Yes We (Paint) Can: Analysis of Weighted Object Movement Using Singular Value Decomposition

Kristen Drummey

Department of Physiology and Biophysics, University of Washington

Github: github.com/kldrummey/AMATH582

Abstract

Real-world datasets are often noisy, redundant, and sub-optimally acquired. Singular value decomposition (SVD) and principal component analysis (PCA) are tools that can be applied to noisy, redundant data in order to determine the underlying dynamics in a system. In this study, an oscillating paint can was recorded using three different camera angles and with varying amounts of noise and oscillation. The path of the paint can was tracked by determining the brightest pixel in each frame, which can be attributed to a flashlight placed on top of the can. Trajectory data was then transformed using SVD and examined. This study shows the usefulness of dimensionality reduction in pulling meaning out of variable datasets.

1 Introduction and Overview

Data collection is difficult and messy. Scientists are often limited by the tools that they have available and data collection is typically accompanied by unwanted noise and redundancy. These limitations can make it difficult to establish the true underlying dynamics of the system being measured. Mathematical tools such as singular value decomposition (SVD) and principal components analysis (PCA) can eliminate some of the noise and redundancy in sampled data and give a clearer picture of the underlying dynamics of a system. In this study, a weighted object in the form of a paint can was attached to a spring and was recorded from three different angles. There were four total iterations of paint can recordings:

1. The paint can was moved straight up and down vertically, and camera recordings were steady.
2. The paint can was moved straight up and down vertically, and the camera recordings were shaky, introducing noise.
3. The paint can was moved straight up and down vertically, and displaced horizontally.
4. The paint can displaced off-center, introducing vertical movement, rotation, and horizontal displacement.

In order to determine the dynamics of the paint can in each video set, the paint can was tracked based on pixel intensity of the brightest spot in each frame (attributed to a flashlight placed on top of the paint can). Trajectory information was then transformed using SVD and projections were determined to show the underlying dynamics of the system.

2 Theoretical Background

SVD builds on the foundational tenets of linear algebra, in that it fundamentally rotates and stretches the components in a matrix. For example, let's consider a dataset s with coordinates $v_1...v_n$. Let's say we multiply s by a matrix A, which stretches s by factors $\sigma_1...\sigma_n$ in the directions $u_1...u_n$. Since V is a unitary matrix, this can be simplified to give us the equation:

$$A = \hat{U}\hat{\Sigma}V^* \tag{1}$$

where $\hat{\Sigma}$ is a diagonal matrix and $\hat{U}$ is a matrix with orthonormal columns. The power of SVD over other methods of diagonalization lie in the fact that it diagonalizes using two bases, U and V, and that it exists for any matrix A, which is in stark opposition to typical eigenvalue decomposition.

Additionally, the SVD is uniquely powerful in producing low rank approximations, or reducing the dimensionality in highly complex datasets to make them easier to examine. This is particularly useful for determining the principal components of a dataset, or which axes explain the most amount of the variance in the data and therefore provide some insight into the data's underlying dynamics. Although principal components are often calculated using diagonlization with eigenvectors, it can also be calculated using SVD. Take for example the transformed variable:

$$Y = U^*X \tag{2}$$

where U is the unitary transformation. Using this, we can determine the variance in Y:

$$\begin{aligned} C &= \frac{1}{n-1}YY^T \\ &= \frac{1}{n-1}(U^*X)(U^*X)^T \\ &= \frac{1}{n-1}U^*(XX^T)U \\ &= \frac{1}{n-1}U^*U\Sigma^2UU^* \\ C &= \frac{1}{n-1}\Sigma^2 \end{aligned}$$

and therefore determine the principal components and their projection.

3 Algorithm Implementation and Development

Video data was stored as a .mat file and initially written to a .avi to allow for visual inspection of the paint can movement.

Algorithm 1: Conversion of .mat data to .avi movies

`load` camN-N.mat
Create `VideoWriter` object and open
Use `writeVideo` to write .mat data to .avi file and close `VideoWriter` object.

After inspection of video and determination that the brightest pixel intensity for each frame was most likely due to the flashlight on top of the paint can, coordinates were determined for the brightest pixel in each frame. Videos were converted to grayscale, which has a pixel intensity range from 0 (darkest) to 255 (brightest). The brightest pixels were determined to be those above 230-240 in intensity.

Algorithm 2: Determination of paint can trajectory based on brightest pixels

for `j=(1:number of frames in shortest video)` **do**
  convert RGB video frames to grayscale
  find coordinates of pixels greater than 255
  assign coordinate values to X and Y vectors for each video
end for

This created six vectors total: an X and Y vector for each of the three videos taken for each condition. These vectors were then used in the SVD.

Algorithm 3: SVD analysis of paint can trajectory

Combine all six vectors into one matrix,A
Determine and subtract the mean
Use `[U,S,V]=svd(A)` to determine the SVD matrices.
Compute the diagonal to give the variance/energy in each mode.

4 Computational Results

Figure 1 shows the results from analyzing the ideal condition, in which there was limited to no noise in recording and the paint can was only moving vertically. Figure 1A shows the raw trajectory of the paint can as determined by tracking the position of the brightest pixels, assumed to be coming from the flashlight placed on top of the can. About 30% of the variance was explained by the first mode, with modes 2-5 explaining between 15-20% of the variance and mode 6 explaining almost none of the variance. The trajectories of the paint can as determined by projections onto the SVD modes are shown in full in Figure 1C, and for the first two modes in figure 1D. All of the modes do appear to show an oscillating displacement, as would be expected based on visual inspection of the video and the raw path of the paint can shown in Figure 1A.

Figure 2 shows the results from analyzing a noisy condition, in which the paint can was still only displaced vertically but the cameras used to record were shaking, introducing noise into the measurements. Even with the additional noise, the SVD analysis was able to roughly determine the expected path of the paint can, as shown in Figures 2C and 2D. Each mode captured less of the variance than those for the ideal condition, as shown in Figures 1B and 2B. In the noisy condition, the first mode, which captures most of the variance, only captured about 25%, less than the amount captured by the 30% captured by the first mode in the ideal condition.

The results from analyzing the paint can's trajectory as it was displaced both vertically and horizontally are shown in Figure 3. The raw trajectory shows increased displacement horizontally, as would be expected. Interestingly, unlike in the ideal and noisy conditions previously described, the vast majority of the variance is captured in mode 1. The first mode captures nearly 50% of the variance. Mode 2, which captures the next greatest amount of the variance, captures dramatically less than mode 1, at about 10%. The projections of the paint can trajectory do appear to show some horizontal movement, although the projection onto mode 6 shows large deviations from projections onto mode 1-5.

Finally, the analysis of the paint can's trajectory when displaced vertically while rotating horizontally are shown in Figure 4. As with the horizontal displacement case, the vast amount of the variance is captured by mode 1, at about 43%. The projections onto each mode do appear to show some oscillating behavior that could be attributed to the paint can rotating, although the projections are difficult to distinguish from those in Figures 3C and 3D, in which the paint can was horizontally displaced with no rotation.

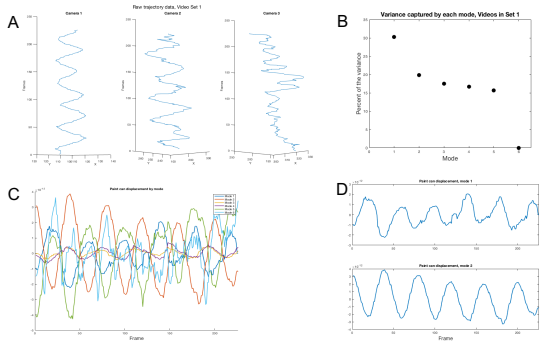

Figure 1: Analysis of videos in the ideal condition. A) Raw trajectories determined by tracking brightest pixels in the image shows an oscillating vertical path in each camera. B) Percent of the variance explained by each mode. The largest percentage of the variance is explained by mode 1, at about 30%. C) and D) show projections of the paint can trajectory according to mode, with C showing all projections and D showing projections onto the first two modes, which combined explain approximately 50% of the variance.

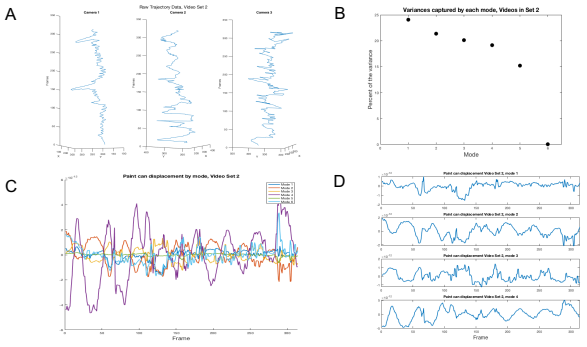

Figure 2: Analysis of videos in the noisy condition. A) Raw trajectories show a less refined path than that seen in Fig. 1, likely due to increased noise from the shaking camera. B) Percent of the variance explained by each mode. The largest percentage of the variance is explained by mode 1, at about 25%. C) and D) show projections of the paint can trajectory according to mode, with C) showing all projections and D) showing individual projections onto the first four modes.

5 Summary and Conclusions

This study shows that using SVD to determine principal components is a uniquely valuable tool in examining unknown underlying dynamics of a system. SVD analysis was able to fairly accurately reconstruct the trajectory of the paint can from video recorded at only three different angles. Increasing the number of recording angles, improving paint can tracking by using a more sophisticated metric than pixel brightness, and better accounting for horizontal displacement would likely yield even better results. Even with these limitations, SVD proves to be a robust method for determining principal components and projection axes for this data. This suggests that SVD is a powerful mechanism for looking at many data types, particularly those where the underlying dynamics of the system are unknown.

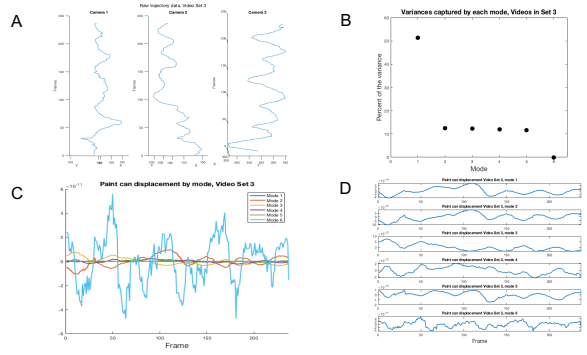

Figure 3: Analysis of videos in with vertical and horizontal displacement. A) Raw trajectories show increased movement horizontally by the paint can. B) Percent of the variance explained by each mode. The largest percentage of the variance is explained by mode 1, at nearly 50%. C) and D) show projections of the paint can trajectory according to mode, with C) showing all projections together and D) showing projections broken out by mode.

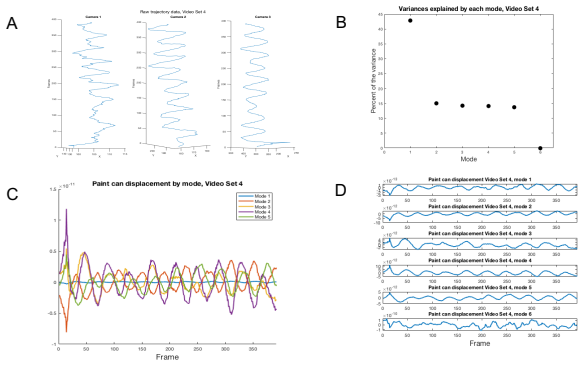

Figure 4: Analysis of videos with horizontal and vertical displacement and oscillation in the z-direction. A) Raw trajectories show increased oscillatory movement by the paint can. B) Percent of the variance explained by each mode. The largest percentage of the variance is explained by mode 1, at about 45%. C) and D) show projections of the paint can trajectory according to mode, with C) showing all projections together and D) showing projections broken out by mode.

6 References

1. Kutz, J.N. (2013). Data-Driven Modeling Scientific Computation: Methods for Complex Systems and Big Data. Chapter 15. Oxford University Press. 1st. Ed.

Appendix A MATLAB Functions

Notable functions used in this analysis:

- `VideoWriter` creates an object to write a video file to.
- `writeVideo` writes video data to a file. In the code in Appendix B, writeVideo was used to write the .mat camera recording files to the object created by VideoWriter, allowing for viewing of the videos.
- `double(X)` Converts data in X from its original data type to the double data type.
- `size(X)` Determines the size of X, or how many rows and columns X has.
- `find(X)` Finds the values in X that correspond to the parameters given (e.g. find(X ¿ 250) finds values in X that are greater than 250).
- `rgb2gray` Converts RGB color images/frames to grayscale.
- `mean(X)` Determines the mean of X.
- `[U S V]=svd(X)` Takes the singular value decomposition of X and returns the corresponding U, S, and V matrices.
- `diag(X)` Returns the values that are on the diagonal of the matrix X.
- `sprintf` Formats input data to return it as a string of text.

Appendix B MATLAB Code

```
%% AMATH582 - Homework 3

%% Ideal Case - Entire motion is in the z direction. camN_1.mat, where N=1,2,3
close all; clear all; clc;

load cam1_1.mat; load cam2_1.mat; load cam3_1.mat

%Convert variables to videos to visually inspect paint can trajectory
```

```
v1_1=VideoWriter('vid1_1.avi'); open(v1_1); writeVideo(v1_1,vidFrames1_1); close(v1_1);
v2_1=VideoWriter('vid2_1.avi'); open(v2_1); writeVideo(v2_1,vidFrames2_1); close(v2_1);
v3_1=VideoWriter('vid3_1.avi'); open(v3_1); writeVideo(v3_1,vidFrames3_1); close(v3_1);

%Convert uint8 to double, determine shortest video to set frame limit
vid11=double(vidFrames1_1); vid21=double(vidFrames2_1); vid31=double(vidFrames3_1);
vidsize1=size(vid11); vidsize2=size(vid21); vidsize3=size(vid31);
vidn1lengths=[size(vid11,4); size(vid21,4); size(vid31,4)];
framesn1=1:min(vidn1lengths);

%Set up X_n and Y_n matrices and assign values based on brightest pixels
X1=zeros(1, length(framesn1)); X2=X1; X3=X1; Y1=X1; Y2=X1; Y3=X1;

for j=1:length(X1)
    v11gray=rgb2gray(vidFrames1_1(:,:,:,j));
    [x1 y1]=find(v11gray> 240);
    X1(j)=mean(x1); Y1(j)=mean(y1);
    v21gray=rgb2gray(vidFrames2_1(:,:,:,j));
    [x2 y2]=find(v21gray>240);
    X2(j)=mean(x2); Y2(j)=mean(y2);
    v31gray=rgb2gray(vidFrames3_1(:,:,:,j));
    [x3 y3]=find(v31gray>240);
    X3(j)=mean(x3); Y3(j)=mean(y3);
end

%Plot X-Y trajectory of the paint can in each video
figure(1)
subplot(1,3,1); plot3(X1,Y1,framesn1); title('Camera 1','Fontsize',[12]); xlabel('X'); ylabel('Y'); zlab
subplot(1,3,2); plot3(X2,Y2,framesn1); title('Camera 2','Fontsize',[12]); xlabel('X'); ylabel('Y'); zlab
subplot(1,3,3); plot3(X3,Y3,framesn1); title('Camera 3','Fontsize',[12]); xlabel('X'); ylabel('Y'); zlab
sgtitle('Raw trajectory data, Video Set 1','Fontsize',[16]);

%SVD analysis on trajectories
n1_matrix=[X1;X2;X3;Y1;Y2;Y3];
[a1,b1]=size(n1_matrix);
ab=mean(n1_matrix,2);
n1_matrix=n1_matrix-repmat(ab,1,b1);
[u1,s1,v1]=svd(n1_matrix/sqrt(n1_matrix-b1)); %SVD
lambda=diag(s1).^2; %diagonal variances
proj1=u1'*n1_matrix;

figure(2)
plot(lambda/sum(lambda)*100,'ko','MarkerSize',[10],'MarkerFaceColor','k'); xlim([0 6.5]);
title('Variance captured by each mode, Videos in Set 1','Fontsize',[16]);
ylabel('Percent of the variance','Fontsize',[16]); xlabel('Mode','Fontsize',[16]);

figure(3)
for j=1:6
    hold on
    plot(framesn1,proj1(j,:),'LineWidth',2)
    title('Paint can displacement by mode')
    legend('Mode 1','Mode 2','Mode 3','Mode 4','Mode 5','Mode 6'); xlim([0 length(framesn1)]);
end
xlabel('Frame','Fontsize',[16]);
```

```
figure(4)
for j=1:2
    subplot(2,1,j)
    plot(framesn1,proj1(j,:),'LineWidth',2)
    title(sprintf('Paint can displacement, mode %d',j)); xlim([0 length(framesn1)]);
end
xlabel('Frame','Fontsize',[16])
%% Noisy Case - Camera is shaking. camN_2.mat, where N=1,2,3
close all; clear all; clc;

load cam1_2.mat; load cam2_2.mat; load cam3_2.mat

%Convert variables to videos to visually inspect paint can trajectory
v1_2=VideoWriter('vid1_2.avi'); open(v1_2); writeVideo(v1_2,vidFrames1_2); close(v1_2);
v2_2=VideoWriter('vid2_2.avi'); open(v2_2); writeVideo(v2_2,vidFrames2_2); close(v2_2);
v3_2=VideoWriter('vid3_2.avi'); open(v3_2); writeVideo(v3_2,vidFrames3_2); close(v3_2);

%Convert uint8 to double, determine shortest video to set frame limit
vid12=double(vidFrames1_2); vid22=double(vidFrames2_2); vid32=double(vidFrames3_2);
vidsize12=size(vid12); vidsize22=size(vid22); vidsize32=size(vid32);
vidn2lengths=[size(vid12,4); size(vid22,4); size(vid32,4)];
framesn2=1:min(vidn2lengths);

%Set up X_n and Y_n matrices and assign values based on brightest pixels
X12=zeros(1, length(framesn2)); X22=X12; X32=X12; Y12=X12; Y22=X12; Y32=X12;

for j=1:length(X12)
    v12gray=rgb2gray(vidFrames1_2(:,:,:,j));
    [x12 y12]=find(v12gray> 240);
    X12(j)=mean(x12); Y12(j)=mean(y12);
    v22gray=rgb2gray(vidFrames2_2(:,:,:,j));
    [x22 y22]=find(v22gray>240);
    X22(j)=mean(x22); Y22(j)=mean(y22);
    v32gray=rgb2gray(vidFrames3_2(:,:,:,j));
    [x32 y32]=find(v32gray>240);
    X32(j)=mean(x32); Y32(j)=mean(y32);
end

%Plot X-Y trajectory of the paint can in each video
figure(1)
subplot(1,3,1); plot3(X12,Y12,framesn2); title('Camera 1','Fontsize',[12]); xlabel('X'); ylabel('Y'); zl
subplot(1,3,2); plot3(X22,Y22,framesn2); title('Camera 2','Fontsize',[12]); xlabel('X'); ylabel('Y'); zl
subplot(1,3,3); plot3(X32,Y32,framesn2); title('Camera 3','Fontsize',[12]); xlabel('X'); ylabel('Y'); zl
sgtitle('Raw Trajectory Data, Video Set 2','Fontsize',[16])

%SVD analysis on trajectories
n2_matrix=[X12;X22;X32;Y12;Y22;Y32];
[a2,b2]=size(n2_matrix);
ab2=mean(n2_matrix,2);
n2_matrix=n2_matrix-repmat(ab2,1,b2);
[u2,s2,v2]=svd(n2_matrix/sqrt(n2_matrix-b2)); %SVD
lambda2=diag(s2).^2; %diagonal variances
proj2=u2'*n2_matrix;
```

```
figure(2)
plot(lambda2/sum(lambda2)*100,'ko','MarkerSize',[10],'MarkerFaceColor','k'); xlim([0 6.5]);
title('Variances captured by each mode, Videos in Set 2','Fontsize',[16]);
ylabel('Percent of the variance','Fontsize',[16]); xlabel('Mode','Fontsize',[16]);

figure(3)
for j=1:6
    hold on
    plot(framesn2,proj2(j,:),'LineWidth',2)
    title('Paint can displacement by mode, Video Set 2','Fontsize',[14])
    legend('Mode 1','Mode 2','Mode 3','Mode 4','Mode 5','Mode 6'); xlim([0 length(framesn2)]);
end
xlabel('Frame','Fontsize',[16]);

figure(4)
for j=1:4
    subplot(4,1,j)
    plot(framesn2,proj2(j,:),'LineWidth',2)
    title(sprintf('Paint can displacement Video Set 2, mode %d',j)); xlim([0 length(framesn2)]);
end
xlabel('Frame','Fontsize',[16])

%% Horizontal Displacement - Motion in x, y, z directions. camN_3.mat, where N=1,2,3

close all; clear all; clc;

load cam1_3.mat; load cam2_3.mat; load cam3_3.mat

%Convert variables to videos to visually inspect paint can trajectory
v1_3=VideoWriter('vid1_1.avi'); open(v1_3); writeVideo(v1_3,vidFrames1_3); close(v1_3);
v2_3=VideoWriter('vid2_1.avi'); open(v2_3); writeVideo(v2_3,vidFrames2_3); close(v2_3);
v3_3=VideoWriter('vid3_1.avi'); open(v3_3); writeVideo(v3_3,vidFrames3_3); close(v3_3);

%Convert uint8 to double, determine shortest video to set frame limit
vid13=double(vidFrames1_3); vid23=double(vidFrames2_3); vid33=double(vidFrames3_3);
vidsize13=size(vid13); vidsize23=size(vid23); vidsize33=size(vid33);
vidn3lengths=[size(vid13,4); size(vid23,4); size(vid33,4)];
framesn3=1:min(vidn3lengths);

%Set up X_n and Y_n matrices and assign values based on brightest pixels
X13=zeros(1, length(framesn3)); X23=X13; X33=X13; Y13=X13; Y23=X13; Y33=X13;

for j=1:length(X13)
    v13gray=rgb2gray(vidFrames1_3(:,:,:,j));
    [x13 y13]=find(v13gray>230);
    X13(j)=mean(x13); Y13(j)=mean(y13);
    v23gray=rgb2gray(vidFrames2_3(:,:,:,j));
    [x23 y23]=find(v23gray>230);
    X23(j)=mean(x23); Y23(j)=mean(y23);
    v33gray=rgb2gray(vidFrames3_3(:,:,:,j));
    [x33 y33]=find(v33gray>230);
    X33(j)=mean(x33); Y33(j)=mean(y33);
end
```

```
%Plot X-Y-Z trajectory of the paint can in each video
figure(1)
subplot(1,3,1); plot3(X13,Y13,framesn3); title('Camera 1','Fontsize',[12]); xlabel('X'); ylabel('Y'); zl
subplot(1,3,2); plot3(X23,Y23,framesn3); title('Camera 2','Fontsize',[12]); xlabel('X'); ylabel('Y'); zl
subplot(1,3,3); plot3(X33,Y33,framesn3); title('Camera 3','Fontsize',[12]); xlabel('X'); ylabel('Y'); zl
sgtitle('Raw trajectory data, Video Set 3')

%SVD analysis on trajectories
n3_matrix=[X13;X23;X33;Y13;Y23;Y33];
[a3,b3]=size(n3_matrix);
ab3=mean(n3_matrix,2);
n3_matrix=n3_matrix-repmat(ab3,1,b3);
[u3,s3,v3]=svd(n3_matrix/sqrt(n3_matrix-b3)); %SVD
lambda3=diag(s3).^2; %diagonal variances
proj3=u3'*n3_matrix;

figure(2)
plot(lambda3/sum(lambda3)*100,'ko','MarkerSize',[10],'MarkerFaceColor','k'); xlim([0 6.5]);
title('Variances captured by each mode, Videos in Set 3','Fontsize',[16]);
ylabel('Percent of the variance','Fontsize',[16]); xlabel('Mode','Fontsize',[16]);

figure(3)
for j=1:6
    hold on
    plot(framesn3,proj3(j,:),'LineWidth',2)
    title('Paint can displacement by mode, Video Set 3','Fontsize',[14])
    legend('Mode 1','Mode 2','Mode 3','Mode 4','Mode 5','Mode 6'); xlim([0 length(framesn3)]);
end
xlabel('Frame','Fontsize',[16]);

figure(4)
for j=1:6
    subplot(6,1,j)
    plot(framesn3,proj3(j,:),'LineWidth',2)
    title(sprintf('Paint can displacement Video Set 3, mode %d',j)); xlim([0 length(framesn3)]);
end
xlabel('Frame','Fontsize',[16])
%% Horizontal Displacement/Rotation - Pendulum motion and oscillation in z. camN_4.mat, where N = 1,2,3

close all; clear all; clc;

%Load .mat files with video data
load cam1_4.mat; load cam2_4.mat; load cam3_4.mat

%Convert variables to videos to visually inspect paint can trajectory
v1_4=VideoWriter('vid1_4.avi'); open(v1_4); writeVideo(v1_4,vidFrames1_4); close(v1_4);
v2_4=VideoWriter('vid2_4.avi'); open(v2_4); writeVideo(v2_4,vidFrames2_4); close(v2_4);
v3_4=VideoWriter('vid3_4.avi'); open(v3_4); writeVideo(v3_4,vidFrames3_4); close(v3_4);

%Convert uint8 data to double, determine shortest video to set frame limit
vid14=double(vidFrames1_4); vid24=double(vidFrames2_4); vid34=double(vidFrames3_4);
vidsize14=size(vid14); vidsize24=size(vid24); vidsize33=size(vid34);
vidn4lengths=[size(vid14,4); size(vid24,4); size(vid34,4)];
```

```
framesn4=1:min(vidn4lengths);

%Set up X_n and Y_n matrices and assign values based on brightest pixels,
%assuming that converting to grayscale will have pixel values of 0-255,
%with closer to 255 being the brightest.
X14=zeros(1, length(framesn4)); X24=X14; X34=X14; Y14=X14; Y24=X14; Y34=X14;
for j=1:length(X14)
    v14gray=rgb2gray(vidFrames1_4(:,:,:,j));
    [x14 y14]=find(v14gray> 230);
    X14(j)=mean(x14); Y14(j)=mean(y14);
    v24gray=rgb2gray(vidFrames2_4(:,:,:,j));
    [x24 y24]=find(v24gray>230);
    X24(j)=mean(x24); Y24(j)=mean(y24);
    v34gray=rgb2gray(vidFrames3_4(:,:,:,j));
    [x34 y34]=find(v34gray>230);
    X34(j)=mean(x34); Y34(j)=mean(y34);
end

%Plot raw trajectory of the paint can in each video
figure(1)
subplot(1,3,1); plot3(X14,Y14,framesn4); title('Camera 1','Fontsize',[12]); xlabel('X'); ylabel('Y'); zl
subplot(1,3,2); plot3(X24,Y24,framesn4); title('Camera 2','Fontsize',[12]); xlabel('X'); ylabel('Y'); zl
subplot(1,3,3); plot3(X34,Y34,framesn4); title('Camera 3','Fontsize',[12]); xlabel('X'); ylabel('Y'); zl
sgtitle('Raw trajectory data, Video Set 4','Fontsize',[16]);

%SVD analysis on trajectories
n4_matrix=[X14;X24;Y14;Y24;X34;Y34];
[a4,b4]=size(n4_matrix);
ab4=mean(n4_matrix,2);
n4_matrix=n4_matrix-repmat(ab4,1,b4);
[u4,s4,v4]=svd(n4_matrix/sqrt(n4_matrix-b4)); %SVD
lambda4=diag(s4).^2; %diagonal variances
proj4=u4'*n4_matrix;

%Plot % of the variance explained by each component
figure(2)
plot(lambda4/sum(lambda4)*100,'ko','MarkerSize',[10],'MarkerFaceColor','k'); xlim([0 6.5]);
title('Variances explained by each mode, Video Set 4','Fontsize',[16]);
ylabel('Percent of the variance','Fontsize',[16]); xlabel('Mode','Fontsize',[16]);

%Plot modes vs time
figure(3)
for j=1:5
    hold on
    plot(framesn4,proj4(j,:),'LineWidth',2)
    title('Paint can displacement by mode, Video Set 4','Fontsize',[14])
    legend('Mode 1','Mode 2','Mode 3','Mode 4','Mode 5','Mode 6'); xlim([0 length(framesn4)]);
end
xlabel('Frame','Fontsize',[16]);

figure(4)
for j=1:6
    subplot(6,1,j)
    plot(framesn4,proj4(j,:),'LineWidth',2)
```

```
    title(sprintf('Paint can displacement Video Set 4, mode %d',j)); xlim([0 length(framesn4)]);
end
xlabel('Frame','Fontsize',[16])
```