eigenvalue, the smallest variance.

Due to the eigenvalues were sorted ascending, we need to mirror eigenvalues on the secondary diagonal to sort descending. Also, for eigenvectors, we flipped the matrix in the horizontal direction. Figure 3 depicts the scree plot of each eigenvector for reconstructing the image. Based on this plot, the eigenvectors related to the eigenvalue after 100 did not contain variance, and they could ignore.

95% of variance stored in 71-top eigenvalues for first dataset and for the second one, more than 98% of variance stored in 2-top eigenvalues. For the reconstruction of the original image based on the eigenvectors, we also need to find the weights of each eigenvector. In other words, each image can be expressed as a weighted combination of eigenvectors and mean image. The below equation explains it clearly.

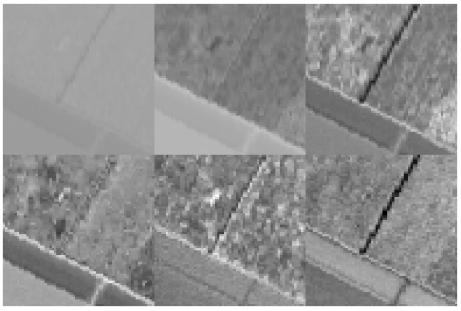
$$image = mean + \sum_{i=1}^{N^2} W_i e_i$$

These coefficients (W_i) show how much that image projects onto the corresponding eigenvector. The equation below shows how to calculate these coefficients.

$$\vec{W} = \vec{e^T} * normalized\_A'$$

Figure 4 shows eigenvectors for the 6-top eigenvalues. Each image focuses on a piece

(a) Dataset 1.



(b) Dataset 2.

Figure 4: The eigenvectors of the 6-top eigenvalues for both datasets.

(a) The Original Image.

(b) The Reconstruction Image.

Figure 5: The original and reconstruction image with 100 PCs for the first dataset.

of information. For example, the coefficient of the second eigenvector shows that the image contains glasses or not (the fifth eigenvectors in figure 4).

Now, we can reconstruct the original image by using the mean and eigenvectors. According to the below equation, we can contribute all of the eigenvectors to reconstruction or a part of it. By changing this part, the quality of the output image changed.

$$reconstructed\ image = mean\ image + \sum_{i=1}^{N^2} W_i e_i$$

Personally, I think that the best limitation for this dataset is 90 PCs. Figure 6

Figure 6: The reconstructed image with different PCs.

displays these reconstructed images with different PCs. As the number of PCs increases the detail of the image would be increased.

Table 1: The result of MSE and SSIM index in reconstructed images by using different PCs.

Image number	PCs number	The MSE index	The SSIM index
Image 1	1 . . . 10	662.1787	0.5446
Image 2	1 . . . 30	342.9688	0.7273
Image 3	1 . . . 50	205.2627	0.8159
Image 4	1 . . . 70	157.9442	0.8511
Image 5	1 . . . 90	95.1956	0.9077
Image 6	1 . . . 110	42.9627	0.9541
Image 7	1 . . . 130	21.2650	0.9765
Image 8	1 . . . 150	05.7734	0.9933
Image 9	1 . . . 170	00.0000	1.0000

2.2 The Second Question

For the second question, we decomposed a high detailed image with wavelet transform. I have chosen the below image from the Digital Image Processing book for this part. This image has nice prominent details in the vertical, horizontal, and diagonal directions. Additionally, I used another picture from Matlab that consists of vertical, horizontal, and diagonal patterns (see Figure 7).

By using the `dwt2()` function, the image decomposed to four images, the horizontal, vertical, and diagonal wavelet coefficients and the approximation coefficients. These images show in figure 9 as an image. While most of the image information is stored in the approximation coefficients, the other coefficients have more detailed information of the original picture (high-frequency information). For this project, we used only a level of decomposition. The cA, cH, cV, and CD stands for approximation, horizontal, vertical, and diagonal wavelet coefficients.

Furthermore, the sym4 was applied for decomposing in this project. Figure 8 illustrates a 1-D shape of this wavelet. Based on the wavelet theory, each mother wavelet must have two properties to use in wavelet transformation. These properties are zero averaging and limitation in time or space.

For composing, the `idwt2()` function were used. we considered several states for this part (see figures 11 to 18).

(a) Gonzalez library (b) Matlab library

Figure 7: The high detailed image.

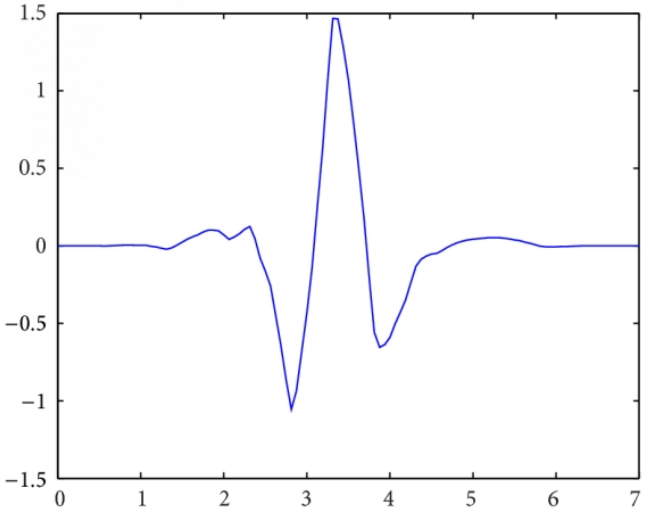
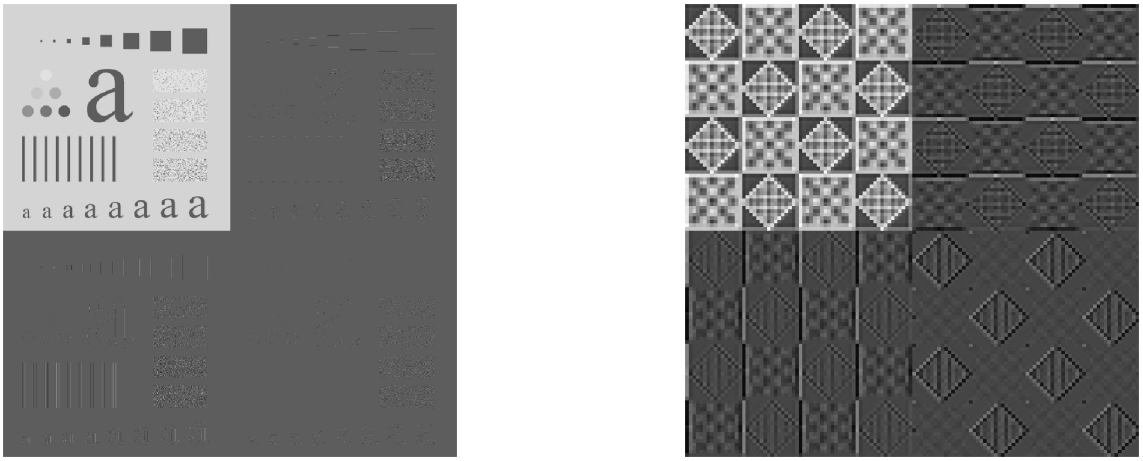


Figure 8: 1-D shape of sym4 wavelet which was used in this project.



(a) Gonzalez library (b) Matlab library

Figure 9: The high detailed images after wavelet decomposition [cA(left-top), cV(right-top), cH(left-bottom) and cD(right-bottom)].

3 Discussion of results

In the next section, nine images were reconstructed based on the different number of PCs. Then, we compared them with the original image. For analyzing these images, we used SSIM and MSE indexes. Table 1 shows two indexes for our images.

As you can see, with increasing the number of principal components, the SSIM index has been increased. This index shows that the similarity of the reconstructed image with the original picture goes up with adding more PCs. For image 9, the SSIM index is 1, which shows two images are similar to each other. In the fifth image, this amount is 0.9, which depicts the reconstructed image look like the original image.

Another index that is used for evaluating the similarity is the MSE index. The MSE index indicates how much two images are different. With increasing the PCs, this index goes down so that we had MSE = 0 for image 9.

Selecting the proper PCs is a hard duty. As you know, in PCA, we always lose the amount of information or variance to decrease our space. So, selecting a good trade off between the losing variance and shrinking feature space is a challenging job.

Personally, I think that the best limitation for this dataset is 90 PCs. Figure 6 displays these reconstruction images with different PCs. for 90 PCs, we are able to see the original image with low detail.

In the second dataset, all of the variance were in the 2-top eigenvectors. figure 10 shows the reconstructed images with different PCs. As it is obvious all images are

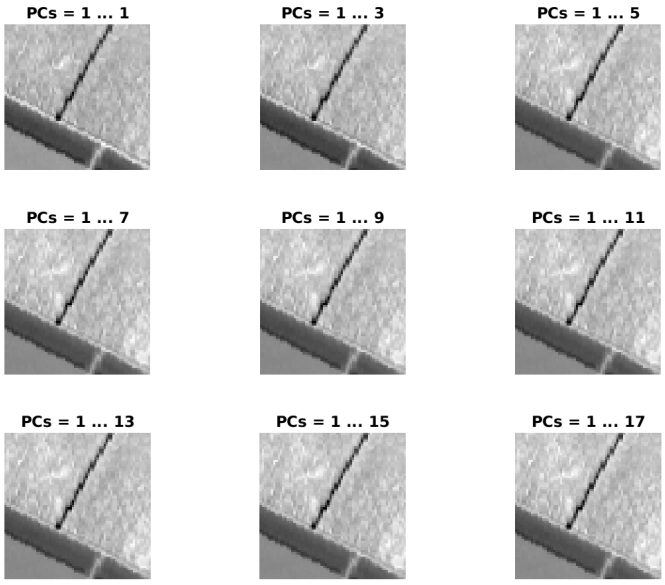


Figure 10: The reconstructed image with different PCs for second dataset.

similar to each other. As a results, we can delete other eigenvectors in the process of reconstruction.

For the second question, different composing was done to see the effect of each coefficient matrixes. In figure 11 and 15, the images were composed with all coefficient matrixes and compared with the original image. As can be seen, both images, composed and original image, are similar to each other.

In figure 12 and 16, the images were composed with only one coefficient matrix. As can be seen, except for image (a), other images show the high-frequency detail in a different direction. Also, the quality of image (a) is not good because the detailed information was eliminated. Therefore, we must add cA because most of the image information is in this matrix. In other words, the approximation coefficient (cA) is necessary for composing.

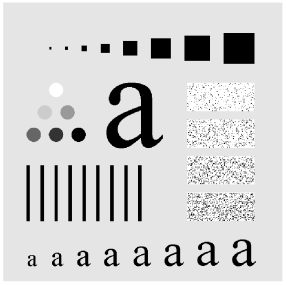
In figure 13 and 17, the images were composed with cA and only one coefficient matrix. As can be seen, image (a) which is used cH as well as cA has good detail in the horizontal direction whereas vertical edges are smoother. As an obvious example, the diamonds in the image (c) which contain edges in diagonal directions, are sharper than squares.

In figure 14 and 18, the images were composed with deleting only one coefficient

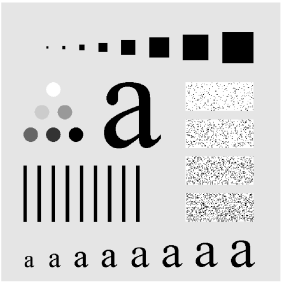
matrix. The reconstructed images have lost the sharpness in the direction of the deleted coefficient. This lack of information appeared in the figure as a blurring effect. For example, the diamonds in the image (a) are less sharper than the squares due to deleting cD matrix in the reconstruction process.

4 Results Images

In this section, other output images are shown.

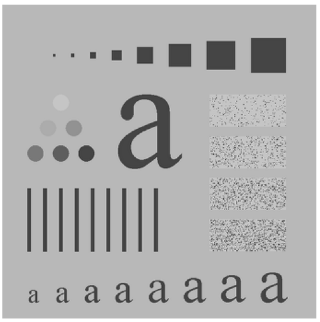


(a) The original image.

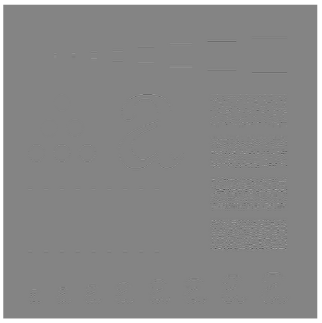


(b) The single-level reconstructed approximation coefficients.

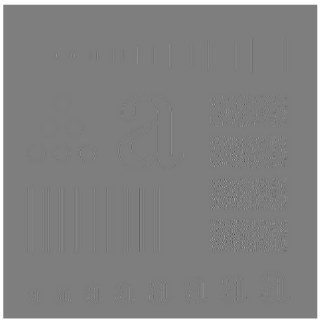
Figure 11: The comparing the original and first level reconstructed image.



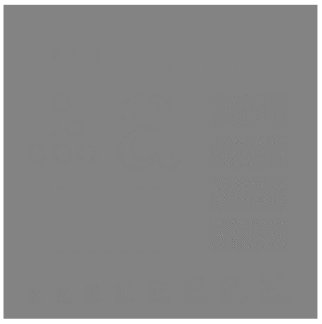
(a) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cA.



(b) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cH.

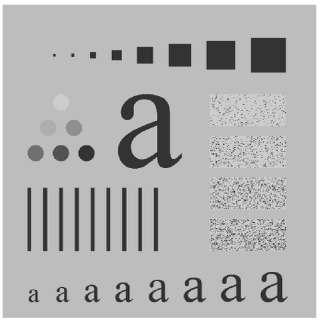


(c) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cV.

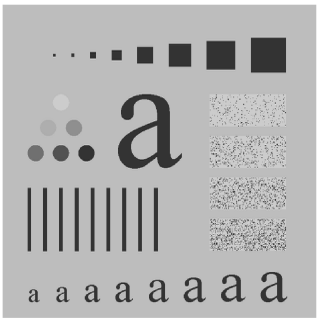


(d) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cD.

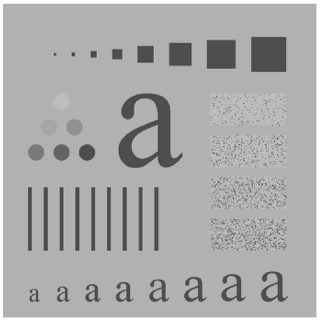
Figure 12: The reconstructed image based on only one approximation coefficients matrix.



(a) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cA and cH.

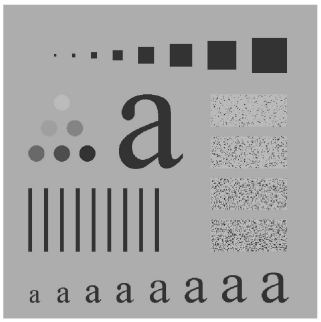


(b) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cA and cV.

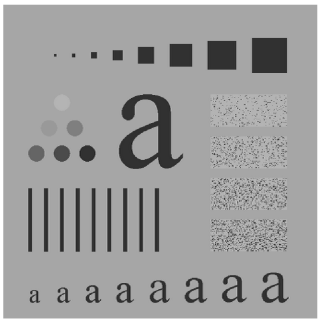


(c) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cA and cD.

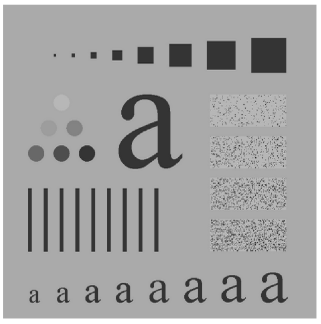
Figure 13: The reconstructed image based on only two approximation coefficients matrixes.



(a) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cA, cV and cH.

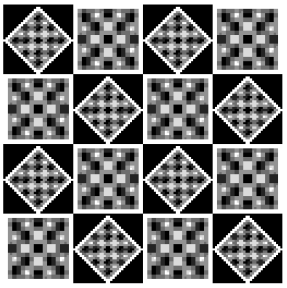


(b) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cA, cH and cD.

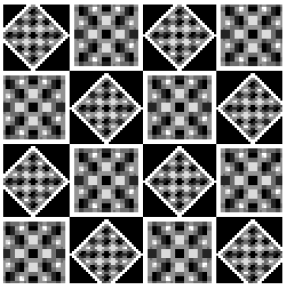


(c) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cA, cV and cD.

Figure 14: The reconstructed image based on only three approximation coefficients matrixes.

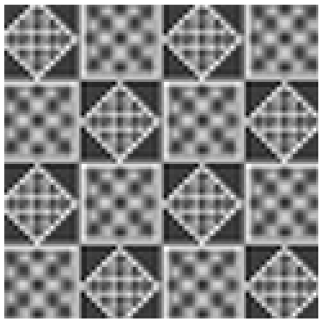


(a) The original image.



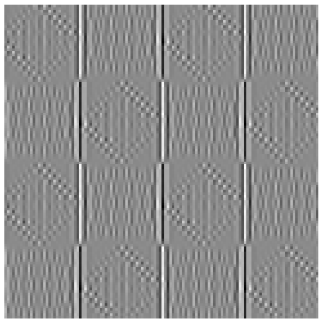
(b) The single-level reconstructed approximation coefficients.

Figure 15: The comparing the original and first level reconstructed image.

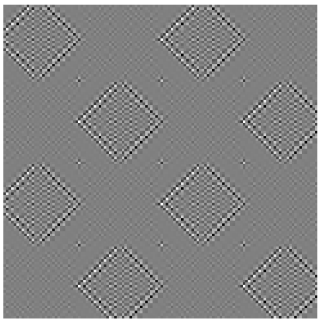


(a) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cA.

(b) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cH.

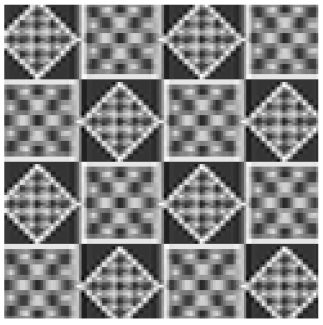


(c) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cV.

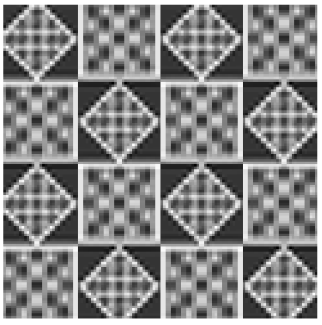


(d) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cD.

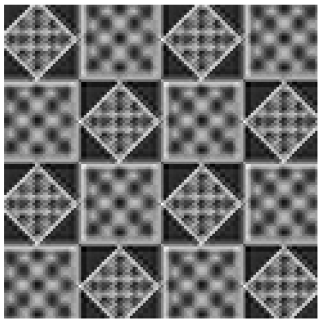
Figure 16: The reconstructed image based on only one approximation coefficients matrix.



(a) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cA and cH.

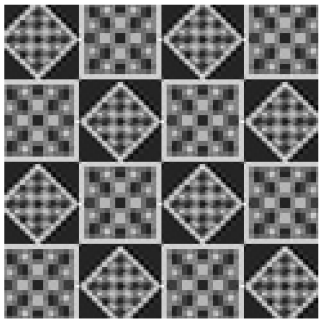


(b) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cA and cV.

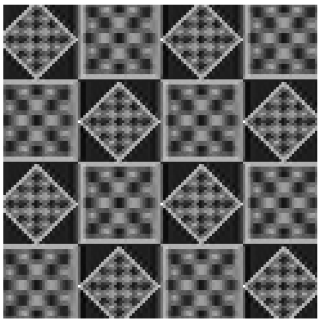


(c) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cA and cD.

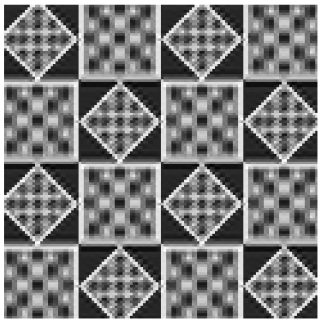
Figure 17: The reconstructed image based on only two approximation coefficients matrixes.



(a) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cA, cV and cH.



(b) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cA, cH and cD.



(c) The single-level reconstructed approximation coefficients output image based on the approximation coefficients matrix cA, cV and cD.

Figure 18: The reconstructed image based on only three approximation coefficients matrixes.

5 Appendix (codes)

5.1 Code of the first question

```
clearvars;
clc
close all


load('EE6553/Lab4/Yale_32x32.mat')
data = fea(:,:);

mean_data = sum(data,1)/size(data, 1);
figure;imshow(reshape(mean_data,32,32),[])
normalized_data = data - (mean_data);

figure;imshow([reshape(data(6,:),32,32),reshape(data(21,:),32,32),...
reshape(data(31,:),32,32); ...
    reshape(data(41,:),32,32),reshape(data(55,:),32,32),...
    reshape(data(66,:),32,32)],[])




A = normalized_data'*normalized_data;
[V,mu] = eig(A);

e = flipdim(V ,2);
lambd = flipdim(flipdim(mu ,1),2);
a1 = reshape(e(:,1),32,32);
a2 = reshape(e(:,2),32,32);
a3 = reshape(e(:,3),32,32);

figure;imshow([a2,a3,reshape(e(:,6),32,32); ...
    reshape(e(:,7),32,32),reshape(e(:,8),32,32),...
    reshape(e(:,9),32,32)],[])


explained = sum(lambd,1)*100/sum(lambd,'all');


sum_explained_a = 0;
idx_a = 0;
while sum_explained_a < 95
    idx_a = idx_a + 1;
    sum_explained_a = sum_explained_a + explained(idx_a);
end
```

```
disp("The dimensionality number that retains 95% of the ...
variance of the dataset is " + num2str(idx_a))


figure;
bar(explained)
xlabel('Principal Component')
ylabel('Variance Explained (%)')

for k = 1 : size(explained,2)
    proportion_of_variance_a(k) = 100*sum(explained(1:k)) ./ ...
        sum(explained);
end
hold on
plot(1:size(explained,2),proportion_of_variance_a , 'r-')
plot(idx_a, proportion_of_variance_a(idx_a),'b*','MarkerSize',10);
text(idx_a, proportion_of_variance_a(idx_a),'   (PC71, 95.08)')




d = 100;%input('please enter the number of PCs: ');
W = e'*normalized_data';
k = 1;%input('please enter the number of image for ...
    reconstruction: ');
aa=mean_data'+e(:,1:d)*W(1:d,k);

figure;imshow(reshape(aa,32,32),[])
figure;imshow(reshape(fea(k,:),32,32),[])
%%
ss=1;
a = reshape(fea(k,:),32,32);
for d = 10:20:170

    aa=mean_data'+e(:,1:d)*W(1:d,k);
    subplot(3,3,ss);imshow(reshape(aa,32,32),[])
    title(['PCs = 1 ... ', num2str(d)])


    ssimval(1,ss) = ssim(reshape(aa,32,32),a);
    err(1,ss) = immse(reshape(aa,32,32),a);



    ss = ss + 1;

end
```

5.2 Code of the second question

```
1
2  clear all
3  close all
4  clc
5
6  wave = 'sym4';
7
8  load tartan;
9  figure;imshow(X, [])
10 saveas(gcf,'/MATLAB Drive/8.png')
11 close(gcf)
12
13 colormap gray
14
15 X = imread('EE6553/Lab4/441.tif');
16 figure;imshow(X)
17 colormap gray
18
19 saveas(gcf,'/MATLAB Drive/6.png')
20 close(gcf)
21
22
23 [cA,cH,cV,cD] = dwt2(X,wave,'mode','per');
24 figure;imshow([cA,cH;cV,cD],[])
25
26
27 xrec = idwt2(cA,[],[],[],'sym4');
28 figure;imshow(xrec,[])
29 saveas(gcf,'/MATLAB Drive/11.png')
30 close(gcf)
31 xrec = idwt2([],cH,[],[],wave);
32 figure;imshow(xrec,[])
33 saveas(gcf,'/MATLAB Drive/12.png')
34 close(gcf)
35 xrec = idwt2([],[],cV,[],wave);
36 figure;imshow(xrec,[])
37 saveas(gcf,'/MATLAB Drive/13.png')
38 close(gcf)
39 xrec = idwt2([],[],[],cD,wave);
40 figure;imshow(xrec,[])
41 saveas(gcf,'/MATLAB Drive/14.png')
42 close(gcf)
43 xrec = idwt2(cA,cH,[],[],wave);
44 figure;imshow(xrec,[])
45 saveas(gcf,'/MATLAB Drive/15.png')
46 close(gcf)
```

```
xrec = idwt2(cA,[],cV,[],wave);
figure;imshow(xrec,[])
saveas(gcf,'/MATLAB Drive/16.png')
close(gcf)
xrec = idwt2(cA,[],[],cD,wave);
figure;imshow(xrec,[])
saveas(gcf,'/MATLAB Drive/17.png')
close(gcf)
xrec = idwt2(cA,cH,cV,[],wave);
figure;imshow(xrec,[])
saveas(gcf,'/MATLAB Drive/18.png')
close(gcf)
xrec = idwt2(cA,cH,[],cD,wave);
figure;imshow(xrec,[])
saveas(gcf,'/MATLAB Drive/19.png')
close(gcf)
xrec = idwt2(cA,[],cV,cD,wave);
figure;imshow(xrec,[])
saveas(gcf,'/MATLAB Drive/20.png')
close(gcf)
xrec = idwt2(cA,cH,cV,cD,wave);
figure;imshow(xrec,[])
saveas(gcf,'/MATLAB Drive/21.png')
close(gcf)
```