AMATH 482 Homework 3:
PCA and a Spring-Mass System

Malikah Nathani
February 24th, 2021

Abstract

This paper examines the practical implementation of the PCA algorithm. Specifically, by examining the displacement of a spring-mass system given videos taken of its motion from three different cameras at distinct angles. We compare the extracted principal components found in four test cases with varying levels of noise and movements to understand the motions present and ultimately explore the impact noise and the true usefulness of the PCA.

1 Introduction and Overview

Principal Component Analysis (PCA) is a vast area of study with numerous applications ranging from quantitative finance to neuroscience. In the context of this paper, we will be applying this data analysis method to the realm of physics, specifically motion. Objects are constantly moving around us, yet, true analysis can come from understanding what motions are present.

For this problem, we are tasked to use the PCA to find the principal motion(s) of a paint can attached to the end of a rope. The data we are given are three videos of the spring-mass system's movement from three unique camera angles. We need to first determine the x and y coordinates of the mass at every frame for each of the videos, then use the PCA to detangle our data allowing us to understand the principal motions presented in the data. We are given videos and asked to complete this process for four test cases: ideal, noisy, horizontal displacement, and horizontal displacement and rotation of the mass. Through comparing and contrasting the results, we can garner a better understanding of the PCA.

In Section 2, I will go about explaining the PCA in more detail while section 3 will go over the implementation of this method and the algorithm used to solve the problem. Sections 4 and 5 will go over the computed results and my conclusions respectively. Finally, I will also present the MATLAB functions used and provide my code.

2 Theoretical Background

2.1 Principal Component Analysis (PCA)

After extracting the x and y coordinates, to understand the principal motions present we need to employ the PCA algorithm. The main idea behind the PCA is to apply a Singular Value Decomposition (SVD) and transform a large dataset to one without redundancies thus obtaining an orthogonal frame of reference of the data.

To ensure that when we calculate our correlations it is at a meaningful level, we will first subtract the mean from our dataset thus we have data with zero mean. This ensures that our results are unbiased as differences between the range of values in separate variables will be omitted, and our variables will be on the same scale. To compute the covariances and variances within the rows of our matrix, we use the matrix multiplication formula:

$$C_X = \frac{1}{n-1} XX^T$$

Having an orthogonal frame of reference, means that the columns of the new dataset are uncorrelated (i.e. covariance equals zero). This ensures that each variable contains new information. To understand which variables contain the most important information about our data, it is important to find the largest variances. This is done by diagonalizing the matrix C_X such that all the covariances (off-diagonal elements) are zero.

$$C_X = V\Lambda V^{-1}$$

Where the principal components are the basis of eigenvectors in the matrix V. The variances of the variables are along the diagonal entries of Λ.

In our calculations, we use the SVD. However, these methods are connected since the SVD of a matrix A is calculated with the eigenvalues and eigenvectors of the matrix AA^T. This is the same as what we used above with the matrix X as:

$$C_X = \frac{1}{n-1} XX^T = AA^T$$

Since we consider:

$$A = \frac{1}{\sqrt{n-1}} X$$

Therefore, when we diagonalize to find our principal components, we are doing the same thing by finding the SVD of the matrix A.

3 Algorithm Implementation and Development

This section details the implementations of the methods above and any other steps taken to solve this problem in MATLAB. For the actual code used, please see Appendix B. An important note is that I will go over the implementation for one test, as the process is simply replicated for each case.

First, we load in the three videos of the mass at the three unique angles. One main part of this assignment is to track the position of the paint can; we do this by determining the x and y coordinates of the bright spot on top of our mass at each frame. In order to make this bright spot more vivid as well as eliminate the color dimension, we convert each frame to a grayscale. To

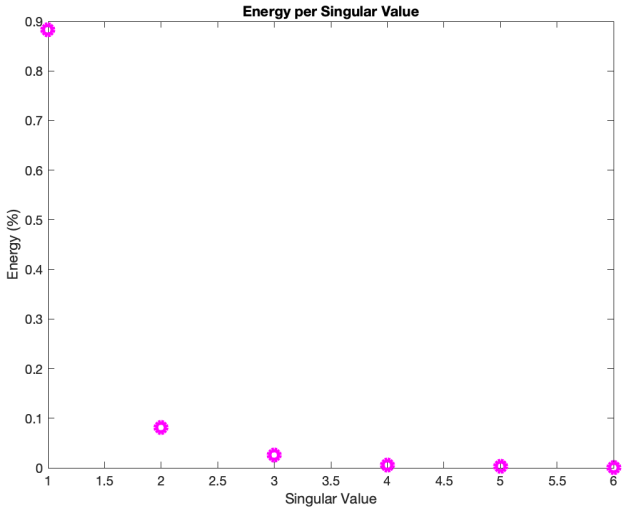
keep track of the coordinates, we initialize a x and y vector with its length being the number of frames in that video.

The brightest pixel has the largest magnitude, therefore for each frame we use the `max` command to find the highest valued pixel, then the `ind2sub` command to get its x and y coordinates. To ensure that MATLAB is solely tracking the bright spot, we can create a Shannon filter centered around the x and y coordinates of the paint can found in the frame before. Since we assume the movement of the mass between each frame in any direction will not be over a 30-pixel threshold, we obtain a 60 x 60-pixel frame that blocks out any other noise. Therefore, guaranteeing the bright spot on top of our paint can is the one detected. We find the initial center of the Shannon filter using the `ginput` function, which allows us to find the general idea of the initial mass position.

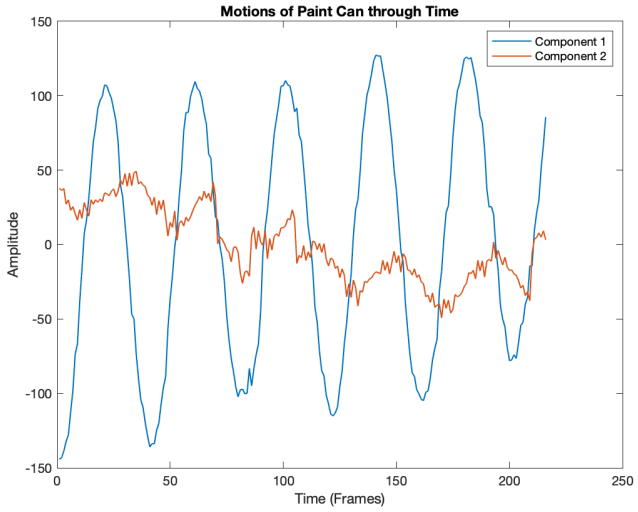
After repeating this process for each of the three videos, it is seen that the videos have a slightly different length thus the vectors lengths also differ. Therefore, it is important to crop the coordinate vectors such that they are all of uniform length and the movement of the mass described in the vectors is synchronized. Meaning, that the general motion of the paint can along the frames is consistent for the vectors.

Afterwards, we add our vectors to a matrix A, while subtracting the dataset by its mean to ensure we have zero mean. Then, we calculate the SVD of the matrix by using the built-in MATLAB function, remember to use `'econ'` along with the function to rid the matrix of any extra rows and columns of zeros. To find the energy of each of the singular values, we divide that singular value squared over the sum of all the squared singular values.

4 Computational Results

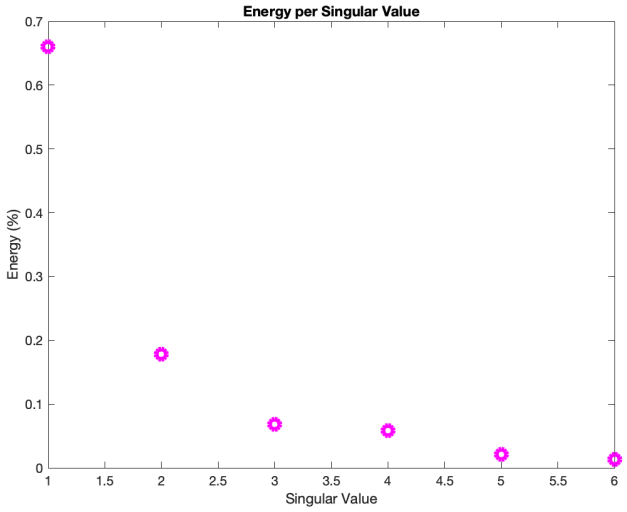


(a) Energies of Singular Values

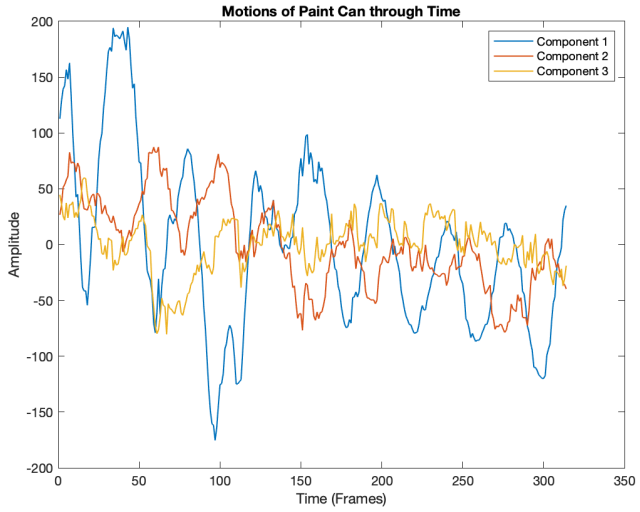


(b) Motions of first two basis components through time, scaled by corresponding singular value

Figure 1 Ideal Case

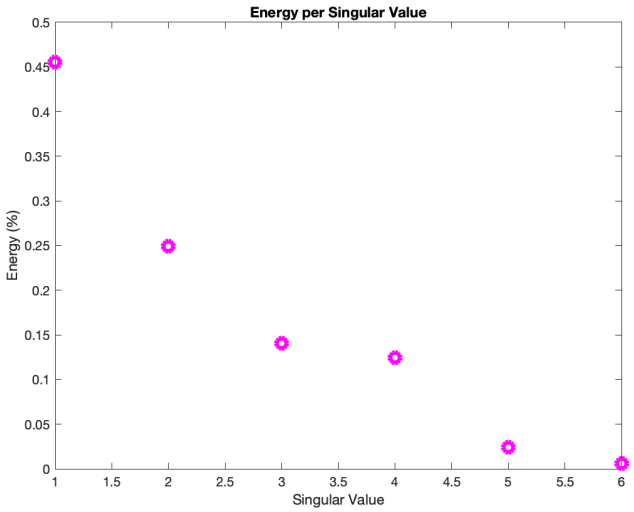


(a) Energies of Singular Values

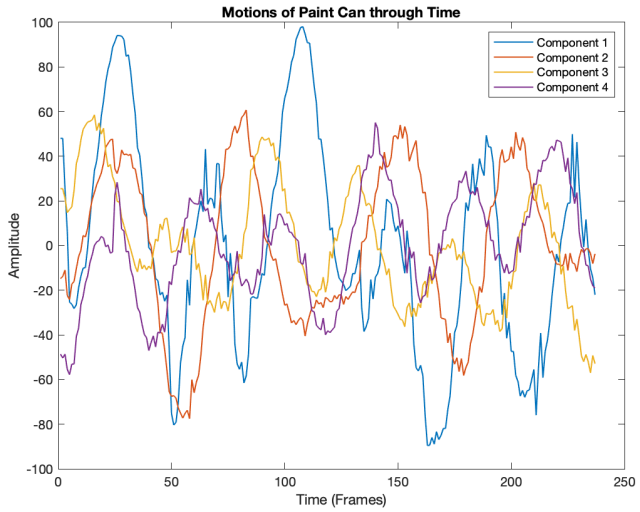


(b) Motions of first three basis components through time, scaled by corresponding singular value

Figure 2 Noisy Case

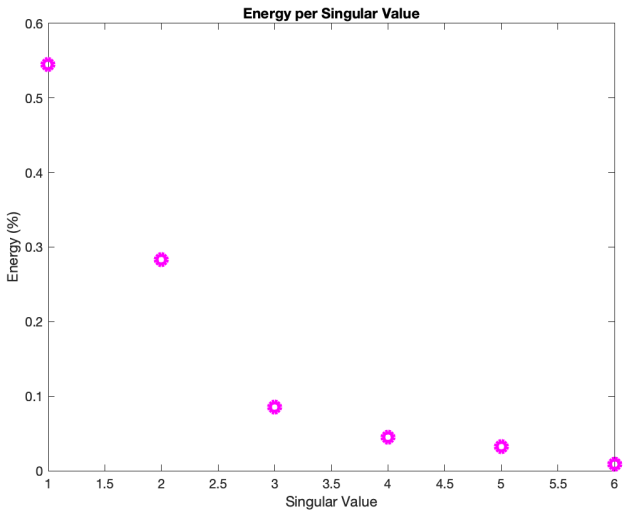


(a) Energies of Singular Values

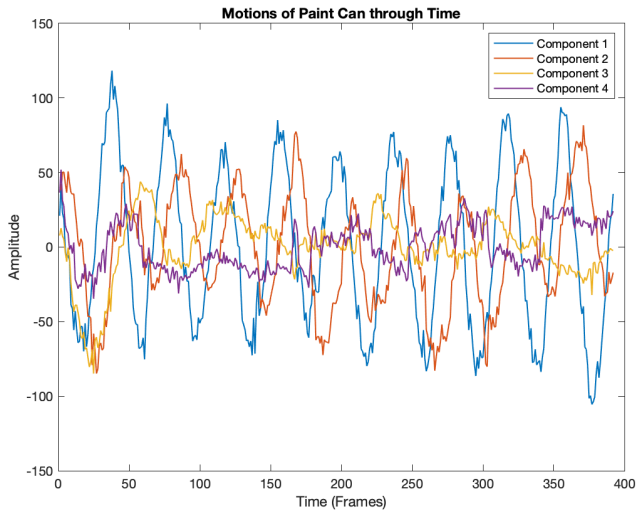


(b) Motions of first four basis components through time, scaled by corresponding singular value

Figure 3 Horizontal Displacement Case



(a) Energies of Singular Values



(b) Motions of first four basis components through time, scaled by corresponding singular value

Figure 4 Horizontal Displacement and Rotation Case

5 Summary and Conclusions

In Test 1, our data showed the outcome we mostly expected. We knew the actual data should be one dimensional as the only movement of the spring-mass system is up and down. This idea is reflected in Figure 1a as the first singular component holds almost 90% of energy. Therefore, showing that the SVD found the most variance of the data for one motion, which we know is the vertical displacement. When plotting the first two columns of the vector V in Figure 1b, our first component has a smooth oscillating pattern whereas the second component is very jagged and has no pattern. Thus, again showing that our first component is capturing most of the motion presented in the dataset.

In Test 2, we have introduced camera shakes which makes our dataset quite noisy. This noise can be seen in the PCA analysis by looking at Figure 2a, now, our first singular value only encompasses about 65% energy and the second singular component is much more prominent than in Test 1 having almost 20% energy. Since we have camera shakes, we have motion in the up and down direction as well as extra movements due to the camera. Our PCA analysis shows this as our SVD found variance in mainly two components. When plotting the first three motions in Figure 2b, we still see component 1 being larger than the other components suggesting that one motion captured was greater. Our component two's amplitude seems larger than it was in Test 1 as well as there is more of an oscillating pattern shown indicating there is another principal motion being observed. Component 3 seems to not have a clear pattern. All the lines for each component seem much more jagged than in Test 1 perhaps due to all the noise present in the data.

In Test 3, we introduced horizontal motion of the spring-mass system along with the already present vertical displacement. It is interesting to see that in Figure 3a, our SVD has given four singular values with energies over 10%. Our first singular component's energy has been significantly reduced from Test 1 to about 46% while singular values 3 & 4 have energies closer

to 15%. In Figure 3b, we can see that our first component still has the greatest amplitude, similar to Test 1. However, in this test our lines seem more jagged than perhaps indicating some noise present. In this specific test, our spring-mass system also moved in the horizontal direction. Therefore, we see that as the SVD has observed up more unique motions for this spring-mass system. Similarly, we can see this idea in Figure 3b as components two & three have oscillating motions that have decently large amplitudes; in Test 1, it is clear that the amplitude for component two is quite small and has no pattern.

Finally, in Test 4, our spring-mass system had motion in the horizontal and vertical direction while the can was also rotating. In Figure 4a, we can see three main singular values. Compared to Test 1, the energy for the first singular value dropped to about 55% in this test, whereas the second and third singular values increased to having about 30% and 10% energy respectively. Using Figure 4b, we see component one still has the largest amplitude, which can perhaps again be contributed to the most prominent motion of the mass in the vertical direction. The line for component one had somewhat smooth curves though not as smooth as Test 1. These differences in singular value energies as well as the oscillating movements and large amplitudes for components two and three in Figure 4b can be attributed to the PCA picking up other principal motions in the data, perhaps the horizontal and rotational movements. Component three and four seem to be quite jagged while component four also does not have clear oscillations.

Using PCA and SVD on a real-world dataset is not perfect and we can see that with the different test cases. Similarly, we may not completely understand why the SVD may be picking up different motions present and what direction it may represent for the spring-mass system. Therefore, although the PCA is an incredibly powerful tool, it does have its limitations that we are more aware of due to this analysis. However, with further study we can understand the PCA at a deeper level and implement this tool to more unique applications.

Appendix A. MATLAB functions used

- `sz = size(A):` returns a row vector whose elements are the lengths of the corresponding dimensions of A.
- `[x,y] = ginput(n):` allows you to identify the coordinates of n points.
- `[M, I] = max(X):` returns the max value as well as the corresponding index in the operating dimension
- `[row, col] = ind2sub(sz, I)`: returns the arrays `row` and `col` containing the equivalent row and column subscripts corresponding to the linear indices `I` for a matrix of size `sz`.
- `[U,S,V] = svd(A, 'econ'):` performs a singular value economy-size decomposition of matrix A, such that A = U*S*V' and removes extra rows or columns of zeros from the diagonal matrix of singular values.
- `D = diag(V):` returns a square diagonal matrix with the elements of vector v on the main diagonal.
- `plot(X, Y):` creates a 2-D line plot of the data in Y versus the corresponding values in X.

Appendix B. MATLAB codes

```
%%
load('cam1_1.mat');
load('cam2_1.mat');
load('cam3_1.mat');
%%
X = vidFrames1_1(:,:,:,1);
X = rgb2gray(X);
```

```
Imshow(X);
[x,y] = ginput(1);
%%
numFrames = size(vidFrames1_1,4);
x1 = zeros(1, numFrames);
y1 = zeros(1, numFrames);
fx = 220;
fy = 300;
for j = 1:numFrames
    X = vidFrames1_1(:,:,:,j);
    Xg = rgb2gray(X);
    Xg = im2double(Xg);
    filter = zeros(480,640);
    filter(((fx-30):(fx+30)), ((fy-30):(fy+30))) = 1;
    Xgf = Xg.*filter;
    [M,I] = max(Xgf(:));
    [x1(j),y1(j)] = ind2sub(size(Xgf), I);
    fx = x1(j);
    fy = y1(j);
    %imshow(Xgf); drawnow;
end

numFrames = size(vidFrames2_1,4);
x2 = zeros(1, numFrames);
y2 = zeros(1, numFrames);
fx = 276;
fy = 278;
for j = 1:numFrames
    X = vidFrames2_1(:,:,:,j);
    Xg = rgb2gray(X);
    Xg = im2double(Xg);
    filter = zeros(480,640);
    filter(((fx-30):(fx+30)), ((fy-30):(fy+30))) = 1;
    Xgf = Xg.*filter;
    [M,I] = max(Xgf(:));
    [x2(j),y2(j)] = ind2sub(size(Xgf), I);
    fx = x2(j);
    fy = y2(j);
    %imshow(Xgf); drawnow;
end

numFrames = size(vidFrames3_1,4);
x3 = zeros(1, numFrames);
y3 = zeros(1, numFrames);
fx = 273;
fy = 329;
for j = 1:numFrames
    X = vidFrames3_1(:,:,:,j);
    Xg = rgb2gray(X);
    Xg = im2double(Xg);
    filter = zeros(480,640);
    filter(((fx-30):(fx+30)), ((fy-30):(fy+30))) = 1;
    Xgf = Xg.*filter;
    [M,I] = max(Xgf(:));
    [x3(j),y3(j)] = ind2sub(size(Xgf), I);
    fx = x3(j);
    fy = y3(j);
    %imshow(Xgf); drawnow;
```

```
end
%%
x1 = x1(:, 11:226);
y1 = y1(:, 11:226);
x2 = x2(:, 19:234);
y2 = y2(:, 19:234);
x3 = x3(:, 8:223);
y3 = y3(:, 8:223);

A = [x1 - mean(x1); y1 - mean(y1); x2 - mean(x2); y2 - mean(y2); x3 -
mean(x3); y3 - mean(y3)];

[U,S,V] = svd(A, 'econ');
sig = diag(S);
energies = zeros(1, length(sig));
for j = 1:length(sig)
    energies(j) = sig(j)^2/sum(sig.^2);
end
%%
figure(1)
plot(energies, 'mo', 'LineWidth', 4, 'MarkerSize', 8)
xlabel('Singular Value')
ylabel('Energy (%)')
title('Energy per Singular Value')
print('T1-1.png', '-dpng');

figure(2)
plot(sig(1:2)'.*V(:,1:2),'LineWidth', 1)
legend('Component 1', 'Component 2')
xlabel('Time (Frames)')
ylabel('Amplitude')
title('Motions of Paint Can through Time')
print('T1-2.png', '-dpng');
%%
load('cam1_2.mat');
load('cam2_2.mat');
load('cam3_2.mat');
%%
numFrames = size(vidFrames1_2,4);
x1 = zeros(1, numFrames);
y1 = zeros(1, numFrames);
fx = 328;
fy = 285;
for j = 1:numFrames
    X = vidFrames1_2(:,:,:,j);
    Xg = rgb2gray(X);
    Xg = im2double(Xg);
    filter = zeros(480,640);
    filter(((fx-30):(fx+30)), ((fy-30):(fy+30))) = 1;
    Xgf = Xg.*filter;
    [M,I] = max(Xgf(:));
    [x1(j),y1(j)] = ind2sub(size(Xgf), I);
    fx = x1(j);
    fy = y1(j);
    %imshow(Xgf); drawnow;
end
```

```
numFrames = size(vidFrames2_2,4);
x2 = zeros(1, numFrames);
y2 = zeros(1, numFrames);
fx = 316;
fy = 356;
for j = 1:numFrames
    X = vidFrames2_2(:,:,:,j);
    Xg = rgb2gray(X);
    Xg = im2double(Xg);
    filter = zeros(480,640);
    filter(((fx-30):(fx+30)), ((fy-30):(fy+30))) = 1;
    Xgf = Xg.*filter;
    [M,I] = max(Xgf(:));
    [x2(j),y2(j)] = ind2sub(size(Xgf), I);
    fx = x2(j);
    fy = y2(j);
    %imshow(Xgf); drawnow;
end

numFrames = size(vidFrames3_2,4);
x3 = zeros(1, numFrames);
y3 = zeros(1, numFrames);
fx = 244;
fy = 350;
for j = 1:numFrames
    X = vidFrames3_2(:,:,:,j);
    Xg = rgb2gray(X);
    Xg = im2double(Xg);
    filter = zeros(480,640);
    filter(((fx-30):(fx+30)), ((fy-30):(fy+30))) = 1;
    Xgf = Xg.*filter;
    [M,I] = max(Xgf(:));
    [x3(j),y3(j)] = ind2sub(size(Xgf), I);
    fx = x3(j);
    fy = y3(j);
    %imshow(Xgf); drawnow;
end
%%
x2 = x2(:, 1:314);
y2 = y2(:, 1:314);
x3 = x3(:, 1:314);
y3 = y3(:, 1:314);

A = [x1 - mean(x1); y1 - mean(y1); x2 - mean(x2); y2 - mean(y2); x3 -
mean(x3); y3 - mean(y3)];

[U,S,V] = svd(A, 'econ');
sig = diag(S);
energies = zeros(1, length(sig));
for j = 1:length(sig)
    energies(j) = sig(j)^2/sum(sig.^2);
end

figure(1)
plot(energies, 'mo', 'LineWidth', 4, 'MarkerSize', 8)
xlabel('Singular Value')
ylabel('Energy (%)')
```

```
title('Energy per Singular Value')
print('T2-1.png', '-dpng');

figure(2)
plot(sig(1:3)'.*V(:,1:3),'LineWidth', 1)
legend('Component 1', 'Component 2', 'Component 3')
xlabel('Time (Frames)')
ylabel('Amplitude')
title('Motions of Paint Can through Time')
print('T2-2.png', '-dpng');

%%
load('cam1_3.mat');
load('cam2_3.mat');
load('cam3_3.mat');
%%
numFrames = size(vidFrames1_3,4);
x1 = zeros(1, numFrames);
y1 = zeros(1, numFrames);
fx = 285;
fy = 290;
for j = 1:numFrames
    X = vidFrames1_3(:,:,:,j);
    Xg = rgb2gray(X);
    Xg = im2double(Xg);
    filter = zeros(480,640);
    filter(((fx-30):(fx+30)), ((fy-30):(fy+30))) = 1;
    Xgf = Xg.*filter;
    [M,I] = max(Xgf(:));
    [x1(j),y1(j)] = ind2sub(size(Xgf), I);
    fx = x1(j);
    fy = y1(j);
    %imshow(X); drawnow;
end

numFrames = size(vidFrames2_3,4);
x2 = zeros(1, numFrames);
y2 = zeros(1, numFrames);
fx = 238;
fy = 292;
for j = 1:numFrames
    X = vidFrames2_3(:,:,:,j);
    Xg = rgb2gray(X);
    Xg = im2double(Xg);
    filter = zeros(480,640);
    filter(((fx-30):(fx+30)), ((fy-30):(fy+30))) = 1;
    Xgf = Xg.*filter;
    [M,I] = max(Xgf(:));
    [x2(j),y2(j)] = ind2sub(size(Xgf), I);
    fx = x2(j);
    fy = y2(j);
    %imshow(Xgf); drawnow;
end

numFrames = size(vidFrames3_3,4);
x3 = zeros(1, numFrames);
y3 = zeros(1, numFrames);
```

```
fx = 228;
fy = 354;
for j = 1:numFrames
    X = vidFrames3_3(:,:,:,j);
    Xg = rgb2gray(X);
    Xg = im2double(Xg);
    filter = zeros(480,640);
    filter(((fx-30):(fx+30)), ((fy-30):(fy+30))) = 1;
    Xgf = Xg.*filter;
    [M,I] = max(Xgf(:));
    [x3(j),y3(j)] = ind2sub(size(Xgf), I);
    fx = x3(j);
    fy = y3(j);
   % imshow(Xgf); drawnow;
end
%%
x1 = x1(:, 1:237);
y1 = y1(:, 1:237);
x2 = x2(:, 1:237);
y2 = y2(:, 1:237);

A = [x1 - mean(x1); y1 - mean(y1); x2 - mean(x2); y2 - mean(y2); x3 -
mean(x3); y3 - mean(y3)];

[U,S,V] = svd(A, 'econ');
sig = diag(S);
energies = zeros(1, length(sig));
for j = 1:length(sig)
    energies(j) = sig(j)^2/sum(sig.^2);
end

figure(1)
plot(energies, 'mo', 'LineWidth', 4, 'MarkerSize', 8)
xlabel('Singular Value')
ylabel('Energy (%)')
title('Energy per Singular Value')
print('T3-1.png', '-dpng');

figure(2)
plot(sig(1:4)'.*V(:,1:4),'LineWidth', 1)
legend('Component 1', 'Component 2', 'Component 3', 'Component 4')
xlabel('Time (Frames)')
ylabel('Amplitude')
title('Motions of Paint Can through Time')
print('T3-2.png', '-dpng');
%%
load('cam1_4.mat');
load('cam2_4.mat');
load('cam3_4.mat');
%%
numFrames = size(vidFrames1_4,4);
x1 = zeros(1, numFrames);
y1 = zeros(1, numFrames);
fx = 272;
fy = 382;
for j = 1:numFrames
    X = vidFrames1_4(:,:,:,j);
```

```
    Xg = rgb2gray(X);
    Xg = im2double(Xg);
    filter = zeros(480,640);
    filter(((fx-30):(fx+30)), ((fy-30):(fy+30))) = 1;
    Xgf = Xg.*filter;
    [M,I] = max(Xgf(:));
    [x1(j),y1(j)] = ind2sub(size(Xgf), I);
    fx = x1(j);
    fy = y1(j);
    %imshow(Xgf); drawnow;
end

numFrames = size(vidFrames2_4,4);
x2 = zeros(1, numFrames);
y2 = zeros(1, numFrames);
fx = 252;
fy = 230;
for j = 1:numFrames
    X = vidFrames2_4(:,:,:,j);
    Xg = rgb2gray(X);
    Xg = im2double(Xg);
    filter = zeros(480,640);
    filter(((fx-30):(fx+30)), ((fy-30):(fy+30))) = 1;
    Xgf = Xg.*filter;
    [M,I] = max(Xgf(:));
    [x2(j),y2(j)] = ind2sub(size(Xgf), I);
    fx = x2(j);
    fy = y2(j);
    %imshow(Xgf); drawnow;
end

numFrames = size(vidFrames3_4,4);
x3 = zeros(1, numFrames);
y3 = zeros(1, numFrames);
fx = 214;
fy = 364;
for j = 1:numFrames
    X = vidFrames3_4(:,:,:,j);
    Xg = rgb2gray(X);
    Xg = im2double(Xg);
    filter = zeros(480,640);
    filter(((fx-30):(fx+30)), ((fy-30):(fy+30))) = 1;
    Xgf = Xg.*filter;
    [M,I] = max(Xgf(:));
    [x3(j),y3(j)] = ind2sub(size(Xgf), I);
    fx = x3(j);
    fy = y3(j);
    %imshow(Xgf); drawnow;
end
%%
x2 = x2(:, 1:392);
y2 = y2(:, 1:392);
x3 = x3(:, 1:392);
y3 = y3(:, 1:392);

A = [x1 - mean(x1); y1 - mean(y1); x2 - mean(x2); y2 - mean(y2); x3 -
mean(x3); y3 - mean(y3)];
```

```
[U,S,V] = svd(A, 'econ');
sig = diag(S);
energies = zeros(1, length(sig));
for j = 1:length(sig)
    energies(j) = sig(j)^2/sum(sig.^2);
end

figure(1)
plot(energies, 'mo', 'LineWidth', 4, 'MarkerSize', 8)
xlabel('Singular Value')
ylabel('Energy (%)')
title('Energy per Singular Value')
print('T4-1.png', '-dpng');

figure(2)
plot(sig(1:4)'.*V(:,1:4),'LineWidth', 1)
legend('Component 1', 'Component 2', 'Component 3', 'Component 4')
xlabel('Time (Frames)')
ylabel('Amplitude')
title('Motions of Paint Can through Time')
print('T4-2.png', '-dpng');
```