Table of Contents

Introduction 1
BEGIN CODE 2
Load in the Cropped face database 2
Calculate the "average" face from all photos 3
Computing the SVD 3
Describing the Compact SVD 4
project each image onto basis 5
Explore the rank of r (see how many modes we really need) 6
Creating and exploring faces 8
The PCA process 9
Load in the uncropped face database 18
Calculate the "average" face from all photos 19
Computing the SVD 20
Explore the rank of r (see how many modes we really need) 20
Creating and exploring faces 21
The PCA process 21
Discussing PCA with the uncropped faces 26

Introduction

```
%{
Written by Ian Good on 10/20/2020. For questions, please reach me at:
iangood@uw.edu


The goal of this file is to explore singular value decomposition (SVD)
 and principal component analysis (PCA).
    SVD and PCA are incredibly powerful tools for a data scientist and
 a good
    understanding of their uses and limitations will pay dividends
 over one's career.
    SVD is often used in data reduction where the original data is
 transformed in a way
    to need significantly fewer dimensions to characterize. This is
 incredibly helpful in machine learning
    where the dimensionality of data is often enormous.
    PCA is often used to identify the variance and covariance of a
 datset while also serving as a data reduction method.
    By understanding the structure of the data you are working with,
 you may be able to extract incredibly valuable features,
    allowing for interpretations which may have been missed.
    Note: This work covers and uses the compact SVD.

Two datasets were sourced for this project, both from the Yale Faces
Project. A link to both can be found below:
https://drive.google.com/drive/
folders/1SQ77P5t5RUWCSucmk4jPFbufFMX8VrJG
```

```
%}
```

BEGIN CODE

```
%Clear Previous Workspace and Figures

clear all
close all
clc
```

Load in the Cropped face database

```
tic
% Specify the folder where the files live.
myFolder = 'C:\Users\PhD\Documents\MATLAB\ME 564\faces\CroppedYale\';
  %CroppedYale\yaleB31   or yalefaces\
% Check to make sure that folder actually exists.  Warn user if it
 doesn't.
if ~isfolder(myFolder)
    errorMessage = sprintf('Error: The following folder does not
 exist:\n%s\nPlease specify a new folder.', myFolder);
    uiwait(wfaceSetdlg(errorMessage));
    myFolder = uigetdir(); % Ask for a new one.
    if myFolder == 0
         % User clicked Cancel
         return;
    end
end
% Get a list of all files in the folder with the desired file name
 pattern.
filePattern = fullfile(myFolder, '**/*.pgm'); %search for whatever you
 need. The **/ searches subfolders as well
theFiles = dir(filePattern);
N = length(theFiles);


% Size of each picture in the cropped dataset
m = 192;
n = 168;

avg = zeros(m*n,1);  % start the average face array
A = [];
count = 0;


for k = 1 : length(theFiles)
    baseFileName = theFiles(k).name;
    fullFileName = fullfile(theFiles(k).folder, baseFileName);
    imageArray = imread(fullFileName);
    %figure
    %imshow(imageArray);  % Display image.
    %drawnow; % Force display to update immediately.
```

```
    if(size(imageArray,3)==1)
        M=double(imageArray);
    else
        M=double(rgb2gray(imageArray));
    end
    %pause(0.01);
    R = reshape(M,m*n,1);
    A = [A,R];
   avg = avg + R;
   count = count + 1;
end
avg = avg /count;
```

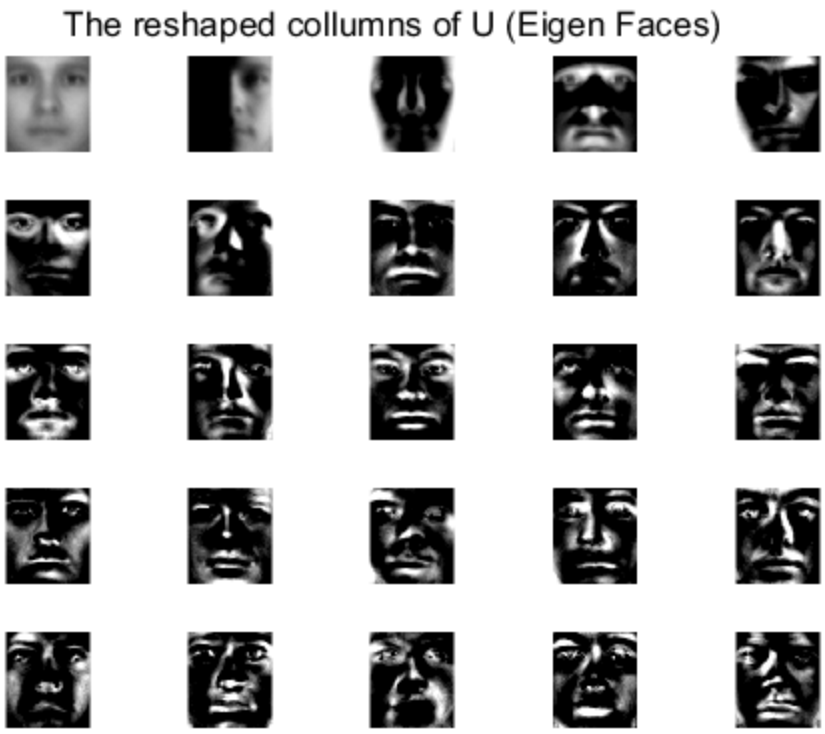
Calculate the "average" face from all photos

```
avgTS = uint8(reshape(avg,m,n));
figure(1), imshow(avgTS);
sgtitle('Average Face');

%{
The average face was taken from all photos in the uncropped dataset.
 It
represents 2414 photos from the Yale Faces Dataset.
%}
```

Computing the SVD

```
[U,S,V] = svd(A,0);
Phi = U(:,1:N);
Phi(:,1) = -1*Phi(:,1);
figure()
count = 1;
for iter=1:5
```

```
    for j=1:5
        subplot(5,5,count)
        imshow(uint8(25000*reshape(Phi(:,count),m,n)));
        count = count + 1;
    end
end
sgtitle('The reshaped collumns of U (Eigen Faces)');
```



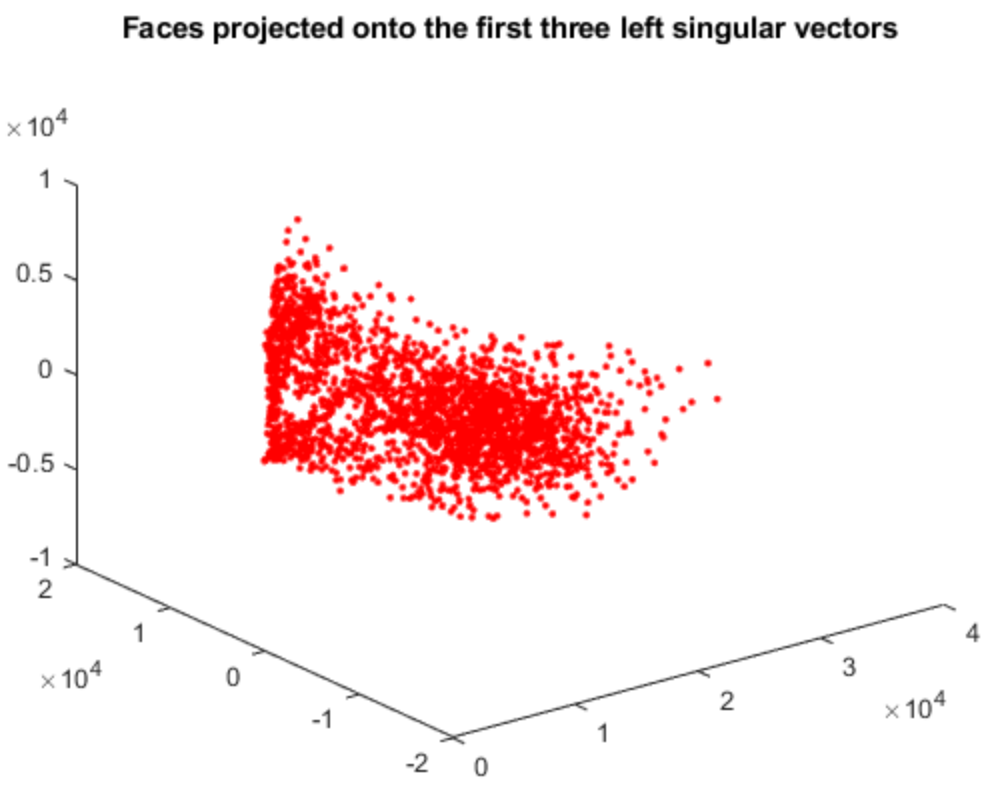
Describing the Compact SVD

```
%{
The SVD consists of three output matrices U,S, and V*.
Given any complex matrix A (m x n) then…
U is an m x n unitary matrix which holds the left singular vectors of
 A
S is an n x n diagonal matrix with the singular vales of A along the
 diagonal arranged from highest to lowest with zeros everywhere else.
V is an n x n unitary matrix which contains the right singular vectors
 of A

Since both U and V are unitary matrices, they form an orthonormal
 basis and can be regarded as basis vectors.
For any linear transform T within the complex numbers of size n x m,
 the svd maps as follows:
T(V_i) = sigma_i U_i, where sigma is the i-th diagonal entry of S.
```

```
The Range and Null space of a given matrix A can be represented using
 the SVD.
The right singular values (the V matrix) represents the null space and
 the left singular values span the Range.



From an intuitive perspective, the singular vectors from the SVD
 encode the semiaxes of an N-D ellipsoid while the singular values
 encode the magnitude of those semiaxes.
The Wikipedia page for SVD has a very beautiful transformation that
displays this in 2D space:
https://en.wikipedia.org/wiki/
Singular_value_decomposition#Geometric_meaning

%}
```

project each image onto basis

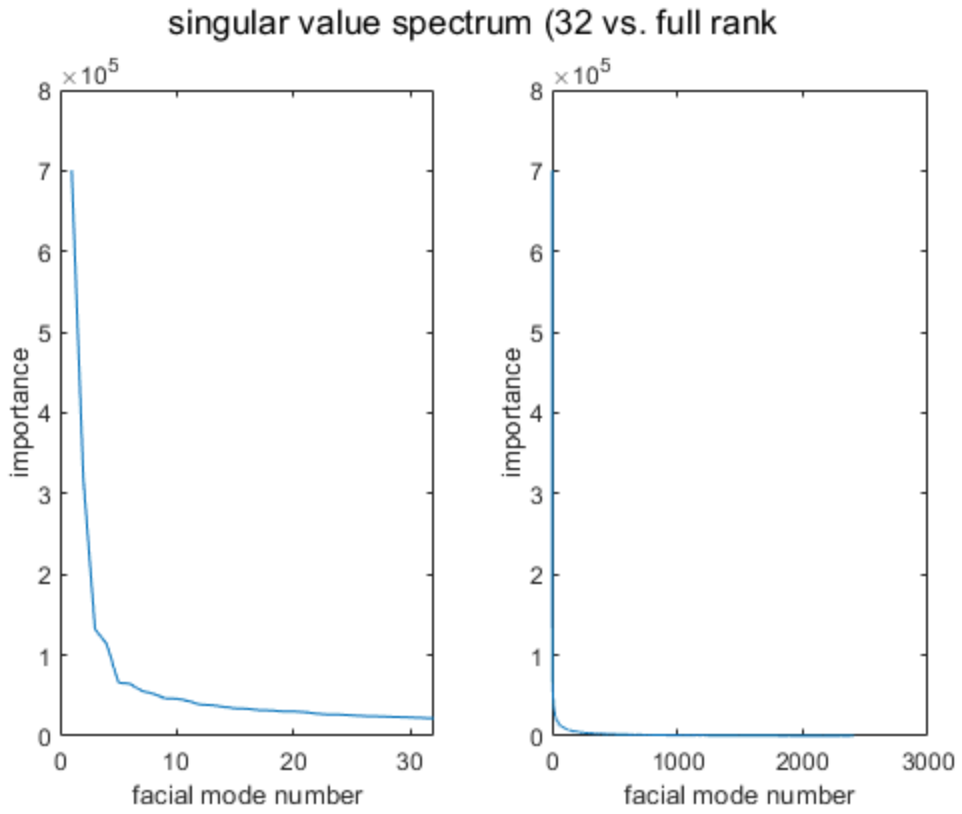
```
    for     j = 1:N
           imvec = A(:,j);
           faceSet(:,j) = imvec'*Phi(:,1:3);
    end
figure()
plot3(faceSet(1,:),faceSet(2,:),faceSet(3,:),'r.')
hold on
title('Faces projected onto the first three left singular vectors');
```



Explore the rank of r (see how many modes we really need)

```
fullModes = diag(S);
x = 1:length(fullModes);
figure();
subplot(1,2,1)
plot(x(1:32),fullModes(1:32)); %pick the first 32 modes so it is
 easier to visualise the highest importance modes.
xlabel('facial mode number');
ylabel('importance');
sgtitle('singular value spectrum (32 vs. full rank');

subplot(1,2,2)
plot(x,fullModes);   %plot all of the modes so we can see the
xlabel('facial mode number');
ylabel('importance');

%{
As seen from the data, the modes take a long time to decay, indicating
 the
rank of the dataset is large. We can also see the modes decay
 following a
```

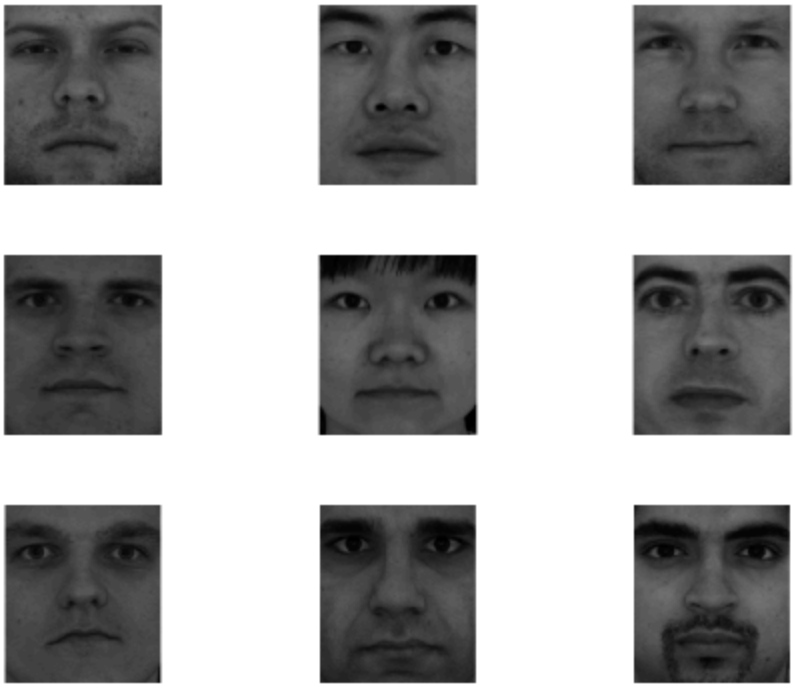
```
1/x relationship. This is often indicative of natural data which
 matches
the dataset. While the first few values are larger, they are still
 within
the same order of magnitude. If  we had a truly dominant mode, we
 would
expect it to be more than an order of magnitude above then next
 highest
value.

It turns out that faces are rather complex structures, even when
 broken
down into a low resolution photo. Within this datasett there are
 around two
thousand, five hundred unique basis to evaluate a face. The fact that
 the
human brain is able to so quickly and with little error is truly
 amazing!

The exact full rank of this dataset is: 2414
%}
```



Creating and exploring faces

```
if k == 2414  %number of photos in full cropped dataset. This is for
 compatability with the smaller set as there wouldn't be enough data
 to do this in the same way.

   hold on
   avgsIndFace = zeros(192,168,9); %Ind is individual
   avgAllFaces = zeros(192,168);
   figure()
   sgtitle('The average face of the first nine people in the
 dataset')
   for people = 1:9 %pick nine people to compare (since that
 fits nicely in a subplot). And the first nine datasets have 64
 unncorrupted pictures each

       for j = 1:64 %photos per person

           imvec = A(:,j+(people-1)*64);
           avgsIndFace(:,:,people)
 =reshape(imvec,m,n)+avgsIndFace(:,:,people);


       end
       avgsIndFace(:,:,people) =
 uint8(1/64.*avgsIndFace(:,:,people));
       hold on

       subplot(3,3,people)
       imshow(avgsIndFace(:,:,people),[0,256]);

       avgAllFaces = 1/9.*avgsIndFace(:,:,people)+ avgAllFaces;

   end

   figure()
   imshow(avgAllFaces,[0,256])
   sgtitle('the average face of all nine photosets')
```

The average face of the first nine people in the dataset



the average face of all nine photosets

The PCA process

```
covariance = (A)*(A');
[eigV,eigD] = eigs(covariance,100,'lm');

for iter = 1:9
    faceEigs = reshape(eigV(:,iter),m,n);
```

```
        subplot (3,3,iter)
        pcolor(flipud(faceEigs)), shading interp, colormap(gray),
 set(gca,'Xtick',[],'Ytick',[])
    end
    sgtitle ('the first nine eigenvectors of the nine sets of
 faces') %note how you can see the forming of the second eVal for the
 full dataset (the half face).


    %now we reconstruct the faces from the eigenvalues and see how
 close we get
    figure()
    sgtitle('The average single face projected onto the eigenvalues of
 the dataset')
    for people = 1:9
        projFace = reshape(avgsIndFace(:,:,people),1,192*168)*eigV;
        subplot (3,3,people)
        bar(projFace(2:20)), set(gca,'Xlim',[0 20],'Ylim',[-2000
 2000],'Xtick',[],'Ytick',[])
        text(0,-1450,strcat('Subject',num2str(people)),'Fontsize',
[15])

    end

    %now do the same for an individual face from each set. They should
 have
    %similar eigenvalues to the average face from their set.


    figure()
    sgtitle('A single persons face projected onto the eigenvales of
 the average face')
    projSingFaceWeights = zeros(100,9);
    for people = 1:9

        projSingFaceWeights(:,people) =
 reshape(A(:,1+(people-1)*64),1,192*168)*eigV; %this picks the first
 photo of each person, any can be chosen
        subplot (3,3,people)
        bar(projSingFaceWeights(2:20,people)), set(gca,'Xlim',[0
 20],'Ylim',[-2000 2000],'Xtick',[],'Ytick',[])
        text(0,-1450,strcat('Subject',num2str(people)),'Fontsize',
[15])

    end

    %plot the control faces
    figure
    sgtitle('Control faces for eigenface comparison')
    for people = 1:9
        subplot (3,3,people)
        imshow(reshape(A(:,1+(people-1)*64),m,n),[0 256])
    end
```

```
    %plot the reconstructions of the faces from the eigenfaces

    eigenvaluesTested = [1,4,8,12,16,64,100];
    for iter = 1:length(eigenvaluesTested)

        eValsTested = eigenvaluesTested(iter);
        figure()
        sgtitle(strcat('The reconstructed faces using
 ',num2str(eValsTested), ' eigenvalue(s)'))

        for people = 1:9

            reconFace(:,:,people) =
 eigV(:,1:eValsTested)*projSingFaceWeights(1:eValsTested,people);
            subplot(3,3,people)
            imshow(reshape(uint8(reconFace(:,:,people)),192,168))

        end

    end
  %%PCA Discussion
%{
    Plotting the face weights for the different subjects yelded
 resutls
    worth discussing. We see a significant difference between the
 average
    of a person's face versus their face in neutral lighting. It is
    suspected that this comes from the photos which have signifcant
    portions of the face shaded in darkness, bringing down the average
    pixel brightness, and requiring a significanly different
 combination of
    singular values to reconstruct. Subject 5, being the only one with
    bangs, had a very prominant identifier within the data. The sixth
    eigenface encoded bangs (as seen from the chart above) which
 allowed
    for easy identification.

    The benefits of looking at a range of subjects allows the
 extraction of
    data which might have been lost when looking just at a single
 person.
    This is a very prominent reminder that the algorithms we produce
 need
    to be trained with a wide variety of people, so as not to incur
 bias
    within the data.

    Looking to the reconstructed faces, we can get a feel for the rank
 of
    the face space. With just rank one, it is impossible to tell the
    subjects apart. With rank four, we start to see some changes in
    brightness but the subjects are largely unidentifiable. With rank
 8 we
```

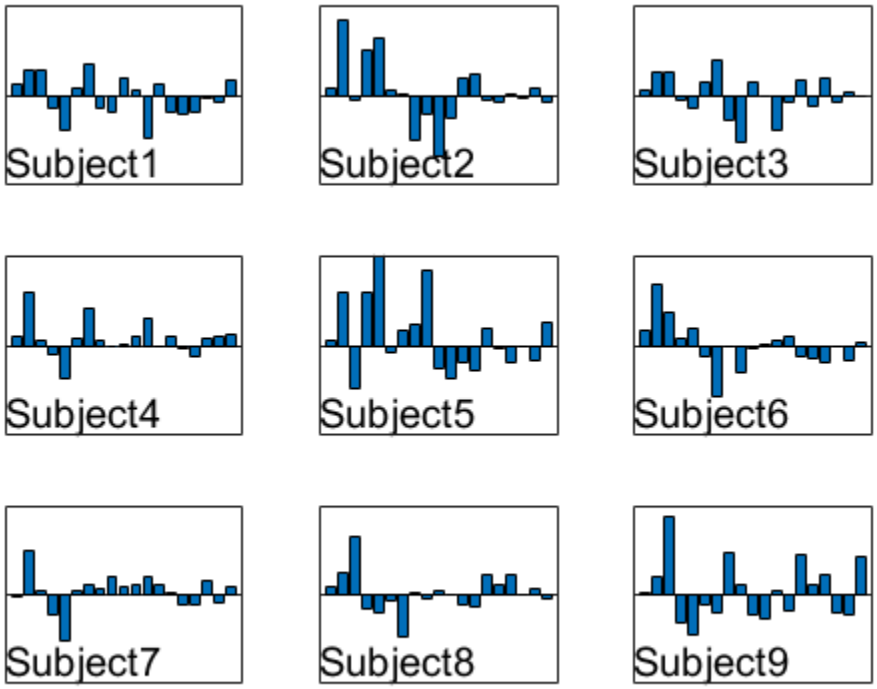
```
    start to see changes present (especialy with subject five as we
 have
    included the 6th eigenface).With rank twelve, we can identify a
 few
    more faces but it is still difficult to see the difference between
 all
    subjects. At rank sixteen, all faces look distinct, but it would
 be
    difficult to exactly identify against the control images
 (especially
    for the leftmost collumn of subjects). At rank 64, all faces are
 clear
    and easily identifyable. There are still reconstruction errors
 present,
    especially around the eyes, but on the whole, the faces compare
 well
    against the control. At rank 100, there is even more detail
 present in
    the faces, with most incorrect blemishes removes. Errors still
 exist
    around the eyebrows and the camera flash is not present but this
 is of
    greater detail needed to identify the individuals.


    Thus, we have taken the rank down from 2414 down to 64. This is a
    massive reduction in the required compute(37.7 times to be
 precise).

%}
```

Elapsed time is 818.173621 seconds.

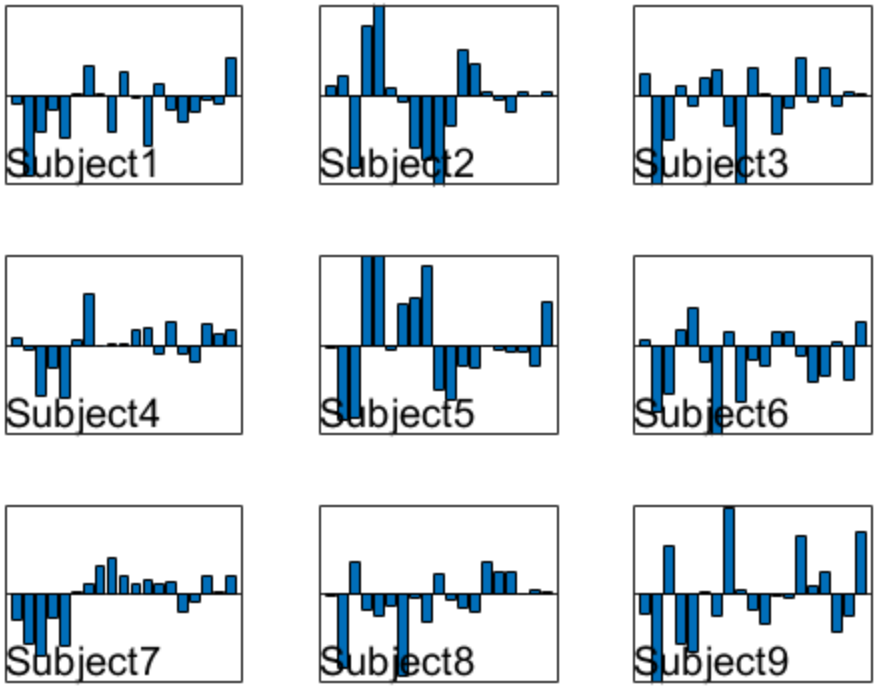
```
else
    %skip the face reconstruction since there isn't enough data here
 with a
    %single person's dataset.
end
toc %to know how long the execute should take next time
```

the first nine eigenvectors of the nine sets of faces

The average single face projected onto the eigenvalues of the dataset

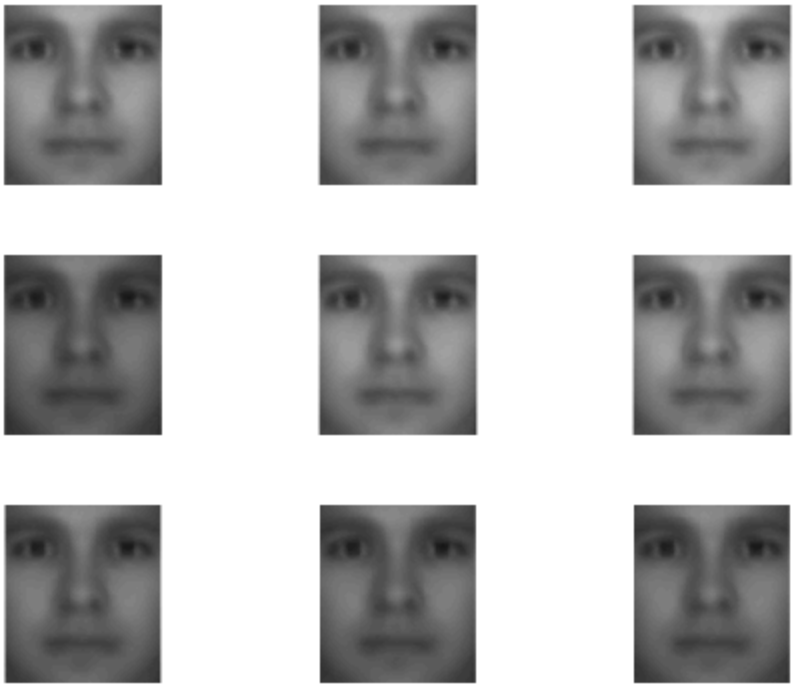


A single persons face projected onto the eigenvales of the average face

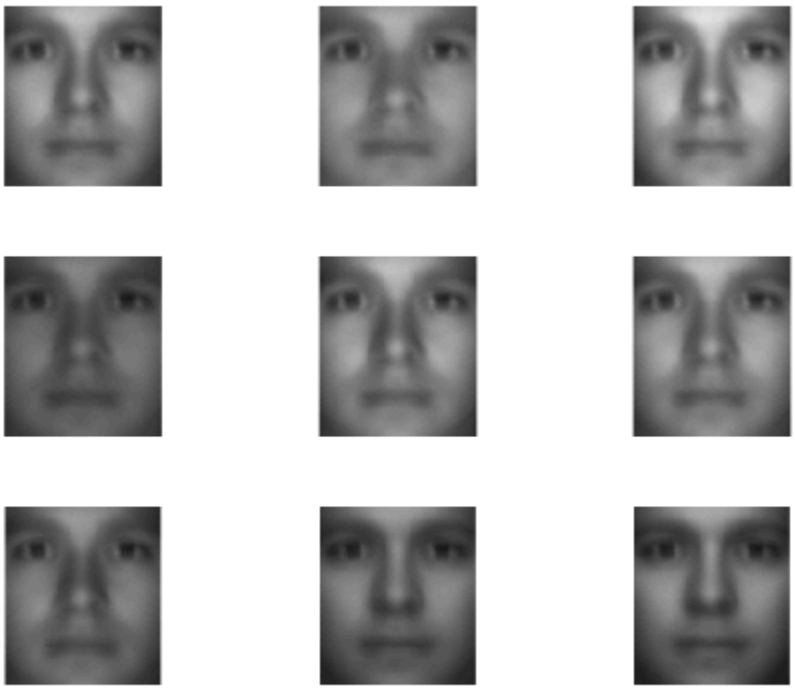


Control faces for eigenface comparison

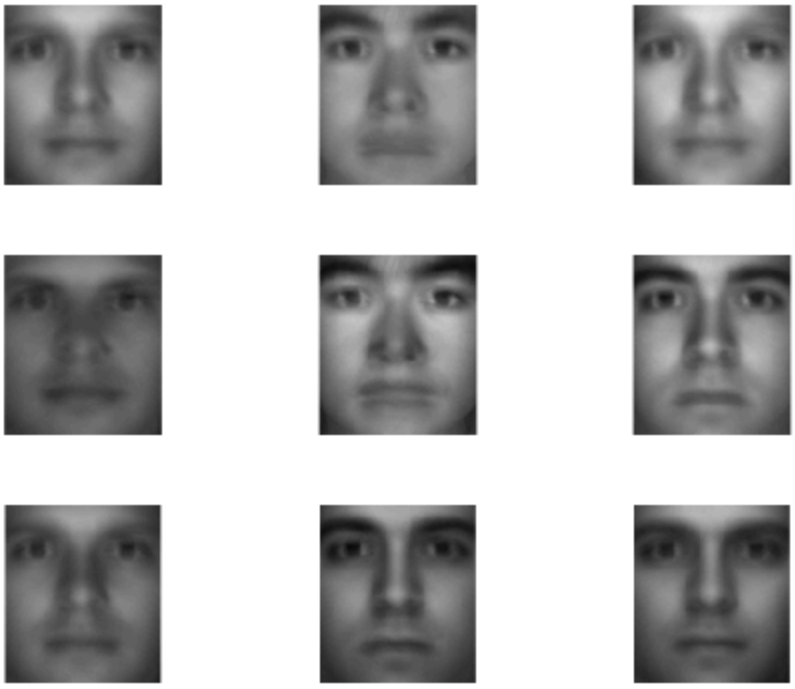
The reconstructed faces using1 eigenvalue(s)



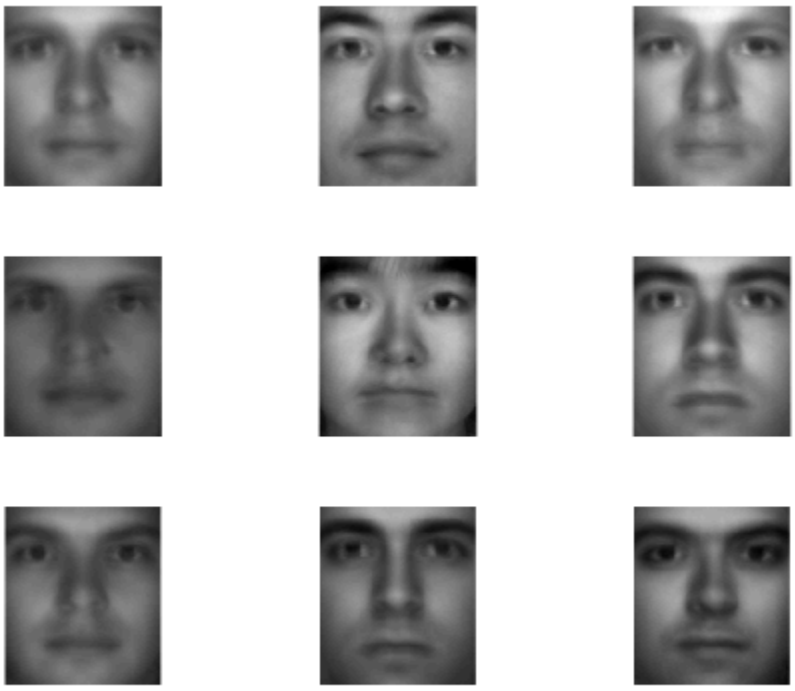
The reconstructed faces using4 eigenvalue(s)



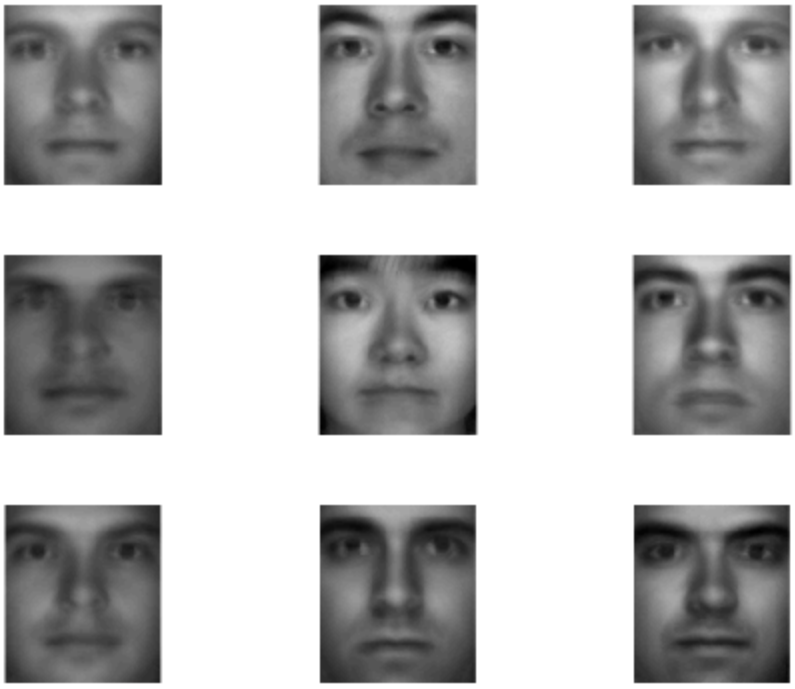
The reconstructed faces using8 eigenvalue(s)



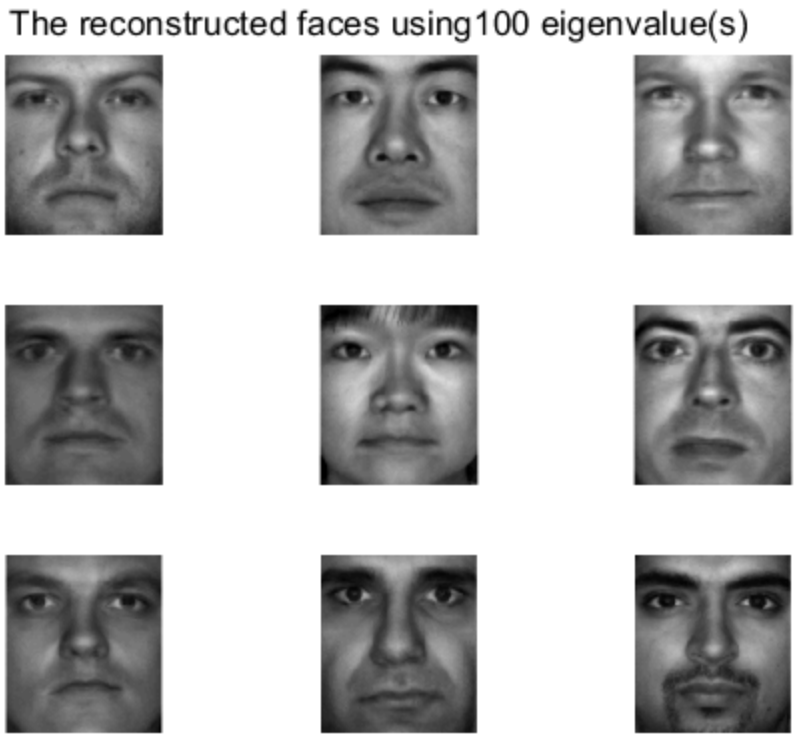
The reconstructed faces using12 eigenvalue(s)



The reconstructed faces using16 eigenvalue(s)



The reconstructed faces using64 eigenvalue(s)



Load in the uncropped face database

```
% Specify the folder where the files live.
myFolder2 = 'C:\Users\PhD\Documents\MATLAB\ME 564\faces\yalefaces\';
  %CroppedYale\yaleB31   or yalefaces\
% Check to make sure that folder actually exists.  Warn user if it
 doesn't.
if ~isfolder(myFolder2)
    errorMessage = sprintf('Error: The following folder does not
 exist:\n%s\nPlease specify a new folder.', myFolder2);
    uiwait(wfaceSetdlg(errorMessage));
    myFolder2 = uigetdir(); % Ask for a new one.
    if myFolder2 == 0
         % User clicked Cancel
         return;
    end
end
% Get a list of all files in the folder with the desired file name
 pattern.
filePattern2 = fullfile(myFolder2, '**/*.pgm'); %search for whatever
 you need. The **/ searches subfolders as well
theFilesUncropped = dir(filePattern2);
N = length(theFilesUncropped);
```

```
% Size of each picture
m_UC = 180;
n_UC = 137;

avgUC = zeros(m_UC*n_UC,1);  % the average face
A_UC = [];
count = 0;

for k = 1 : length(theFilesUncropped)
    baseFileNameUC = theFilesUncropped(k).name;
    fullFileNamUC = fullfile(theFilesUncropped(k).folder,
 baseFileNameUC);
    imageArrayUC = imresize(imread(fullFileNamUC),[n_UC
 m_UC]); %resize the imported photos to be easier to process in Matlab
%     figure
%     imshow(imageArrayUC);  % Display image.
%     drawnow; % Force display to update immediately.


    if(size(imageArrayUC,3)==1)
        M=double(imageArrayUC);
    else
        M=double(rgb2gray(imageArrayUC));
    end
    %pause(0.01);
    R = reshape(M,n_UC*m_UC,1);
    A_UC = [A_UC,R];
   avgUC = avgUC + R;
   count = count + 1;
end
avgUC = avgUC /count;
```

Calculate the "average" face from all photos

```
avgTS_UC = uint8(reshape(avgUC,n_UC,m_UC));
figure(), imshow(avgTS_UC);
sgtitle('Average Face of the uncropped dataset');
```

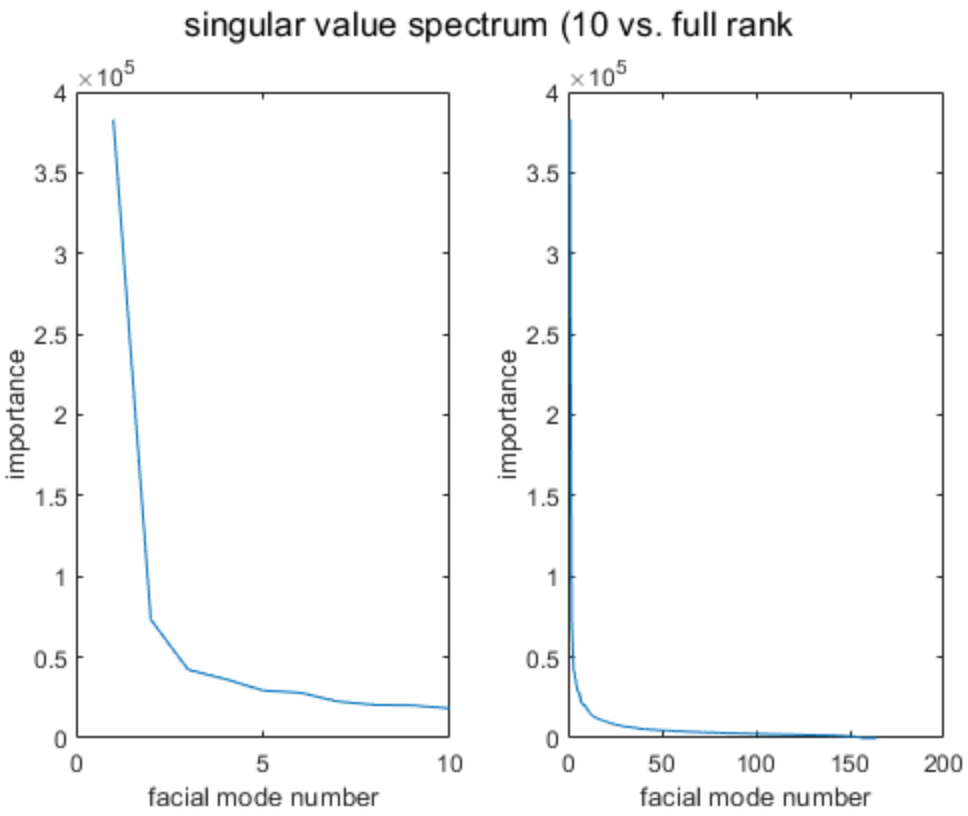
Computing the SVD

```
[U_UC,S_UC,V_UC] = svd(A_UC,0);
PhiUC = U_UC(:,1:N);
PhiUC(:,1) = -1*PhiUC(:,1);
figure()
count = 1;
for iter=1:5
    for j=1:5
        subplot(5,5,count)
        imshow(uint8(25000*reshape(PhiUC(:,count),n_UC,m_UC)));
        count = count + 1;
    end
end
sgtitle('The reshaped collumns of U (Eigen Faces)');
```

Explore the rank of r (see how many modes we really need)

```
fullModesUC = diag(S_UC);
xUC = 1:length(fullModesUC);
figure();
subplot(1,2,1)
plot(xUC(1:10),fullModesUC(1:10));
```

```
xlabel('facial mode number');
ylabel('importance');
sgtitle('singular value spectrum (10 vs. full rank');

subplot(1,2,2)
plot(xUC,fullModesUC);
xlabel('facial mode number');
ylabel('importance');
```



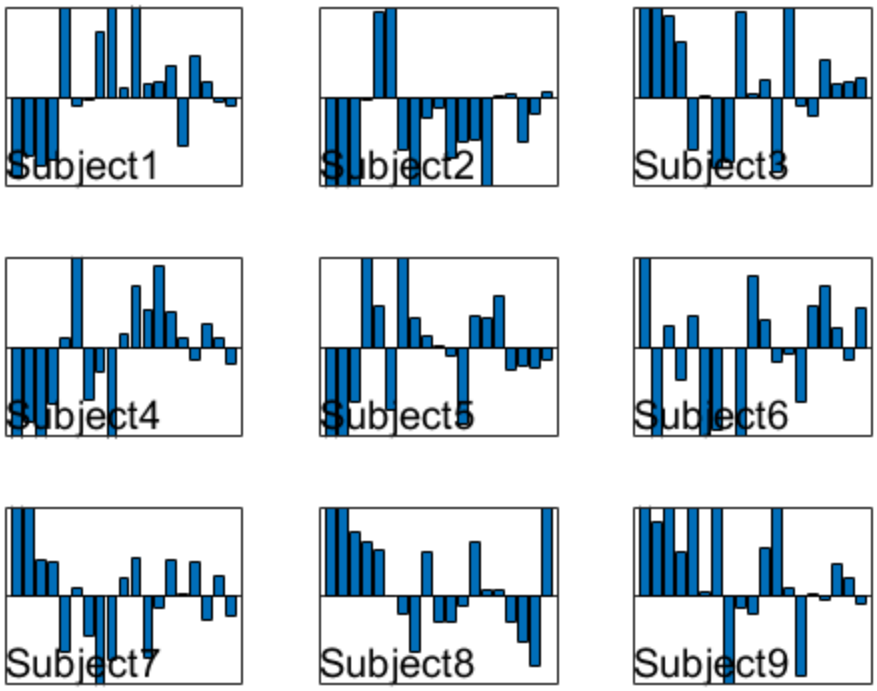
Creating and exploring faces

The PCA process

```
covarianceUC = (A_UC)*(A_UC');
[eigV_UC,eigD_UC] = eigs(covarianceUC,100,'lm');


%now do the same for an individual face from each set. They should
have
%similar eigenvalues to the average face from their set.


figure()
```

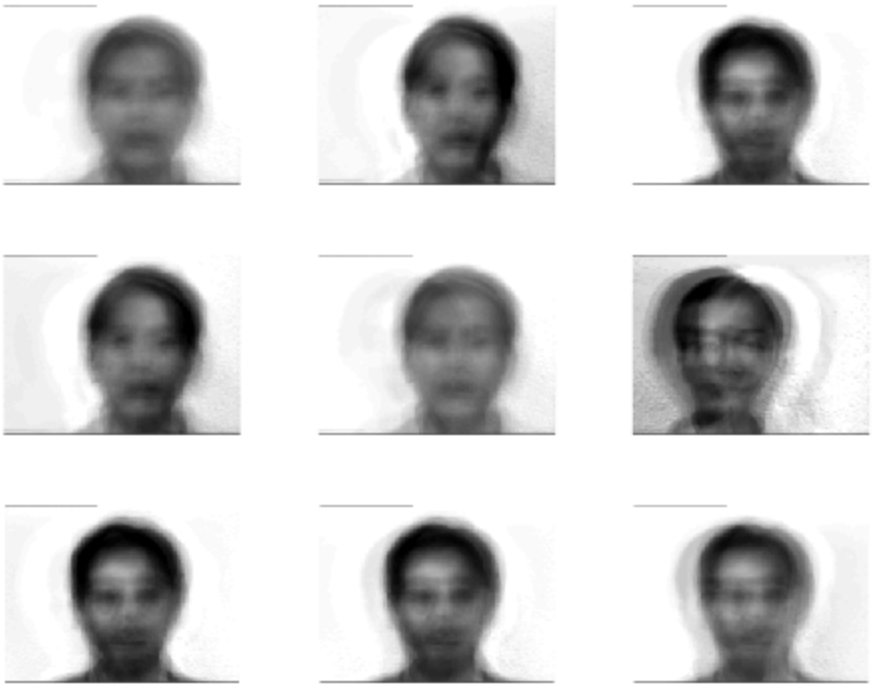
```
    sgtitle('A single persons face projected onto the eigenvales of
 the average face') %but now with the weird ass faces
    projSingFaceWeights_UC = zeros(100,9);
    for people = 1:9

        projSingFaceWeights_UC(:,people) =
 reshape(A_UC(:,5+(people-1)*11),1,n_UC*m_UC)*eigV_UC; %this picks the
 first photo of each person, any can be chosen
        subplot (3,3,people)
        bar(projSingFaceWeights_UC(2:20,people)), set(gca,'Xlim',[0
 20],'Ylim',[-2000 2000],'Xtick',[],'Ytick',[])
        text(0,-1450,strcat('Subject',num2str(people)),'Fontsize',
[15])

    end

    %plot the reconstructions of the faces from the eigenfaces

    eigenvaluesTested_UC = [1,4,16,32,64,100];
    for iter = 1:length(eigenvaluesTested_UC)

        eValsTestedUC = eigenvaluesTested_UC(iter);
        figure()
        sgtitle(strcat('The reconstructed faces using
 ',num2str(eValsTestedUC), ' eigenvalue(s)'))

        for people = 1:9

            reconFace_UC(:,:,people) =
 eigV_UC(:,1:eValsTestedUC)*projSingFaceWeights_UC(1:eValsTestedUC,people);
            subplot(3,3,people)
            imshow(reshape(uint8(reconFace_UC(:,:,people)),n_UC,m_UC))

        end

    end
```

A single persons face projected onto the eigenvales of the average face

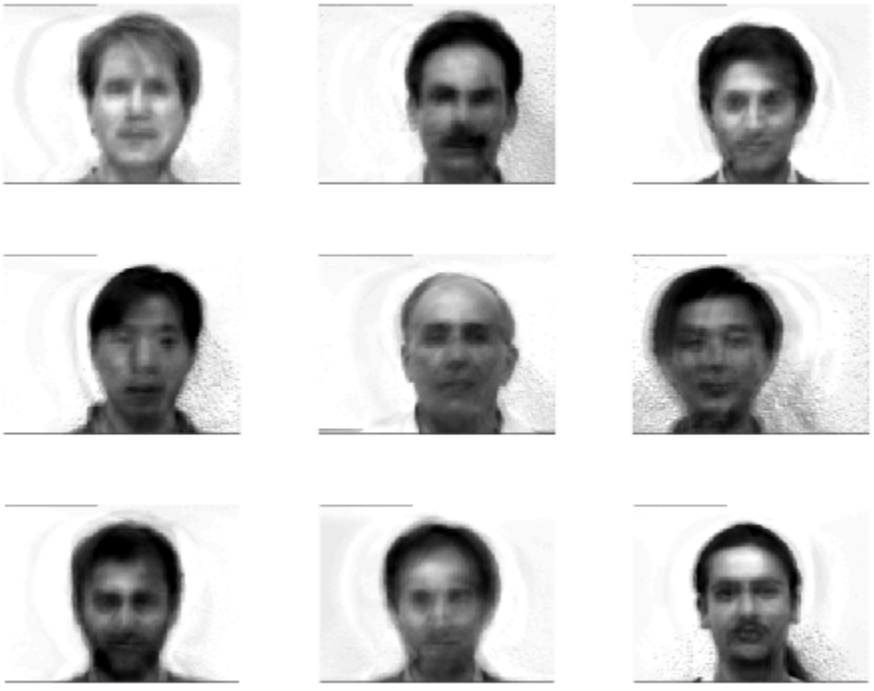


The reconstructed faces using1 eigenvalue(s)

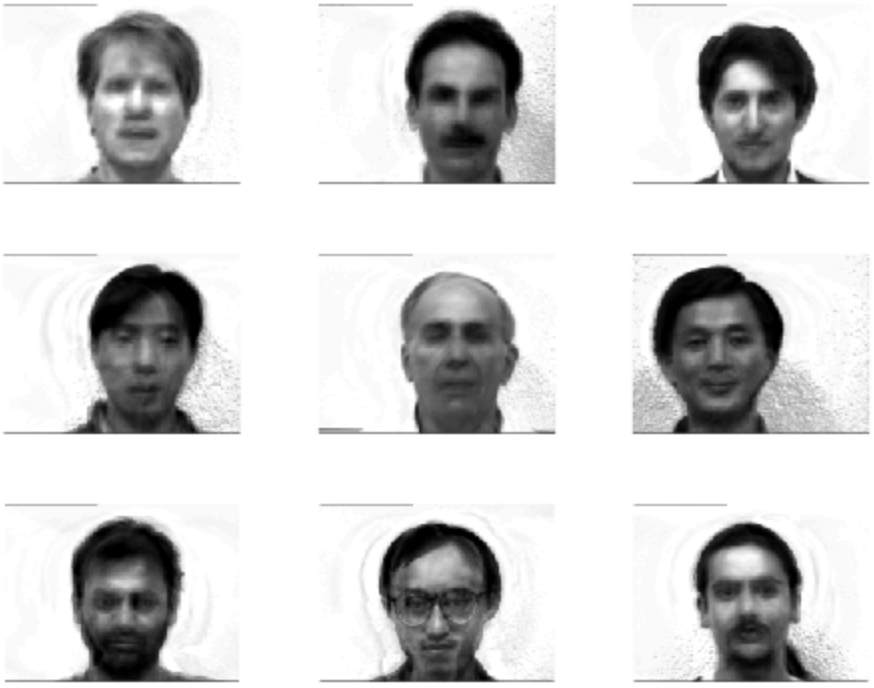
The reconstructed faces using4 eigenvalue(s)



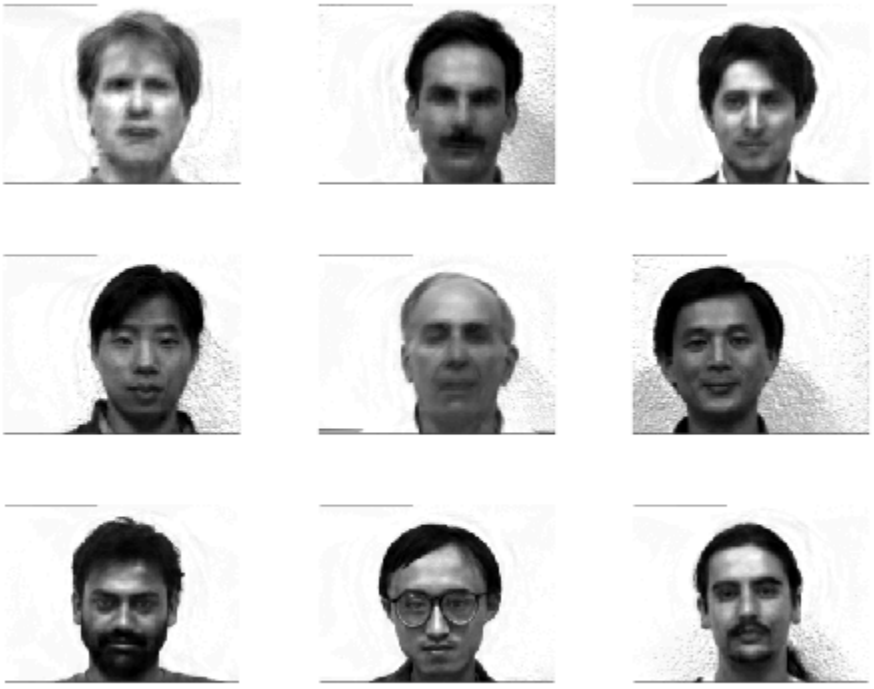
The reconstructed faces using16 eigenvalue(s)

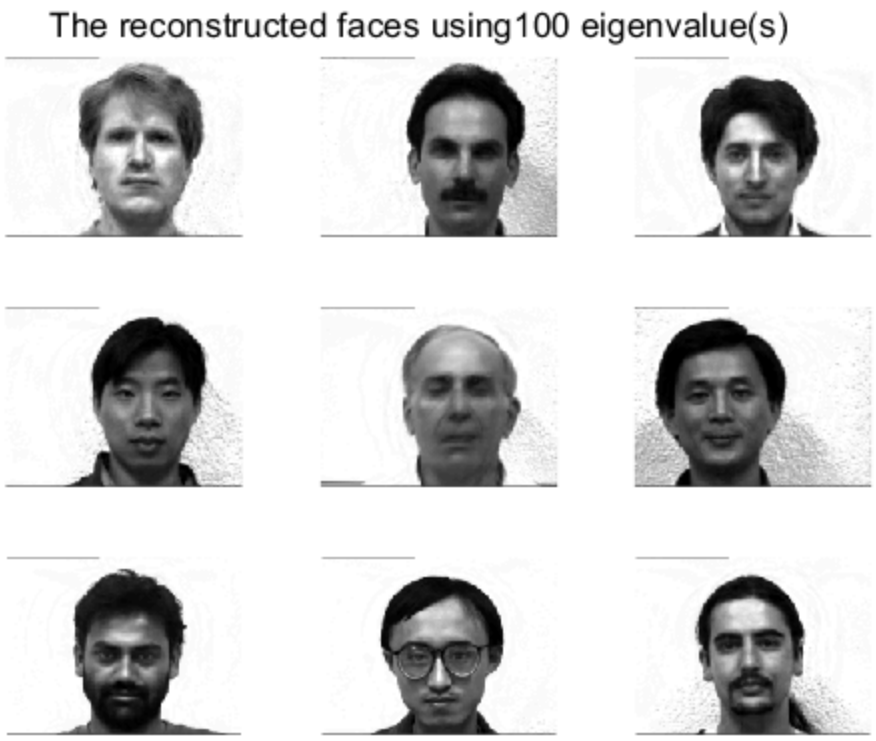


The reconstructed faces using32 eigenvalue(s)



The reconstructed faces using64 eigenvalue(s)



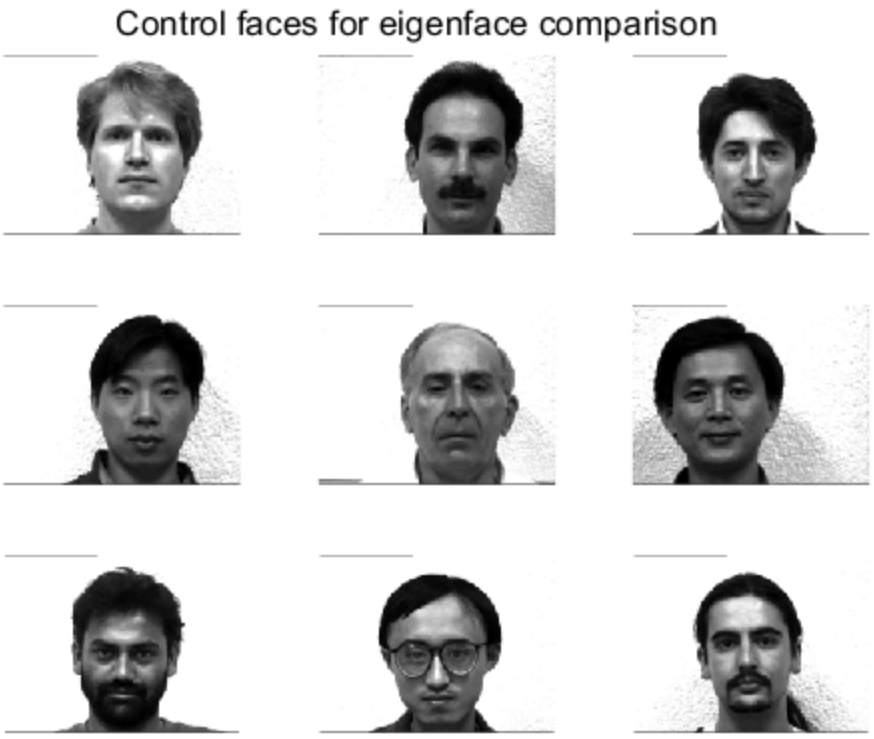


Discussing PCA with the uncropped faces

```
   %{
       Looking to the reconstructed faces, we can get a feel for the
rank of
   the face space. With just rank one, it is impossible to tell the
   subjects apart. With rank four, we start to see some changes in
   brightness but the subjects are largely unidentifiable and full of
   ghosting. Subject six is showing massive ghosting from the
different
   positions their photos were taken from. Moving up to rank sixteen,
we
   start to see discernable differences between subjects but most of
their
   facial features are obscured or blurred. At rank 32, it is still
very
   difficult to identify faces but features like hair color and hair
   shape are discernable. There is noticably less background ghosting
but
   it is still very present within the reconstructions. At rank sixty
four
   facial features are now idenitfiable but eyes and eyebrows are a
   blurred mess. At rank 100, eyes and eyebrows are now seperated for
all
   subjects and would likely be idenitfiable. There is still a large
```

```
    ammount of background ghosting present in the photos.

    We also notice the decay of the singular values in this dataset
 was
    slower than that in the cropped and alligned data. We also see the
    singular values only stretch out to ~170. The fact that we needed
 most
    of the rank to have a halfway good chance at reconstructing the
    subject's faces meant we were not getting good data reduction. The
 set
    also follows a 1/x relationship telling us, it is likely made from
    natural sources.

    Compared to the cropped and alligned images, this dataset required
 a
    much larger rank to recreate. Even at rank 100, the data was about
    equivalent to rank 16 with the cropped and alligned dataset. This
 shows
    the importance of properly setting up your data collection to get
 the
    desired results. There is a reason the saying garbage in, garbage
 out
    exists!

    %}


    %plot the control faces
    figure
    sgtitle('Control faces for eigenface comparison')
    for people = 1:9
        subplot (3,3,people)
        imshow(reshape(A_UC(:,5+(people-1)*11),n_UC,m_UC),[0 256])
    end
%%End Code
    toc
```

Elapsed time is 950.256648 seconds.



Published with MATLAB® R2019b