AMATH 482- Computational Methods in Data Analysis: Principal Component Analysis

Gargi Kher

February 21st, 2020

Abstract: Principal Component Analysis (PCA) is a statistical analysis technique that uses the Singular Value Decomposition (SVD) method to analyze large sets of data. The PCA uses certain linear algebra techniques found in SVD to help determine the lowest dimension of the data that can be worked with while still retaining most of the information needed to make conclusions, saving the amount of information needed to be stored. I will be discussing how I used PCA to analyze twelve videos of a paint can's motion as it moved through space.

Sec. I. Introduction and Overview

In this assignment, we were given twelve videos of a paint can moving through a room while attached to a string. There were four test cases that held three of the videos: an "ideal" case (Test 1), a "shaky camera" case (Test 2), a "horizontal displacement" case (Test 3), and a "horizontal displacement and rotation" case (Test 4). All of the videos in each case corresponded to three different cameras positioned at different angles. The first two cameras captured the paint can's vertical motion while the third camera in each case was recording sideways, making the paint can appear as if it were moving horizontally. All of the video recordings were given to us and we were given the code that split them into different time frames. We first had to obtain the position of the paint can at each time for each video so that we could apply Principal Component Analysis to each case. This analysis was supposed to tell us the dimension of the paint can's motion.

Sec. II. Theoretical Background

The PCA is one form of a method that uses the Singular Value Decomposition (SVD). To understand PCA, we must discuss SVD. SVD takes a matrix and basically decomposes it into other matrices which represent the same information as the original matrix. Just like a matrix transformation, the SVD takes a set of vectors and represents them in a different basis.

$A = U\Sigma V^T$ (1) Singular Value Decomposition of a matrix A

Equation (1) depicts the SVD of a matrix A. This matrix is broken down into three components: U, which is a rotation matrix; Σ, which is a diagonal matrix representing the stretching factor and also known as the singular value matrix; and V^T, which is a rotation matrix (Note: V^T stands for "V transpose". This is important to do when you are dealing with complex values). To better understand what SVD does, it is best to consider an example. Imagine you have a unit circle (or sphere) in some dimension n. Pick two orthogonal vectors stating from the origin and pointing to the ends of this region. These will be known as $\vec{v}_1$ and $\vec{v}_2$. Now imagine that you took this sphere and transformed it to another sphere in some dimension m (m must be greater than n for this to work). You will then be able to pick two new orthogonal vectors in that sphere that can be called $\vec{u}_1$ and $\vec{u}_2$. These $\vec{u}$ vectors have been scaled by some factor σ during the transformation, where σ_1 corresponds to $\vec{u}_1$, σ_2 corresponds to $\vec{u}_2$, and so on. It is probably clear now that $\vec{u}$ makes up U, $\vec{v}$ makes up V^T, and σ makes up Σ in our SVD equation. If you think about the transformation

geometrically, this makes sense. If A is a transformation applied to some vector $\vec{x}$, then you would get something like this: $A\vec{x} = U\Sigma V^T\vec{x}$. The matrix V^T would first be applied to $\vec{x}$, rotating the vector and changing the basis. $U\Sigma$ would then be applied to this result, which would respectively rotate and stretch the vector.

With this example, there are several important rules to note. First, the rank of the matrix A is equal to all of the nonzero singular values, or the number of rows of Σ. The singular values are in descending order of magnitude. These will be able to tell us the lowest rank of A. The vectors $\vec{u}$ represent a basis for the range of A, or all of the dimensions represented by A. The vectors $\vec{v}$ are a basis for the null space of A, or a basis for the vectors that get sent to $\vec{0}$ after the transformation occurs.

PCA uses SVD, and much of these concepts are the same. The singular values of PCA are called the "variances". The matrices U and V^T are called the "principal components" and are contain the left- and right- singular vectors, respectively. Other than this name change, the function of these vectors remains the same as in SVD. Doing PCA just allows us to find a new basis where the basis vectors are uncorrelated, or the covariance matrix is diagonal. The term covariance indicates how closely two datasets vary with each other. The higher the covariance, the more correlated the datasets are. This can tell us about redundancies in our data, something that was important to this project.

The matrix I applied the PCA to is known as a snapshot matrix, because it contains position data for each time. Snapshot matrices can contain position data in the rows while each column represents a different time. In the case of a video, the columns would hold the positions for each time frame.

Once you have the PCA of a matrix, you can use the information it gives you to determine the general trend of the data. The SVD gives the best low-rank data, meaning it tells you the smallest dimension of A to look at that will still give you most of the information without having to store or work with as much information as the original. This is done by looking at the "energies" of the singular values. The energy is a measure of the information captured from each "mode" or "rank".

$$\frac{\sigma_r^2}{\sigma_1^2+\cdots+\sigma_N^2}$$ (2) Energy equation

Equation (2) shows how the energy of a particular mode (corresponding to Σ) r can be captured. The lowest rank of the matrix observed you can observe will correspond to the first singular value. Plotting the energies of each rank will allow you to see whether or not they are correlated with each other. The rank of the matrix should be picked so that the energies aren't correlated with each other. Generally, the rank can be determined by seeing the number of singular values whose summed energies are close to 100%. Once this lowest rank is known, data can be extracted from U and V^T. For instance, if the lowest rank is 2, you can look at the first two columns of U to obtain the direction of the basis of A. The first two columns of V^T would tell you how each singular value changes with respect to time. In this case, looking at the energies of

the singular values and determining the rank tells us about the dimension of motion of the paint can.

Sec. III. Algorithm Implementation and Development

The first thing to do was to obtain the position of the paint can for each frame in each video. Code was provided by the instructor for this as well as for dividing the video into time frames. I separated my analysis for each camera into sections, ordered by the test number. The way I obtained the points for Tests 1-3 followed the same method, while my strategy for Test 4 differed slightly from these.

For the cameras in Tests 1-3, I started off by creating a grayscale image of each time frame, because the grayscale image stores pixel values (which is what I eventually need) in a two-dimensional matrix. I then found the sum of all the grayscale images and averaged them. Going back into a for loop, I subtracted this average from all of the time frames. This way, I thought averaging the frames would keep most of the background the same while showing most of the change happening with the paint can, which was the object that was moving the most (Appendix B: Lines: 2-16,48-62,94-108,140-154,186-200,232-246,269-284, 316-331,363-378). For each of these cameras, I used the $ginput()$ function to manually find the position of the paint can in the first frame, or first time. These coordinates were recorded. I then went through the remaining time frames, finding the maximum difference between pixels and recording the value that was closest to the previous point. I took advantage of the fact that cameras 1 and 2 captured nearly just the vertical motion and camera 3 captured basically only horizontal motion. I assumed that during vertical motion, the x-coordinate basically stays the same while the y-coordinate changes. I also similarly assumed that during horizontal motion, the y-coordinate doesn't vary as much from frame to frame as much as the x-coordinate does. The x-value of the first pixel coordinate I received corresponded to the first element of what I called my "$tempcolsN\_t$", where "N" was the camera number and "t" was the test number. This was why, for cameras 1-2 on these tests, I found each sequential point by seeing which of the given maximum points was closest to the previous column or x-value. Similarly, because camera 3 captured horizontal motion, I found the location coordinate by comparing the given maximum difference values to the row value ("$temprowsN\_t$") of the previous frame (Appendix B: Lines 18-37,64-83, 100-129,156-175,202-221,248-267,286-305,332-352,380-399).

I found the location of the paint can for Test 4 slightly differently than the other tests. Because this camera was shaking while the paint can was rotating, implementing the same strategy as above resulted in most of the obtained locations being off of the paint can. I instead found the difference between each consecutive frame, and used that to find the maximum difference values at each time. From there, obtaining the points was similar to Tests 1-3 (Appendix B: Lines 401-525).

Once I had obtained the paint can locations for each camera across all tests, I had two matrices for each camera: "$temprowsN\_t$" for the y-coordinate of the pixel, and "$tempcolsN\_t$" for the x-coordinate of the pixel. For each test, I used these vectors to plot the motion of the paint can for each camera ("$temprowsN\_t$" for cameras 1-2 and "$tempcolsN\_t$" for camera 3), and used the resulting figures to determine how much to cut the vectors for each

camera. These cut vectors were then put into a snapshot matrix, where I subtracted the mean of each row from their respective rows in order to center the coordinates at zero. I then performed SVD on all of the snapshot matrices for all tests (Appendix B: Lines 528-587).

I used the singular values of the matrix to plot the energies. This is what I used to help me determine the dimensions (Appendix B: Lines 589-608).

Sec. IV. Computational Results

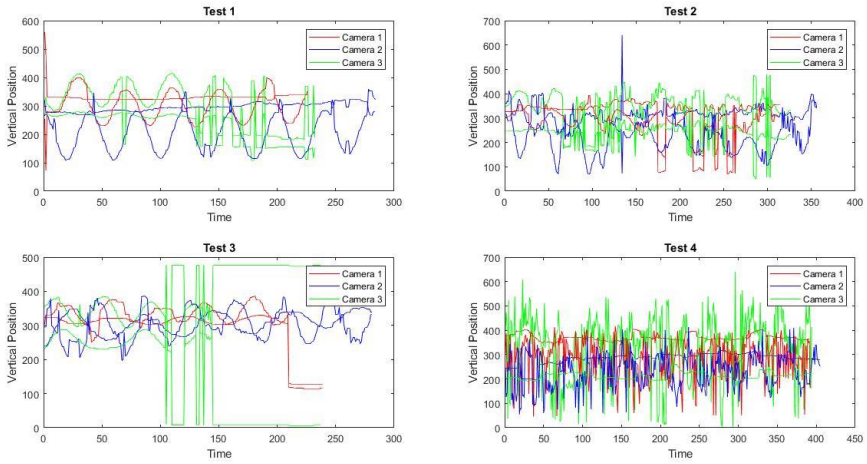

Figure 1: Vertical (y-coordinate) positions of the paint can for all the cameras in each test.

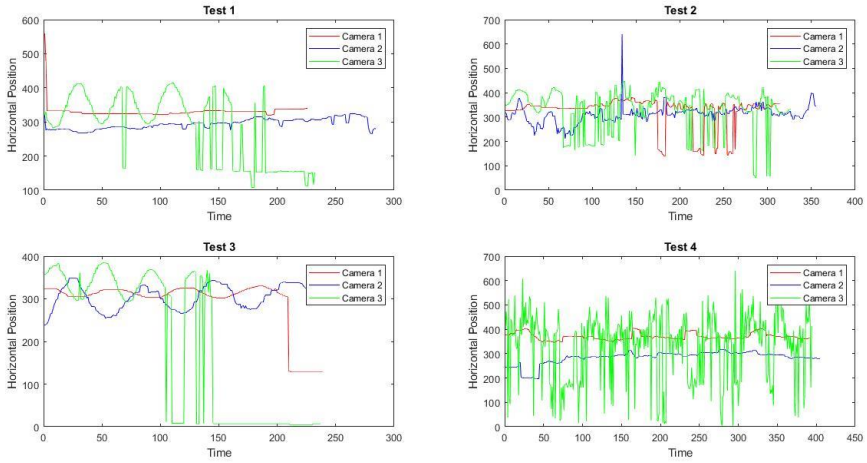

Figure 2: Horizontal (x-coordinate) positions of the paint can for all the cameras in each test.

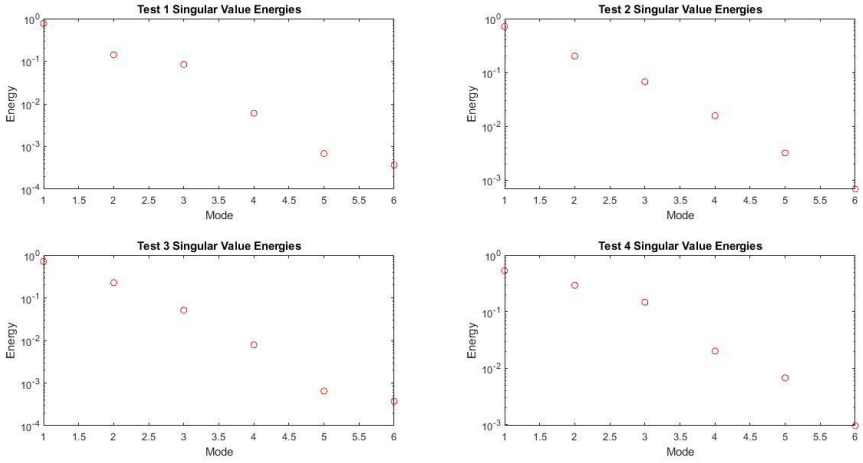

Figure 3: Energy values for all six of the modes for each tests' snapshot matrix.

Sec. V. Summary and Conclusions

Clearly, the vertical and horizontal positions of the paint can's entire motion for each test aren't smooth (Sec.IV, Figures 1 and 2, respectively). Because the paint can moves up and down, they should be somewhat like sine waves. While I tried to keep the locations on the same point of the paint can for each camera, they were often times different between frames. For instance, in one frame the tracked point could have been at the top of the paint can while in the next frame the tracked point was on the bottom of the paint can. There were also points, especially in tests 3 and 4, that were located off of the paint can. All of this could have contributed to the noise seen in these figures. However, if you compare the horizontal and vertical conditions' positions for each camera in each test, you can see that for the most part, one of the positions remains constant (x-coordinate for camera 1 and 2, y-coordinate for camera 3) while the other is clearly changing, signifying the paint can was moving in that direction for that camera.

To determine the rank of the snapshot matrix, and ultimately the dimension of motion, I looked at the energy plots for each test and the energy values for each singular value. From watching the videos, I knew that the motion of the paint can was one-dimensional for tests 1 and 2 and three-dimensional for tests 3 and 4. I was expecting the energies of the singular values for these tests to reflect this information. However, I did receive unexpected results. For test 1, the singular value energy plot appears to show a rank of 1 (Sec.IV, Figure 3, plot 1). It seems to be this way because even though the second and third singular values are only an order of a magnitude lower than the first, they seem to be close to each other. When I look at the energy values, they suggest this test to be rank 2, because about 90% of the data is contained within the first two singular values (The energy for the first singular value is about 76% while the energy of the second is about 14%). Test 2 seems to show almost rank 4 or 5 (Sec.IV, Figure 3, plot 2). The paint can was moving in one-dimensional motion, so these values don't make sense. Looking at the energy values, however, about 71% of the energy is contained in the first singular value while about 20% is contained in the second, suggesting a rank of 2. The energy plot for test 3 suggests a rank of two or three, but it is difficult to tell for sure (Sec.IV, Figure 3, plot 3). Looking at the energy values for this test, we see that it is likely rank 2, as the energy of the first

singular value is about 84% and the energy of the second is about 12%. Test 4's energy plot shows a rank of 3 (Sec.IV, Figure 3, plot 4). The energy values, about 52%, 28%, and 16% for the first, second and third singular values, also suggest that Test 4 has a rank of 3.

Other than the camera shake in test 2, some of these unexpected results could be due to the way I was picking my points on each video frame. For the most part, especially for test 1, the point for each video frame was on the same part of the paint can- the flashlight. During tests 3 and 4 however, the point would be on the top of the paint can for one frame and on the bottom of the paint can for another. While this is still selecting the location of the right object, this data could make it seem like the paint can moved down when it actually moved up, for instance. This likely affected results of the PCA.

As the ranks are supposed to tell you the dimension of motion, the rank approximation can be obtained by taking as many columns in U and V^T as the number of ranks and multiplying them by the singular values.

Sec. VI. Sources/References

Kutz, Nathan J. (2013). *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*

Appendix A. MATLAB functions used and brief implementation explanation

$imshow(I)$ – Displays an image I.
$implay(x)$ – Plays a movie or video with an input file x.
$ginput(x)$ – Gives a pixel value for an image. Number of pixels given is defined by an integer x.
$imadd(x)$ – Adds image frames together.
$reshape(X,[])$ – Reshapes a matrix X to a different size specified in [], keeping the number of elements the same.
$find(x)$ – Finds the index value of x that matches a specific condition.
$semiology()$ - Works just like $plot()$, but converts the y-axis to log scale.
$diag(A)$ – Diagonalizes a matrix A, Or converts a diagonal matrix into a vector.
$[U,S,V] = svd(A,'econ')$ – Takes the reduced singular value decomposition of matrix A, giving the left singular vectors in U, singular values in S, and right singular vectors in V.

Appendix B. MATLAB codes

```
%% Test 1
load('cam1_1.mat')
numFrames = size(vidFrames1_1,4);
dif = zeros(1,307200);
for j = 1:numFrames
X = vidFrames1_1(:,:,:,j);
X_gray = rgb2gray(X);
if j ==1
new = reshape(rgb2gray(vidFrames1_1(:,:,:,j)),[1,307200]);
end
```

```
new = imadd(reshape(rgb2gray(vidFrames1_1(:,:,:,j)),[1,307200]),new);
end
avgframe = new./numFrames;
for j = 1:numFrames
dif(j,:) = reshape(rgb2gray(vidFrames1_1(:,:,:,j)),[1,307200])-avgframe;
end
%%
X = vidFrames1_1(:,:,:,1);
X_gray = rgb2gray(X);
imshow(X_gray)
ginput(1)
%% find greatest difference between frames
temprows1_1 = zeros();
tempcols1_1 = zeros();
temprows1_1(1) = 253;
tempcols1_1(1) = 559;
for i = 2:numFrames
    X = vidFrames1_1(:,:,:,i);
    X_gray = rgb2gray(X);
    temp = find(dif(i,:) == max(abs(dif(i,:))));
        [rows,cols] = find(X_gray==X_gray(temp(1)));
           count1 = abs(cols-tempcols1_1(i-1));
           count = find(count1 == min(count1));
           [c, rindex] = min(abs(rows(count)-temprows1_1(i-1,:)));
           temprows1_1(i,:) = (rows(count(rindex)));
           tempcols1_1(i,:) = (cols(count(rindex)));
end
%%
for j = 1:size(vidFrames1_1,4)
X = vidFrames1_1(:,:,:,j);
X_gray = rgb2gray(X);
imshow(X);
hold on
plot(tempcols1_1(j),temprows1_1(j),'ro')
pause(0.1)
end
%%
load('cam2_1.mat')
numFrames = size(vidFrames2_1,4);
dif = zeros(1,307200);
for j = 1:numFrames
X = vidFrames2_1(:,:,:,j);
X_gray = rgb2gray(X);
if j ==1
new = reshape(rgb2gray(vidFrames2_1(:,:,:,j)),[1,307200]);
end
new = imadd(reshape(rgb2gray(vidFrames2_1(:,:,:,j)),[1,307200]),new);
end
avgframe = new./numFrames;
for j = 1:numFrames
dif(j,:) = reshape(rgb2gray(vidFrames2_1(:,:,:,j)),[1,307200])-avgframe;
end
%%
X = vidFrames2_1(:,:,:,1);
X_gray = rgb2gray(X);
imshow(X_gray)
ginput(1)
```

```
%% find greatest difference between frames
temprows2_1 = zeros();
tempcols2_1 = zeros();
temprows2_1(1) = 266;
tempcols2_1(1) = 317;
for i = 2:numFrames
    X = vidFrames2_1(:,:,:,i);
    X_gray = rgb2gray(X);
    temp = find(dif(i,:) == max(abs(dif(i,:))));
        [rows,cols] = find(X_gray==X_gray(temp(1)));
           count1 = abs(cols-tempcols2_1(i-1));
           count = find(count1 == min(count1));
           [c, rindex] = min(abs(rows(count)-temprows2_1(i-1,:)));
           temprows2_1(i,:) = (rows(count(rindex)));
           tempcols2_1(i,:) = (cols(count(rindex)));
end
%%
for j = 1:size(vidFrames2_1,4)
X = vidFrames2_1(:,:,:,j);
X_gray = rgb2gray(X);
imshow(X);
hold on
plot(tempcols2_1(j),temprows2_1(j),'ro')
pause(0.1)
end
%%
load('cam3_1.mat')
numFrames = size(vidFrames3_1,4);
dif = zeros(1,307200);
for j = 1:numFrames
X = vidFrames3_1(:,:,:,j);
X_gray = rgb2gray(X);
if j ==1
new = reshape(rgb2gray(vidFrames3_1(:,:,:,j)),[1,307200]);
end
new = imadd(reshape(rgb2gray(vidFrames3_1(:,:,:,j)),[1,307200]),new);
end
avgframe = new./numFrames;
for j = 1:numFrames
dif(j,:) = reshape(rgb2gray(vidFrames3_1(:,:,:,j)),[1,307200])-avgframe;
end
%%
X = vidFrames3_1(:,:,:,1);
X_gray = rgb2gray(X);
imshow(X_gray)
ginput(1)
%% find greatest difference between frames
temprows3_1 = zeros();
tempcols3_1 = zeros();
temprows3_1(1) = 271;
tempcols3_1(1) = 319;
for i = 2:numFrames
    X = vidFrames3_1(:,:,:,i);
    X_gray = rgb2gray(X);
    temp = find(dif(i,:) == max(abs(dif(i,:))));
        [rows,cols] = find(X_gray==X_gray(temp(1)));
           count1 = abs(rows-temprows3_1(i-1));
```

```
            count = find(count1 == min(count1));
            [c, rindex] = min(abs(cols(count)-tempcols3_1(i-1,:)));
            temprows3_1(i,:) = (rows(count(rindex)));
            tempcols3_1(i,:) = (cols(count(rindex)));
end
%%
for j = 1:numFrames
X = vidFrames3_1(:,:,:,j);
X_gray = rgb2gray(X);
imshow(X);
hold on
plot(tempcols3_1(j),temprows3_1(j),'ro')
pause(0.1)
end
%% Test 2
load('cam1_2.mat')
numFrames = size(vidFrames1_2,4);
dif = zeros(1,307200);
for j = 1:numFrames
X = vidFrames1_2(:,:,:,j);
X_gray = rgb2gray(X);
if j ==1
new = reshape(rgb2gray(vidFrames1_2(:,:,:,j)),[1,307200]);
end
new = imadd(reshape(rgb2gray(vidFrames1_2(:,:,:,j)),[1,307200]),new);
end
avgframe = new./numFrames;
for j = 1:numFrames
dif(j,:) = reshape(rgb2gray(vidFrames1_2(:,:,:,j)),[1,307200])-avgframe;
end
%%
X = vidFrames1_2(:,:,:,1);
X_gray = rgb2gray(X);
imshow(X_gray)
ginput(1)
%% find greatest difference between frames
temprows1_2 = zeros();
tempcols1_2 = zeros();
temprows1_2(1) = 309;
tempcols1_2(1) = 326;
for i = 2:numFrames
    X = vidFrames1_2(:,:,:,i);
    X_gray = rgb2gray(X);
    temp = find(dif(i,:) == max(abs(dif(i,:))));
        [rows,cols] = find(X_gray==X_gray(temp(1)));
           count1 = abs(cols-tempcols1_2(i-1));
           count = find(count1 == min(count1));
           [c, rindex] = min(abs(rows(count)-temprows1_2(i-1,:)));
           temprows1_2(i,:) = (rows(count(rindex)));
           tempcols1_2(i,:) = (cols(count(rindex)));
end
%%
for j = 1:size(vidFrames1_2,4)
X = vidFrames1_2(:,:,:,j);
X_gray = rgb2gray(X);
imshow(X);
hold on
```

```
plot(tempcols1_2(j),temprows1_2(j),'ro')
pause(0.1)
end
%%
load('cam2_2.mat')
numFrames = size(vidFrames2_2,4);
dif = zeros(1,307200);
for j = 1:numFrames
X = vidFrames2_2(:,:,:,j);
X_gray = rgb2gray(X);
if j ==1
new = reshape(rgb2gray(vidFrames2_2(:,:,:,j)),[1,307200]);
end
new = imadd(reshape(rgb2gray(vidFrames2_2(:,:,:,j)),[1,307200]),new);
end
avgframe = new./numFrames;
for j = 1:numFrames
dif(j,:) = reshape(rgb2gray(vidFrames2_2(:,:,:,j)),[1,307200])-avgframe;
end
%%
X = vidFrames2_2(:,:,:,1);
X_gray = rgb2gray(X);
imshow(X_gray)
ginput(1)
%% find greatest difference between frames
temprows2_2 = zeros();
tempcols2_2 = zeros();
temprows2_2(1) = 361;
tempcols2_2(1) = 317;
for i = 2:numFrames
    X = vidFrames2_2(:,:,:,i);
    X_gray = rgb2gray(X);
    temp = find(dif(i,:) == max(abs(dif(i,:))));
        [rows,cols] = find(X_gray==X_gray(temp(1)));
           count1 = abs(cols-tempcols2_2(i-1));
           count = find(count1 == min(count1));
           [c, rindex] = min(abs(rows(count)-temprows2_2(i-1,:)));
           temprows2_2(i,:) = (rows(count(rindex)));
           tempcols2_2(i,:) = (cols(count(rindex)));
end
%%
for j = 1:numFrames
X = vidFrames2_2(:,:,:,j);
X_gray = rgb2gray(X);
imshow(X);
hold on
plot(tempcols2_2(j),temprows2_2(j),'ro')
pause(0.1)
end
%%
load('cam3_2.mat')
numFrames = size(vidFrames3_2,4);
dif = zeros(1,307200);
for j = 1:numFrames
X = vidFrames3_2(:,:,:,j);
X_gray = rgb2gray(X);
if j ==1
```

```
new = reshape(rgb2gray(vidFrames3_2(:,:,:,j)),[1,307200]);
end
new = imadd(reshape(rgb2gray(vidFrames3_2(:,:,:,j)),[1,307200]),new);
end
avgframe = new./numFrames;
for j = 1:numFrames
dif(j,:) = reshape(rgb2gray(vidFrames3_2(:,:,:,j)),[1,307200])-avgframe;
end
%%
X = vidFrames3_2(:,:,:,1);
X_gray = rgb2gray(X);
imshow(X_gray)
ginput(1)
%% find greatest difference between frames
temprows3_2 = zeros();
tempcols3_2 = zeros();
temprows3_2(1) = 246;
tempcols3_2(1) = 350;
for i = 2:numFrames
    X = vidFrames3_2(:,:,:,i);
    X_gray = rgb2gray(X);
    temp = find(dif(i,:) == max(abs(dif(i,:))));
        [rows,cols] = find(X_gray==X_gray(temp(1)));
           count1 = abs(rows-temprows3_2(i-1));
           count = find(count1 == min(count1));
           [c, rindex] = min(abs(cols(count)-tempcols3_2(i-1,:)));
           temprows3_2(i,:) = (rows(count(rindex)));
           tempcols3_2(i,:) = (cols(count(rindex)));
end
%% Test 3
load('cam1_3.mat')
numFrames = size(vidFrames1_3,4);
dif = zeros(1,307200);
%find the mean
for j = 1:numFrames
X = vidFrames1_3(:,:,:,j);
X_gray = rgb2gray(X);
if j ==1
new = reshape(rgb2gray(vidFrames1_3(:,:,:,j)),[1,307200]);
end
new = imadd(reshape(rgb2gray(vidFrames1_3(:,:,:,j)),[1,307200]),new);
end
avgframe = new./numFrames;
for j = 1:numFrames
dif(j,:) = reshape(rgb2gray(vidFrames1_3(:,:,:,j)),[1,307200])-avgframe;
end
%%
X = vidFrames1_3(:,:,:,1);
X_gray = rgb2gray(X);
imshow(X_gray)
ginput(1)
%% find greatest difference between frames
temprows1_3 = zeros();
tempcols1_3 = zeros();
temprows1_3(1) = 306;
tempcols1_3(1) = 324;
for i = 2:numFrames
```

```
    X = vidFrames1_3(:,:,:,i);
    X_gray = rgb2gray(X);
    temp = find(dif(i,:) == max(abs(dif(i,:))));
        [rows,cols] = find(X_gray==X_gray(temp(1)));
           count1 = abs(cols-tempcols1_3(i-1));
           count = find(count1 == min(count1));
           [c, rindex] = min(abs(rows(count)-temprows1_3(i-1,:)));
           temprows1_3(i,:) = (rows(count(rindex)));
           tempcols1_3(i,:) = (cols(count(rindex)));
end
%%
for j = 1:numFrames
X = vidFrames1_3(:,:,:,j);
X_gray = rgb2gray(X);
imshow(X);
hold on
plot(tempcols1_3(j),temprows1_3(j),'ro')
pause(0.1)
end
%%
load('cam2_3.mat')
numFrames = size(vidFrames2_3,4);
dif = zeros(1,307200);
%find the mean
for j = 1:numFrames
X = vidFrames2_3(:,:,:,j);
X_gray = rgb2gray(X);
if j ==1
new = reshape(rgb2gray(vidFrames2_3(:,:,:,j)),[1,307200]);
end
new = imadd(reshape(rgb2gray(vidFrames2_3(:,:,:,j)),[1,307200]),new);
end
avgframe = new./numFrames;
for j = 1:numFrames
dif(j,:) = reshape(rgb2gray(vidFrames2_3(:,:,:,j)),[1,307200])-avgframe;
end
%%
X = vidFrames2_3(:,:,:,1);
X_gray = rgb2gray(X);
imshow(X_gray)
ginput(1)
%% find greatest difference between frames
temprows2_3 = zeros();
tempcols2_3 = zeros();
temprows2_3(1) = 296;
tempcols2_3(1) = 239;
for i = 2:numFrames
    X = vidFrames2_3(:,:,:,i);
    X_gray = rgb2gray(X);
    temp = find(dif(i,:) == max(abs(dif(i,:))));
        [rows,cols] = find(X_gray==X_gray(temp(1)));
           count1 = abs(cols-tempcols2_3(i-1));
           count = find(count1 == min(count1));
           [c, rindex] = min(abs(rows(count)-temprows2_3(i-1,:)));
           temprows2_3(i,:) = (rows(count(rindex)));
           tempcols2_3(i,:) = (cols(count(rindex)));
end
```

```
%%
for j = 1:numFrames
X = vidFrames2_3(:,:,:,j);
X_gray = rgb2gray(X);
imshow(X);
hold on
plot(tempcols2_3(j),temprows2_3(j),'ro')
pause(0.1)
end
%%
load('cam3_3.mat')
numFrames = size(vidFrames3_3,4);
dif = zeros(1,307200);
%find the mean
for j = 1:numFrames
X = vidFrames3_3(:,:,:,j);
X_gray = rgb2gray(X);
if j ==1
new = reshape(rgb2gray(vidFrames3_3(:,:,:,j)),[1,307200]);
end
new = imadd(reshape(rgb2gray(vidFrames3_3(:,:,:,j)),[1,307200]),new);
end
avgframe = new./numFrames;
for j = 1:numFrames
dif(j,:) = reshape(rgb2gray(vidFrames3_3(:,:,:,j)),[1,307200])-avgframe;
end
%%
X = vidFrames3_3(:,:,:,1);
X_gray = rgb2gray(X);
imshow(X_gray)
ginput(1)
%% find greatest difference between frames
temprows3_3 = zeros();
tempcols3_3 = zeros();
temprows3_3(1) = 233;
tempcols3_3(1) = 357;
for i = 2:numFrames
    X = vidFrames3_3(:,:,:,i);
    X_gray = rgb2gray(X);
    temp = find(dif(i,:) == max(abs(dif(i,:))));
        [rows,cols] = find(X_gray==X_gray(temp(1)));
           count1 = abs(rows-temprows3_3(i-1));
           count = find(count1 == min(count1));
           [c, rindex] = min(abs(cols(count)-tempcols3_3(i-1,:)));
           temprows3_3(i,:) = (rows(count(rindex)));
           tempcols3_3(i,:) = (cols(count(rindex)));
end
%% Test 4
load('cam1_4.mat')
numFrames = size(vidFrames1_4,4);
dif = zeros(1,307200);
for j = 1:numFrames
X = vidFrames1_4(:,:,:,j);
X_gray = rgb2gray(X);
if j ~=1
dif(j,:) = reshape(rgb2gray(vidFrames1_4(:,:,:,j)) -
rgb2gray(vidFrames1_4(:,:,:,j-1)),[1,307200]);
```

```
end
end
%%
X = vidFrames1_4(:,:,:,1);
X_gray = rgb2gray(X);
imshow(X_gray)
ginput(1)
%% find greatest difference between frames
temprows1_4 = zeros();
tempcols1_4 = zeros();
temprows1_4(1) = 305;
tempcols1_4(1) = 376;
for i = 2:numFrames
    X = vidFrames1_4(:,:,:,i);
    X_gray = rgb2gray(X);
    temp = find(dif(i,:) == max(abs(dif(i,:))));
        [rows,cols] = find(X_gray==X_gray(temp(1)));
           count1 = abs(cols-tempcols1_4(i-1));
           count = find(count1 == min(count1));
           [c, rindex] = min(abs(rows(count)-temprows1_4(i-1,:)));
           [d, cindex] = min(abs(cols(count)-tempcols1_4(i-1,:)));
           temprows1_4(i,:) = (rows(count(rindex)));
           tempcols1_4(i,:) = (cols(count(cindex)));
end
%%
for j = 1:numFrames
X = vidFrames1_4(:,:,:,j);
X_gray = rgb2gray(X);
imshow(X);
hold on
plot(tempcols1_4(j),temprows1_4(j),'ro')
pause(0.1)
end
%%
load('cam2_4.mat')
numFrames = size(vidFrames2_4,4);
dif = zeros(1,307200);
for j = 1:numFrames
X = vidFrames2_4(:,:,:,j);
X_gray = rgb2gray(X);
if j ~=1
dif(j,:) = reshape(rgb2gray(vidFrames2_4(:,:,:,j)) -
rgb2gray(vidFrames2_4(:,:,:,j-1)),[1,307200]);
end
end
%%
X = vidFrames2_4(:,:,:,1);
X_gray = rgb2gray(X);
imshow(X_gray)
ginput(1)
%% find greatest difference between frames
temprows2_4 = zeros();
tempcols2_4 = zeros();
temprows2_4(1) = 244;
tempcols2_4(1) = 244;
for i = 2:numFrames
    X = vidFrames2_4(:,:,:,i);
```

```
    X_gray = rgb2gray(X);
    temp = find(dif(i,:) == max(abs(dif(i,:))));
        [rows,cols] = find(X_gray==X_gray(temp(1)));
           count1 = abs(cols-tempcols2_4(i-1));
           count = find(count1 == min(count1));
           [c, rindex] = min(abs(rows(count)-temprows2_4(i-1,:)));
           [d, cindex] = min(abs(cols(count)-tempcols2_4(i-1,:)));
           temprows2_4(i,:) = (rows(count(rindex)));
           tempcols2_4(i,:) = (cols(count(cindex)));
end
%%
for j = 1:numFrames
X = vidFrames2_4(:,:,:,j);
X_gray = rgb2gray(X);
imshow(X);
hold on
plot(tempcols2_4(j),temprows2_4(j),'ro')
pause(0.1)
end
%%
load('cam3_4.mat')
numFrames = size(vidFrames3_4,4);
dif = zeros(1,307200);
for j = 1:numFrames
X = vidFrames3_4(:,:,:,j);
X_gray = rgb2gray(X);
if j ~=1
dif(j,:) = reshape(rgb2gray(vidFrames3_4(:,:,:,j)) -
rgb2gray(vidFrames3_4(:,:,:,j-1)),[1,307200]);
end
end
%%
X = vidFrames3_4(:,:,:,1);
X_gray = rgb2gray(X);
imshow(X_gray)
ginput(1)
%% find greatest difference between frames
temprows3_4 = zeros();
tempcols3_4 = zeros();
temprows3_4(1) = 208;
tempcols3_4(1) = 399;
for i = 2:numFrames
    X = vidFrames3_4(:,:,:,i);
    X_gray = rgb2gray(X);
    temp = find(dif(i,:) == max(abs(dif(i,:))));
        [rows,cols] = find(X_gray==X_gray(temp(1)));
           count1 = abs(rows-temprows3_4(i-1));
           count = find(count1 == min(count1));
           [c, rindex] = min(abs(rows(count)-temprows3_4(i-1,:)));
           [d, cindex] = min(abs(cols(count)-tempcols3_4(i-1,:)));
           temprows3_4(i,:) = (rows(count(rindex)));
           tempcols3_4(i,:) = (cols(count(cindex)));
end
%%
for j = 1:numFrames
X = vidFrames3_4(:,:,:,j);
X_gray = rgb2gray(X);
```

```
imshow(X);
hold on
plot(tempcols3_4(j),temprows3_4(j),'ro')
pause(0.1)
end
%% Creating snapshot matrix and peforming the SVD on all tests
%% Test 1
figure(1)
plot(1:length(temprows1_1),temprows1_1,'k-')
figure(2)
plot(1:length(temprows2_1),temprows2_1,'k-')
figure(3)
plot( 1:length(tempcols3_1),tempcols3_1,'k-')
%120 points: camera 1, x=11:131 camera 2: x = 19:139 camera 3: x = 9:129
parsed1_1 = [tempcols1_1(13:133) temprows1_1(13:133)];
parsed2_1 = [tempcols2_1(19:139) temprows2_1(19:139)];
parsed3_1 = [tempcols3_1(9:129) temprows3_1(9:129)];
snapshot_1 = [parsed1_1.';parsed2_1.';parsed3_1.'];
%subtracting the mean from each row
snapshot_1 = snapshot_1 -
[mean(snapshot_1(1,:));mean(snapshot_1(2,:));mean(snapshot_1(3,:));mean(snaps
hot_1(4,:));mean(snapshot_1(5,:));mean(snapshot_1(6,:))];
[U_1,S_1,V_1] = svd(snapshot_1,'econ');
sig1 = diag(S_1);
%% Test 2
figure(1)
plot(1:length(temprows1_2),temprows1_2,'k-')
figure(2)
plot(1:length(temprows2_2),temprows2_2,'k-')
figure(3)
plot(1:length(tempcols3_2),tempcols3_2,'k-')
% 71 points: camera 1, x=1:71 camera 2: x = 1:71 camera 3: x = 1:71
parsed1_2 = [tempcols1_2(1:71) temprows1_2(1:71)];
parsed2_2 = [tempcols2_2(1:71) temprows2_2(1:71)];
parsed3_2 = [tempcols3_2(1:71) temprows3_2(1:71)];
snapshot_2 = [parsed1_2.';parsed2_2.';parsed3_2.'];
snapshot_2= snapshot_2 -
[mean(snapshot_2(1,:));mean(snapshot_2(2,:));mean(snapshot_2(3,:));mean(snaps
hot_2(4,:));mean(snapshot_2(5,:));mean(snapshot_2(6,:))];
[U_2,S_2,V_2] = svd(snapshot_2,'econ');
sig2 = diag(S_2);
%% Test 3
figure(1)
plot(1:length(temprows1_3),temprows1_3,'k-')
figure(2)
plot(1:length(temprows2_3),temprows2_3,'k-')
figure(3)
plot( 1:length(tempcols3_3),tempcols3_3,'k-')
%104 points: 1:104
parsed1_3 = [tempcols1_3(1:21) temprows1_3(1:21)];
parsed2_3 = [tempcols2_3(1:21) temprows2_3(1:21)];
parsed3_3 = [tempcols3_3(1:21) temprows3_3(1:21)];
snapshot_3 = [parsed1_3.';parsed2_3.';parsed3_3.'];
snapshot_3 = snapshot_3 -
[mean(snapshot_3(1,:));mean(snapshot_3(2,:));mean(snapshot_3(3,:));mean(snaps
hot_3(4,:));mean(snapshot_3(5,:));mean(snapshot_3(6,:))];
[U_3,S_3,V_3] = svd(snapshot_3,'econ');
```

```
sig3 = diag(S_3);
%% Test 4
figure(1)
plot(1:length(temprows1_4),temprows1_4,'k-')
figure(2)
plot(1:length(temprows2_4),temprows2_4,'k-')
figure(3)
plot( 1:length(tempcols3_4),tempcols3_4,'k-')
%48 points: 23:71
parsed1_4 = [tempcols1_4(6:54) temprows1_4(6:54)];
parsed2_4 = [tempcols2_4(7:55) temprows2_4(7:55)];
parsed3_4 = [tempcols3_4(23:71) temprows3_4(23:71)];
snapshot_4 = [parsed1_4.';parsed2_4.';parsed3_4.'];
snapshot_4 = snapshot_4 -
[mean(snapshot_4(1,:));mean(snapshot_4(2,:));mean(snapshot_4(3,:));mean(snaps
hot_4(4,:));mean(snapshot_4(5,:));mean(snapshot_4(6,:))];
[U_4,S_4,V_4] = svd(snapshot_4,'econ');
sig4 = diag(S_4);
%% Plotting energies
subplot(2,2,1)
semilogy(sig1.^2/sum(sig1.^2),'ro')
title('Test 1 Singular Value Energies')
xlabel('Mode')
ylabel('Energy')
subplot(2,2,2)
semilogy(sig2.^2/sum(sig2.^2),'ro')
title('Test 2 Singular Value Energies')
xlabel('Mode')
ylabel('Energy')
subplot(2,2,3)
semilogy(sig3.^2/sum(sig3.^2),'ro')
title('Test 3 Singular Value Energies')
xlabel('Mode')
ylabel('Energy')
subplot(2,2,4)
semilogy(sig4.^2/sum(sig4.^2),'ro')
title('Test 4 Singular Value Energies')
xlabel('Mode')
ylabel('Energy')
%% plotting x- and y-coordinates
subplot(2,2,1)
plot(1:length(temprows1_1),temprows1_1,'r')
hold on
plot(1:length(temprows2_1),temprows2_1,'b')
plot(1:length(temprows3_1),temprows3_1,'g')
title('Test 1')
xlabel('Time')
ylabel('Vertical Position')
legend('Camera 1', 'Camera 2', 'Camera 3')
subplot(2,2,2)
plot(1:length(temprows1_2),temprows1_2,'r')
hold on
plot(1:length(temprows2_2),temprows2_2,'b')
plot(1:length(temprows3_2),temprows3_2,'g')
title('Test 2')
xlabel('Time')
ylabel('Vertical Position')
```

```
legend('Camera 1', 'Camera 2', 'Camera 3')
subplot(2,2,3)
plot(1:length(temprows1_3),temprows1_3,'r')
hold on
plot(1:length(temprows2_3),temprows2_3,'b')
plot(1:length(temprows3_3),temprows3_3,'g')
title('Test 3')
xlabel('Time')
ylabel('Vertical Position')
legend('Camera 1', 'Camera 2', 'Camera 3')
subplot(2,2,4)
plot(1:length(temprows1_4),temprows1_4,'r')
hold on
plot(1:length(temprows2_4),temprows2_4,'b')
plot(1:length(temprows3_4),temprows3_4,'g')
title('Test 4')
xlabel('Time')
ylabel('Vertical Position')
legend('Camera 1', 'Camera 2', 'Camera 3')
%%
subplot(2,2,1)
plot(1:length(tempcols1_1),tempcols1_1,'r')
hold on
plot(1:length(tempcols2_1),tempcols2_1,'b')
plot(1:length(tempcols3_1),tempcols3_1,'g')
title('Test 1')
xlabel('Time')
ylabel('Horizontal Position')
legend('Camera 1', 'Camera 2', 'Camera 3')
subplot(2,2,2)
plot(1:length(tempcols1_2),tempcols1_2,'r')
hold on
plot(1:length(tempcols2_2),tempcols2_2,'b')
plot(1:length(tempcols3_2),tempcols3_2,'g')
title('Test 2')
xlabel('Time')
ylabel('Horizontal Position')
legend('Camera 1', 'Camera 2', 'Camera 3')
subplot(2,2,3)
plot(1:length(tempcols1_3),tempcols1_3,'r')
hold on
plot(1:length(tempcols2_3),tempcols2_3,'b')
plot(1:length(tempcols3_3),tempcols3_3,'g')
title('Test 3')
xlabel('Time')
ylabel('Horizontal Position')
legend('Camera 1', 'Camera 2', 'Camera 3')
subplot(2,2,4)
plot(1:length(tempcols1_4),tempcols1_4,'r')
hold on
plot(1:length(tempcols2_4),tempcols2_4,'b')
plot(1:length(tempcols3_4),tempcols3_4,'g')
title('Test 4')
xlabel('Time')
ylabel('Horizontal Position')
legend('Camera 1', 'Camera 2', 'Camera 3')
```