Washington University in St. Louis

James McKelvey School of Engineering

MEMS 201 Numerical Methods and Matrix Algebra

Final Project: Application of Principal Component Analysis to Face Recognition

Instructor: Dr. Louis Woodhams

Date of submission: Friday, April 28th, 2023

Alex Webert	Student	[illegible]
Jacob Rapoza	Student	[illegible]

ABSTRACT

The aim of this project was to use Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) in order to ascertain the number of principal components needed to reconstruct an image that is recognizable to humans as well as purview the possibilities of using PCA and SVD for goals such as distinguishing whether a face is human or identifying facial expressions. This was done by creating a Matlab code that can evaluate grayscale images' pixel density, create a mean center difference, perform SVD, and then reconstruct the image using a variable number of principal components. It was found that 15 principal components allow an individual to be clearly recognized, but that 22 components are needed to clearly distinguish facial expressions. It was found that for a small data set, it is possible to use PCA and SVD in order to reasonably accurately tell if a face is human or not.

INTRODUCTION

A person's face is an integral part of an individual's identity. Recognizing someone's face and the emotions that they portray is key for social interaction. In this paper, we seek to create a computer algorithm capable of determining the minimum number of principal components necessary to pick up on facial expressions and the recognition of an individual.

A computer algorithm capable of such feats can contribute to practical applications including criminal identification, security systems, image and film processing, and human-computer interactions [1]. However, creating such an algorithm does not come without challenges. The human face is complex and multidimensional, so simplifying the problem to a 2-D approach is optimal. Using an image set of ”straight-on” views, the human face can be recognized by a much smaller set of the most relevant 2-D characteristic features. This method was explored by Turk and Pentland in the paper ”Eigenfaces for Recognition” [1]. By treating each image as a set of vectors making up a person's face, they were able to determine the principal components of the distribution of faces in a data set. These are the eigenvectors that when combined can characterize the variation between human faces. Each image in a data set contributes a certain amount to a given eigenvector, and when combined creates an eigenface. Each individual face can be reconstructed exactly by using a linear combination of eigenfaces. By using more or fewer eigenfaces in a linear combination, the reconstructed image may resemble a human face more or less.

Our goal is to expand on this work by creating a working algorithm to determine the minimum number of principal components needed to reconstruct a human face that is recognizable. Using Matlab, we will analyze an image set of ”straight-on” faces using Principal Component Analysis (PCA). This requires image processing to convert each image into a set of vectors that can then be described by eigenvectors, eigenfaces, and a set of linear coefficients. By recombining these three terms at a set number of components, the reconstructed image will have a certain resemblance to a human face. In addition to this goal, we wanted to extend our work to figure out if it would be possible to identify an image as a human face. This requires the use of PCA to look at the importance of each eigenface in reconstruction and compare it to a built-in threshold of known values from our data set.

METHODS

In order to achieve Aim 1 of the final project we had to reduce the dimensionality of images of faces using Principal Component Analysis (PCA) and then reconstruct the images using a set number of principal components. This was done through a Matlab code following the workflow seen in Fig. 1 below.

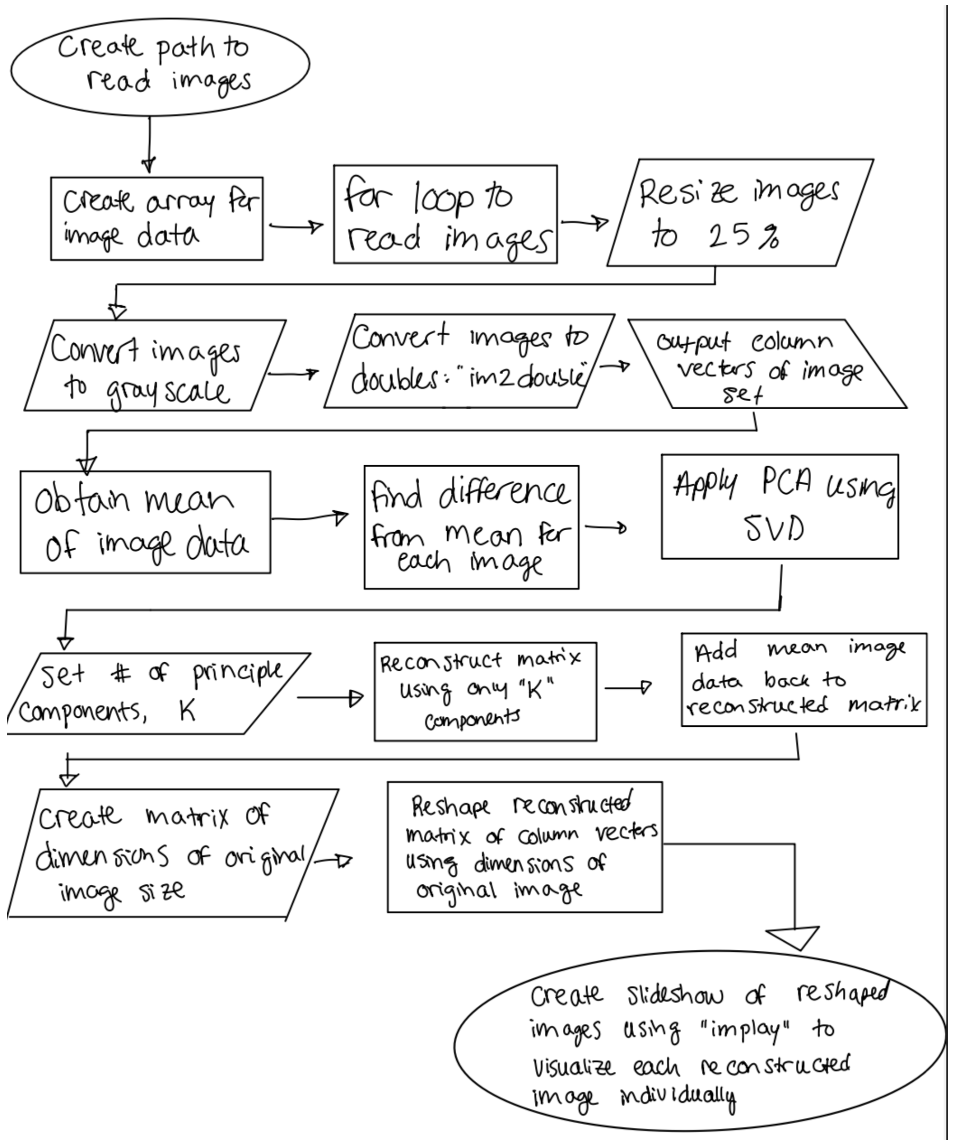

Create path to read images
Create array for image data
for loop to read images
Resize images to 25%
Convert images to gray scale
Convert images to doubles: "im2double"
output column vectors of image set
Obtain mean of image data
Find difference from mean for each image
Apply PCA using SVD
Set # of principle Components, K
Reconstruct matrix using only "K" components
Add mean image data back to reconstructed matrix
Create matrix of dimensions of original image size
Reshape reconstructed matrix of column vectors using dimensions of original image
Create slideshow of reshaped images using "implay" to visualize each reconstructed image individually

Figure 1 Flow chart of Matlab code for Aim 1

To begin the analysis, we first had to pull our image data set into Matlab. This was done by creating a path to the folder in which the pictures were stored. Once the path was set, we then had to read each image into Matlab and store the data. This was done by creating an empty array and then performing a series of commands in a loop for each image in the path folder. The steps that were performed on each image in the loop were as follows. First, the image was read into Matlab using imread(). Second, the size of the image was reduced to 25% of the original size using imresize(). Third, the images were converted to grayscale using im2gray(). Fourth, the class of the image was changed from a uint8 to a double using im2double(). Fifth, each image was converted into a column vector and horizontally concatenated into our earlier array. The data was converted to grayscale to reduce the variables so that the values are only combinations/percentages of black and white. The image size was reduced to minimize the processing power required for later steps. The class was changed because uint8 would not be compatible with the variable class used for the mean of the image of the data.

Once the for loop ran through every image in our data set, we were left with a single matrix of size 68,026 x 54. This represents 68,026 data points for each of the 54 images in our data set. Before applying PCA and making use of the Singular Value Decomposition(SVD) function, we had to mean-center the data. This was done by calculating the mean of each row for the image data array we had created [2]. Once we had found the mean column vector, in order to find the difference from the mean we performed simple subtraction between our original image data and the mean values.

Once we had the mean center data, we were able to apply PCA using the SVD. When applying SVD to the data, "econ" was used to produce an economy-size decomposition of our data. The resulting outputs were U, S, and V. U defines the left singular vectors, returned as the columns of a matrix representing the eigenfaces of our data [2]. S defines the singular values returned as a diagonal matrix, representing the significance of each eigenface in the reconstruction of a face [2]. V defines the right singular vectors returned as the columns of a matrix representing the linear coefficients needed to reconstruct the faces [2].

To begin the reconstruction of the images, a variable "k" was created to set the number of principal components that would be used in the reconstruction of the image. To put the left and right singular vectors and singular values back into a readable matrix, we then used a component form of Equation 1 which can be seen below in Equation 2

$$A = U * S * V' \tag{1}$$

$$A = U(:, 1 : k) * S(1 : k, 1 : k) * V(:, 1 : k)' \tag{2}$$

where A is the reconstructed double [3]. This step where we control the number of principal components to be used in the reconstruction is the PCA analysis.

Before reshaping A to visualize the images, the mean of the image was added back to A so that the values correlate to the difference plus the original offset. After the mean is added back, images are then reconstructed using the reshape function in Matlab where the first two dimensions are the known height and width dimensions from the original image and the last dimension is free to adjust to whatever dimension is needed by the other two. The final step to view the reconstructed images is to create a slideshow using implay().

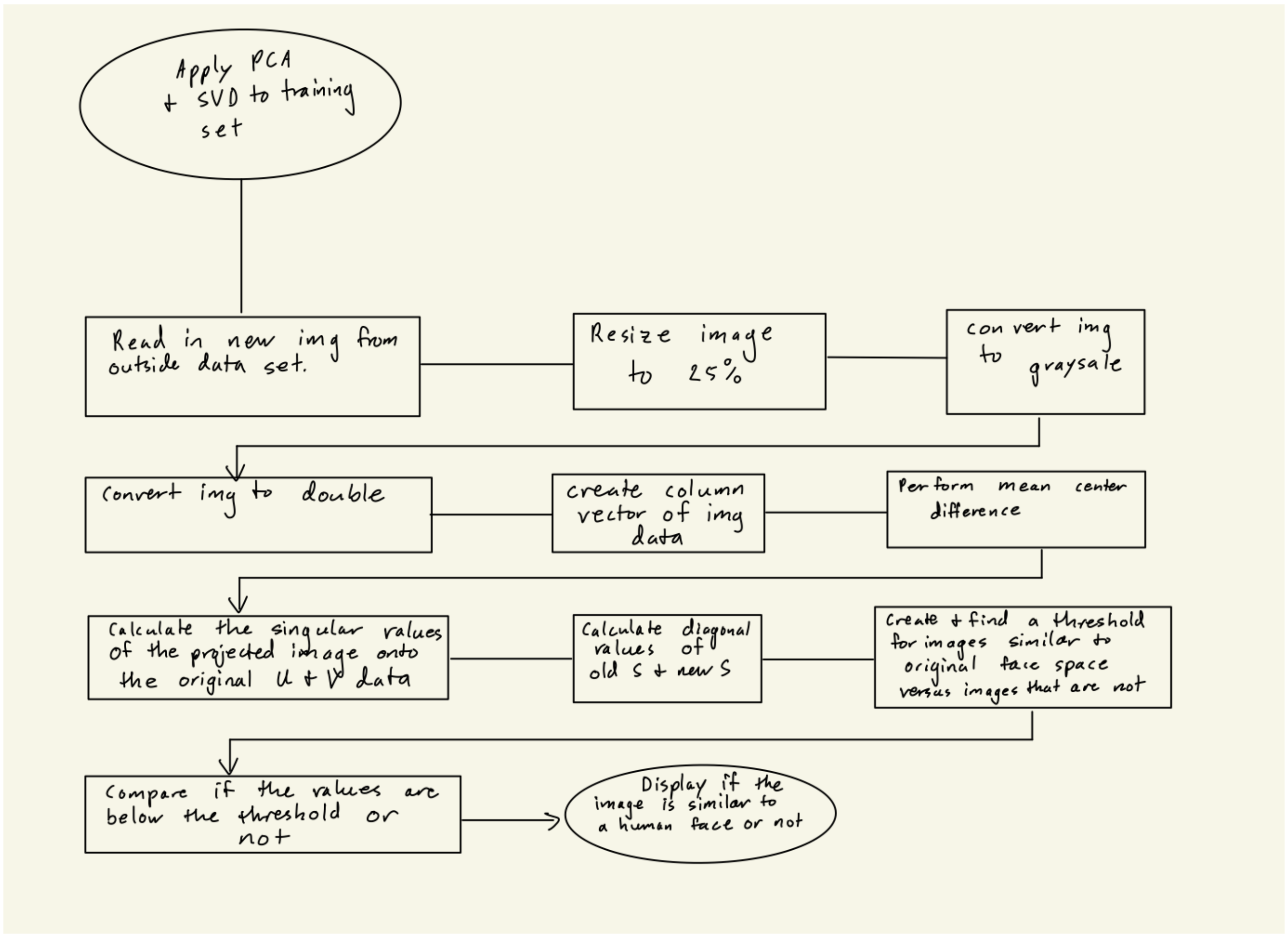
The code used to achieve aim 1 can be seen below in Fig. 4.

```
%Final Project Code
clear
close all

% reading in an images
path= 'C:\Users\bocaj\Documents\WASHU Spring 2023\Numerical Methods\Final ↙
Project\data1Cropped\';
img_file= dir(fullfile(path,'*.jpg'));

%create array for image data
img_data=[];

%use a for loop to loop to read through all the images and convert them to grayscale ↙
and to colum
%vectors (note that you must change the class of img from uint8 to a double
%in order to do the mean centering part)
for i=1:length(img_file)
    img=imread(fullfile(path,img_file(i).name));
    img=imresize(img,.25);
    img=im2gray(img);
    img=im2double(img);
    img_data=[img_data, img(:)];
end


%mean centering the data
mean_img= mean(img_data, 2); %get the mean of the img_data
mean_center_data = minus(img_data,mean_img); %calculate difference from the mean

%Apply PCA using SVD
[U,S,V]=svd(mean_center_data, "econ");
V=V'; %transposing the V values
%U is the left singular vector (eigenfaces)
%S is the singular values that tell how important each egienface is in
%reconstructing the faces in "face space"
%V is the right singular vector that gives the linear coefficents needed to
%reconstruct the faces

k=15; %the number of principal components we want to use
A=U(:,1:k)*S(1:k,1:k)*V(1:k,:); %putting back together the principal
%components using only k components

A=plus(A,mean_img); %Adding back the mean

[height, width]=size(img); %use the size(img) function to figure out the height and ↙
width of images

reshape_face= reshape(A,height,width,[]); %reshaping the colum vectors back
implay(reshape_face); %creating a slideshow of the reshaped faces
```

Figure 2 Matlab code for Aim 1

In addition to Aim 1, an attempt at Aim 2 was undertaken through the steps seen below in the Fig.

3 flow chart based on the work of Sandipan Dey in [4]. Note that this flow chart starts assuming the code for Aim 1 has been completed.



Apply PCA + SVD to training set
Read in new img from outside data set.
Resize image to 25%
convert img to graysale
Convert img to double
create column vector of img data
Perform mean center difference
Calculate the singular values of the projected image onto the original U + V data
Calculate diagonal values of old S + new S
Create + find a threshold for images similar to original face space versus images that are not
Compare if the values are below the threshold or not
Display if the image is similar to a human face or not

Figure 3 Flow chart of Matlab code for Aim 2

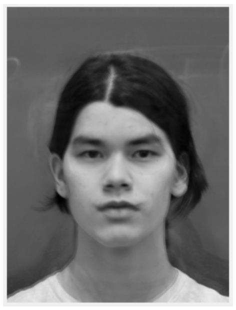
To begin the analysis for aim 2, a new image of an animal or human face that was not part of the original training data set was read in. Then the same steps that were carried out on the original training data set were completed so that we had a grayscale, resized, column vector of the data for the new image. Once the image was in column vector form it was mean centered based on the mean that was calculated for the original training data. After this, a new S_2 singular vector was created using a rearranging of Equation 1 based on the projection of the new image onto the face space of the original training data. Then both of the S diagonal matrices were converted to column vectors. The human-calculated threshold was created based on comparing the values from face images and animal images to find a relative cut-off point. Finally, the threshold was used in an if else statement to decide if the projected S values were close enough to the original ones to warrant the image being a face or not.

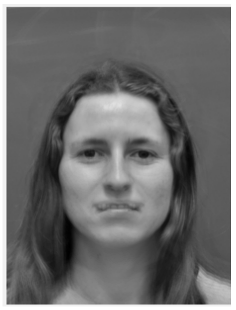
```
% Part 2: Analysis of a single image and whether or not it is likely a
% human face
path_2 = 'C:\Users\bocaj\Documents\WASHU Spring 2023\Numerical Methods\Final↙
Project\data_3\img013.jpg'
new_img = imread(path_2); % replace with the path to your new image
new_img = imresize(new_img,.25);
new_img = im2gray(new_img);
new_img = im2double(new_img);
new_img_data=new_img(:);
new_img_data=new_img_data-mean_img;% completing the same steps from aim 1 code
V=V';%transposing V back to its original

projection= U(:,1:k)'*new_img_data*V(1,1:k); %calculating S using the new images↙
column vector data

S_2=diag(projection)% getting the diagonal new S values

S_1=diag(S)%getting the diagonal original S values
threshold =.046; %user defined threshold found by trial and error

if max(abs(S_2)) < max(S_1)*threshold %code to decide and display if the face is↙
human or not
    disp("This is a human face.");
else
    disp("This is most likely not a human face");
end
imshow(path_2);
```

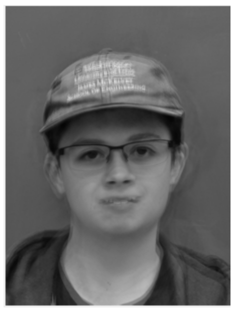
Figure 4 Matlab code for Aim 2

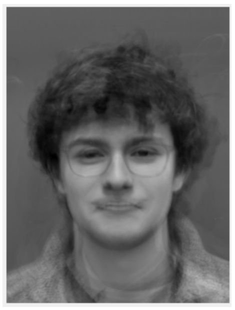
RESULTS

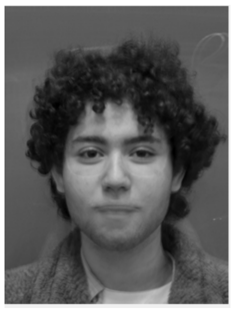
For Aim 1 it was determined through inspection, that 15 is the minimum number of principal components to identify an image as a person. Shown below in Fig. 5 are six example images of facial reconstruction using 15 principal components.


(a) Image 1.


(b) Image 2.


(c) Image 3.


(d) Image 4.


(e) Image 5.

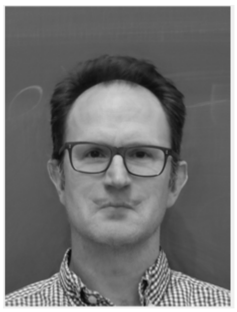
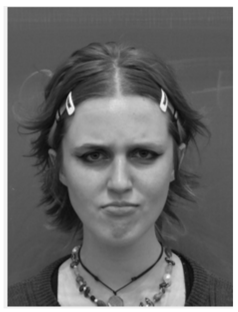
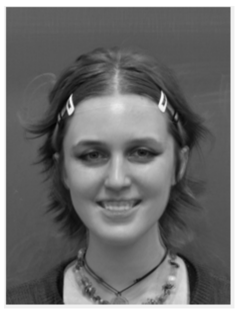
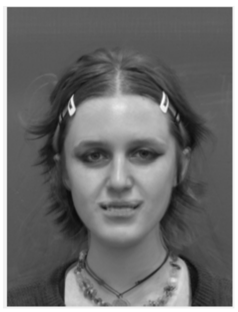

(f) Image 6.

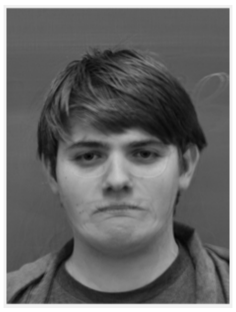
Figure 5 Example images using 15 principal components.

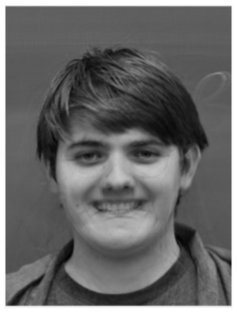
As shown in the figure above it is clear that 15 principal components allow an individual to be recognized in an image. However, while an individual may be recognized, the facial expression they are portraying is less clear at this level. By upping the number of principal components to 22, you may begin to recognize facial expressions, as seen below in Fig. 6.


(a) Frowning.


(b) Smiling.


(c) Straight face.


(d) Frowning.


(e) Smiling.

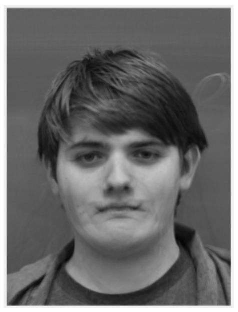
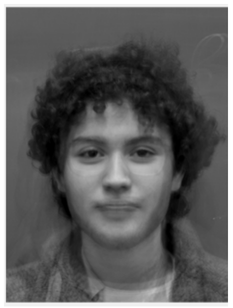
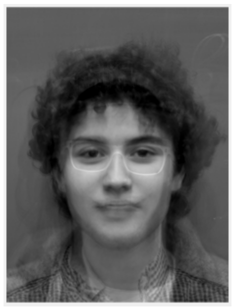

(f) Straight face.

Figure 6 Example of facial expressions using 22 principal components.

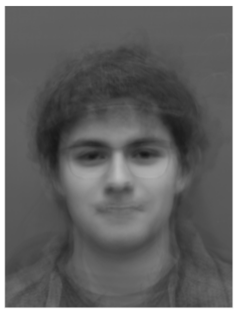
The figure above shows that 22 principal components not only allow you to recognize an individual but also the facial expressions that they make. Including more principal components in the image, reconstruction would allow for even clearer images. On the contrary, by removing principal components it can be even more difficult to recognize an individual from image reconstruction. As demonstrated below in Fig. 7, fewer principal components present in an image reconstruction lead to worse image quality. Going from left to right, we see 3 individuals portrayed at 10 principal components, then 5, then 2. With only 2 principal components present, the 3 images look very similar even though they are supposed to be portraying different individuals. An interesting discovery is that at the lowest levels of principal components, gender becomes obsolete as well.



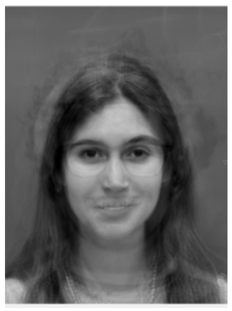
(a) 10 principal components.



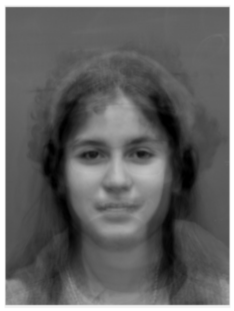
(b) 5 principal components.



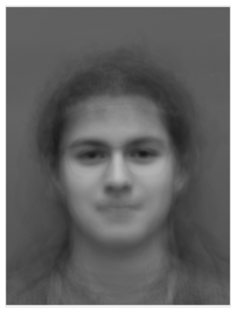
(c) 2 principal components.



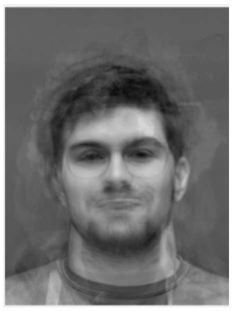
(d) 10 principal components.



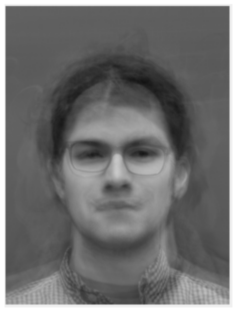
(e) 5 principal components.



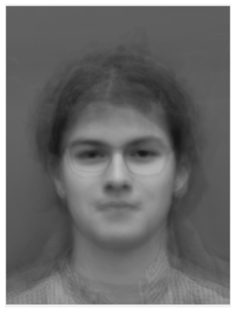
(f) 2 principal components.



(g) 10 principal components.



(h) 5 principal components.



(i) 2 principal components.

Figure 7 Example images using 10, 5, and 2 principal components.

Our findings above are confirmed by a plot of the singular values. The graph in Fig. 8 shows how the significance of each singular value varies. We can see that for smaller singular values, the importance of the eigenface is large. As the singular values get larger, the importance of the eigenface decreases until at around 22, the significance stabilizes and levels out. This tells us that any additional eigenfaces after 22 are not necessary to reconstruct an image. Additionally, it tells us that if we leave out eigenfaces before k=22, we will lose important details in the image such as facial expressions which makes the possibility of using PCA to detect emotion in facial expressions harder.

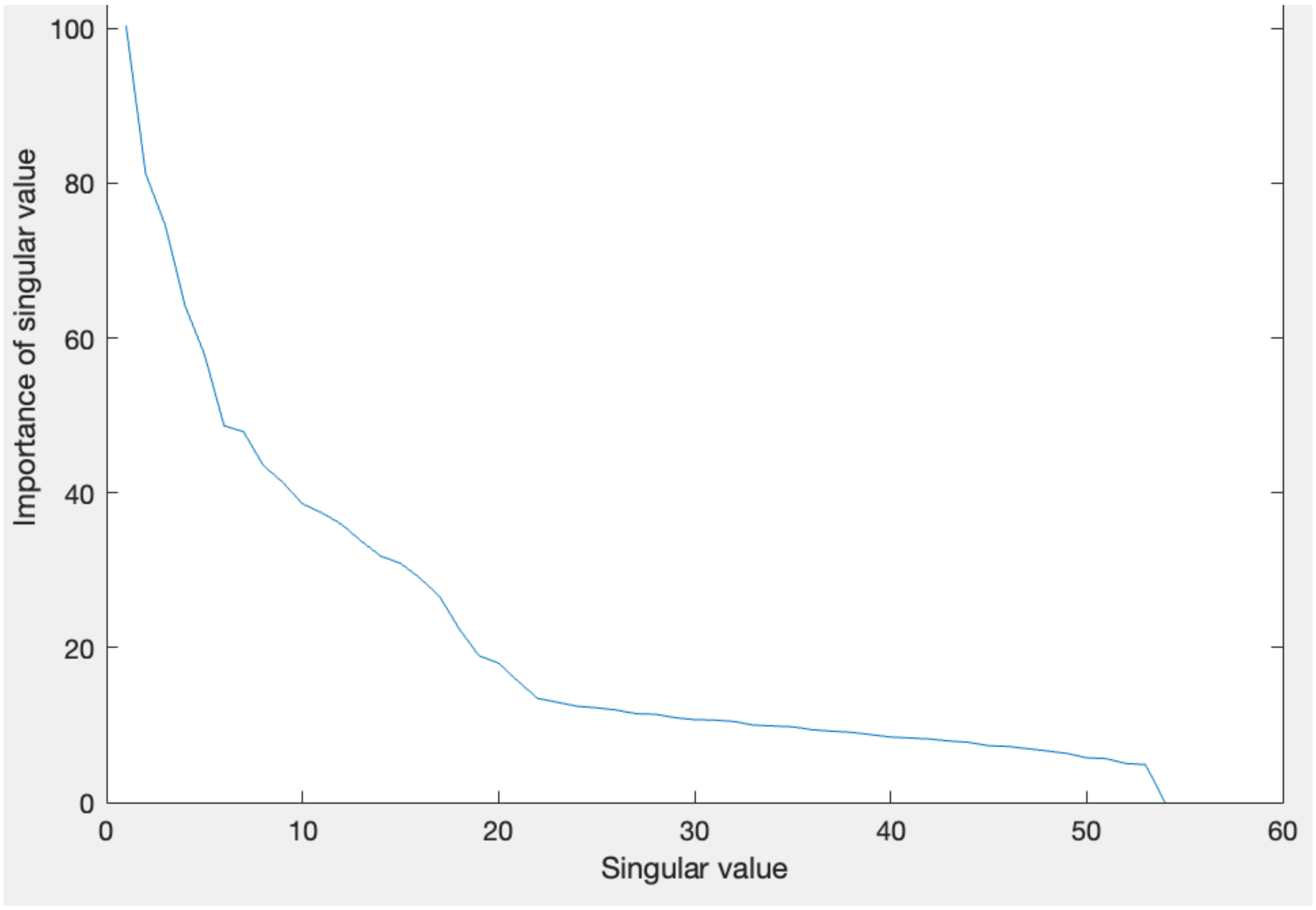
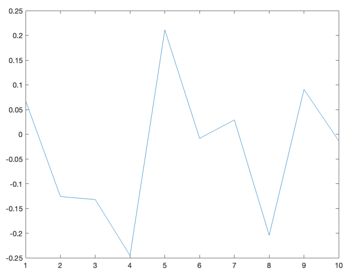


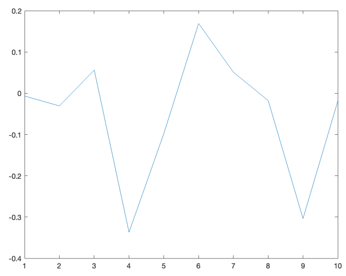
Figure 8 Plot of singular values

In Fig. 10 below, we compare images to plots of the first 10 right singular vectors that make up the reconstruction of the image. The right singular vectors, which designate the linear coefficients applied to reconstruct an image, differ between individuals which proves that each person is a unique image.



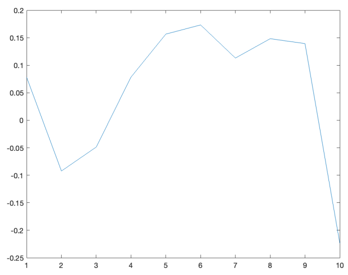
(a) Image 13 right singular vectors.

(b) Image 13.



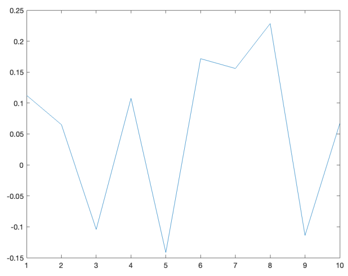
(c) Image 21 right singular vectors.

(d) Image 21.

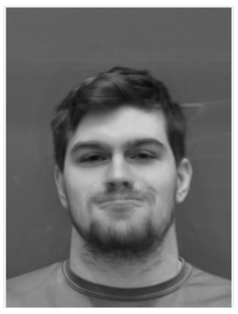


(e) Image 28 right singular vectors.

(f) Image 28.



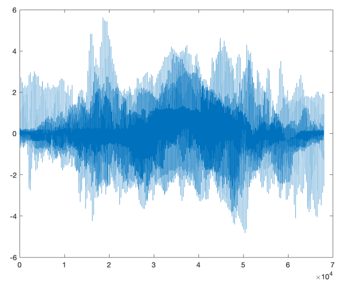
(g) Image 33 right singular vectors.



(h) Image 33

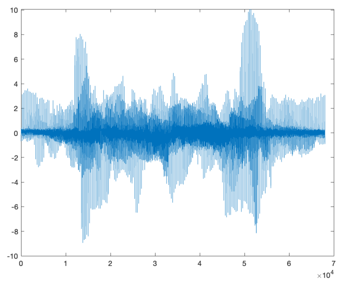
Figure 9 Example images and the first 10 right singular vectors that make them up

Finally, shown below are the same images as above next to plots of the eigenfaces used to reconstruct the images. As you can see, the data is centered around zero with bounds being 1 standard deviation.



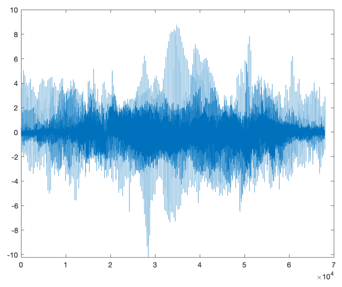
(a) Image 13 eigenfaces.

(b) Image 13.



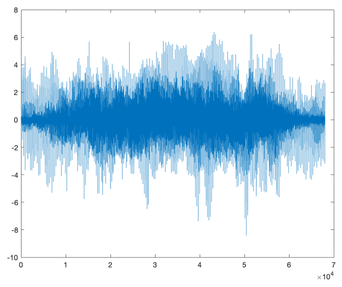
(c) Image 21 eigenfaces.

(d) Image 21.

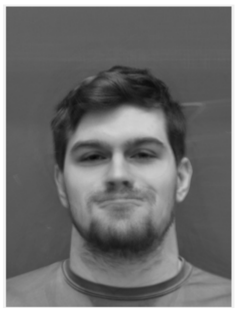


(e) Image 28 eigenfaces.

(f) Image 28.



(g) Image 33 eigenfaces.



(h) Image 33

Figure 10 Example images and the first 10 right singular vectors that make them up

For Aim 2, it was determined that a reasonably accurate threshold for the given training data set and comparative data sets was .046 assuming the use of all 54 principal components for the calculation of the new S values. The code for Aim 2 did perform relatively well as it was able to identify 5 animal faces as nonhuman and 5 human faces, not from the original training data, as human faces.

DISCUSSION

An important takeaway from this project is that it requires 15 principal components to reconstruct a recognizable human face but that it requires 22 components in order to tell important facial expressions such as smiling or frowning. Going along with this idea it is important to realize that with fewer than 22 principal components important details are left out of the reconstruction but with all 54 components absolutely nothing is left out and the reconstructed image should be the same as the original image. Additionally, it is possible to use the SVD and PCA in order to relatively accurately determine whether an image is of a human face or not.

A possible application of this work is for systems that need to minimize data space and computation power. This could be done by only using the important principal components to reconstruct images so that images are relatively clear but require less data. Another possible application of this work is for analysis of projects/systems where there is a lot of noise at higher order principal components and someone only wants to look at the important data. If that's the case they can just use the lower order principal components to get a reconstruction with the important details but not the extraneous ones [3]. Aim 2's results could be used to help with facial recognition or face sorting software.

While we are able to draw strong conclusions on the number of principal components needed to recognize a human face and facial expressions, there are still limitations and sources of error to be discussed. One limitation is the sample size and the number of principal components we have access to. With a larger data set, we could look at more principal components. Another limitation is that while our current model can predict the number of principal components to identify a face, it does not have any way to classify who a face is. Because our initial results were based on inspection, there is certainly some degree of error behind the number of principal components needed to identify a face and facial expression. While the plot of singular values, S, is assuring, it still does not tell us the exact number of principal components needed to identify a face in a reconstructed image. In the case of Aim 2, the results although useful are not done on a large enough data set to be considered conclusive. Likewise, the code for Aim 2 requires a human to mess around a bit in order to find the sweet spot where the projected image is considered close to the original data's face space or not which makes it somewhat clunky.

In future work, it might be worth exploring the possibility of identifying which person a reconstructed image is of. While it would take a lot of time to create an accurate algorithm, a potential lead may lie in the right singular vectors. As we found out, the right singular vectors are unique to each individual. With a large enough data set and a large number of principal components, an algorithm may be created to match the right singular vectors to a certain person. It would also be interesting to look into whether or not it is possible to accurately tell whether an image is of a human face using larger data sets. While it would take serious tweaking and testing for larger data sets to find an accurate threshold of error, it is definitely possible to reconstruct non-human-face images using the ”face space” created from the original human face space and look at the large residual error that results as an indicator that the image does not appear to be of a face [4].

CONCLUSION

The important takeaways from this project are as follows. Based on the original training data set given to MEMS 201 students, a recognizable image reconstruction of a human face requires 15 principal components. In order to definitively tell the emotional expression of a reconstructed image, 22 principal components are required. It is possible to project an image onto the face space and tell with some accuracy whether the S singular values of the projected image are close enough in importance to those of the original training data and thus decide if an image is of a face or not.

CONTRIBUTIONS

In terms of author contributions, Jacob wrote the majority of the code, abstract, and conclusion. Alex wrote the majority of the results, and introduction. Both Alex and Jacob worked on the methods section, discussion, and did research into the SVD.

References

[1] Turk M., P. A., 1991, “Eigenfaces for Recognition,” Journal of Cognative Neuroscience.

[2] Woodhams, L., “Application of Principal Component Analysis to face recognition,” final project.

[3] Woodhams, L., 2017, “Application of PCA to Cardiac Optical Mapping,” Master’s thesis, Washington University in St. Louis, St. Louis, MO.

[4] Dey, S., 2018, “EigenFaces and A Simple Face Detector with PCA/SVD in Python,” Sandipanweb, accessed Apr. 25, 2023, https://sandipanweb.wordpress.com/2018/01/06/eigenfaces-and-a-simple-face-detector-with-pca-svd-in-python/