AMATH 582 Homework 3: The Principal Components of a Paint Can in it Various States of Motion Found Using SVD

Gavin M. Chase

February 21, 2020

Abstract

This report seeks to reproduce the principal components of a paint can in motion through Single Variable Decomposition (SVD). It considers several scenarios that include oscillations, horizontal displacement, and rotation. In each case, three videos taken from different angles and with various orientations are used in the principal component analysis (PCA). The elements of the SVD are then used to interpret and reconstruct the paint can's motion.

1 Introduction and Overview

The tasks of evaluating unfamiliar systems and determining low dimensional ways to understand them lie at the heart of data analysis. One way to approach this kind of system is by performing a Principle Component Analysis (PCA) using the Singular Value Decomposition (SVD). PCA seeks to identify and reduce redundancy amongst a set of measurements. This leaves the user with a low number of dynamics of interest from a system[1].

This report seeks to reproduce the the principal dynamics of interest for a paint can suspended from a spring. Each trial is measured with three video cameras oriented differently on the paint can. Trial 1 exhibits displacement in the z-direction only. Trial 2 exhibits displacement in the z-direction while the cameras are shaken to induce noise. Trial 3 exhibits displacement in the z-direction and horizontally. Trial 4 recreates Trial 3 with rotation.

2 Theoretical Background

The SVD, which exists for any matrix $\mathbf{A} \in \mathbb{C}^{mxn}$, is

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

where $\mathbf{V}^*$ is a unitary matrix in which each column holds the unit vectors for each principal component of the new basis, $\mathbf{\Sigma}$ is a matrix whose diagonal elements are the singular values describing the energy or significance of each component, and $\mathbf{U}$ is a unitary matrix describing each principal component in time. The decomposition of $\mathbf{A}$ into $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}^*$ is an ideal basis in which all redundancies have been removed and the largest variances are ordered. Thus, if $\mathbf{A}$ is an original set of data describing a system, a rank 1 approximation of system $\mathbf{A}$ would describe the first and most substantial component elements of $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}^*$ at $\mathbf{U} \in \mathbb{R}^{mx1}$, $\mathbf{\Sigma} \in \mathbb{R}^{1x1}$, and $\mathbf{V}^* \in \mathbb{R}^{nx1}$. Performing a rank 2 approximation would be to combine the first and second most significant components into a single approximation of the system.

$\mathbf{\Sigma}$ is directly related to the variance in $\mathbf{A}$ through the following equation

$$\mathbf{C_A} = \mathbf{A^T A} = \frac{1}{n-1}\mathbf{\Sigma}^2 \tag{2}$$

where n is the number rows in $\mathbf{A}$.

3 Algorithm Implementation and Development

Generically, there are 3 steps for determining the principle components of a data set[1].

1. Organize data into a matrix $\mathbf{A} \in \mathbb{C}^{mxn}$ where m is the number of measurement types and n is the number of measurements taken.
2. Subtract off the mean for each measurement type or row of $\mathbf{A}$.
3. Compute the SVD and singular values of the covariance matrix to determine the principal components.

In the experiment conducted for this report, each row of $\mathbf{A}$ corresponded to a distance x or y in pixel space of a position on the paint can. Thus, $\mathbf{A}$ took the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{x}_{\text{camera 1,Frame } n} \\ \mathbf{y}_{\text{camera 1,Frame } n} \\ \mathbf{x}_{\text{camera 2,Frame } n} \\ \mathbf{y}_{\text{camera 2,Frame } n} \\ \mathbf{x}_{\text{camera 3,Frame } n} \\ \mathbf{y}_{\text{camera 3,Frame } n} \end{bmatrix} \tag{3}$$

The final matrix $\mathbf{A}$ becomes 6xn where n is the number of video frames measured. There are a number of implied tasks for Step 1 from the above list. These tasks require the most effort due to the high risk of skewing the PCA results out of laziness. The SVD is a powerful tool when applied to relatively clean data, but as will be shown in the next section, poor data collection can almost eliminate the utility of this tool. Thus, I took several pre-processing actions to reduce the chances of self-induced error. First, I synchronized the three videos by cutting a few of the initial frames. This ensured that for any column n of $\mathbf{A}$ each row should be detecting the paint can at the same moment in time. Next, I cut video frames at the end of all but the shortest video to ensure all rows of $\mathbf{A}$ were of equal length. Next, I selected a region of the frames that captured the paint can for the duration of the video, and cropped each frame to those dimensions. This reduced the area I needed to operate on to measure the paint can.

After reducing the video data to a more useful, synchronized set, I began tracking the paint can. I applied Algorithm 1 to Trials 1, 2, and 4. On Trial 3, I made a modification to Line 12 of Algorithm 1. Instead of collecting the first nonzero pixel position, I collected the last nonzero pixel position.

Algorithm 1: Measuring x and y Positions of the Paint Can

```
Create empty vectors xposition and yposition
set a pixel strength to screen at
for $j = 1$ :number of frames do
  Isolate Frame $j$ and convert it to gray scale
  for $j = 1$ :number of pixels in $x$-direction do
    for $i = 1$ :number of pixels in $y$-direction do
      if Pixel at position $j, i >$ pixel strength then
        Set pixel value at position $j, i = 255$
      else
        Set pixel value at position $j, i = 0$
      end if
    end for
  end for
  Identify $x$ and $y$ positions of first nonzero pixel
  Add $x$ and $y$ positions to xposition and yposition
end for
Subtract the means for the final rows xposition and yposition from each element of those vectors,
respectively
```

Note that Step 2, subtract the mean, occurs immediately after collecting each row of measurements and is shown as the last step in Algorithm 1. This step further synchronizes measurements by setting the frame of reference relative to the origin of the object's motion and not to the shape or size of the video frames. The final step, computing the svd, is equally as simple, and provides the data needed for PCA. Specifically, for $\mathbf{V}$, each column corresponds to same n^{th} term of $\mathbf{\Sigma}$, and each row m corresponds to the same m^{th} row of measurements in $\mathbf{A}$.

4 Computational Results

4.1 Trial 1

Trial 1 consisted of a paint can oscillating in the z-direction only. Its singular values are shown in Figure 1 with an energy for the rank 1 approximation being 68% and the energy of a rank 3 approximation being 92%. While these energies seem somewhat low, they do accurately capture the motion of the paint can during this trial. A reconstruction of the 6 measurements can be found in Figure 2. Figure 3 depicts how $\mathbf{U}$ and $\mathbf{V}$ act for Trial 1. The rank 1 approximation clearly matches the harmonic motion in the z-direction as expected. The second mode appears to include some noise, which is why the line for $\mathbf{U}_2$ is so jagged, but also captures the small amount of rotation of the paint can in a counterclockwise manner. In the 2D plane of a video frame, this motion is captured as motion from left to right, which translates to a general decrease in value for elements of $\mathbf{U}_2$ across time. The additions of modes 3-6 no longer add new dynamics of interest, but rather try to fit the more accurate low rank approximation to the noisier reality of the data. This is the unfortunate consequence of poorly obtained data when using SVD.

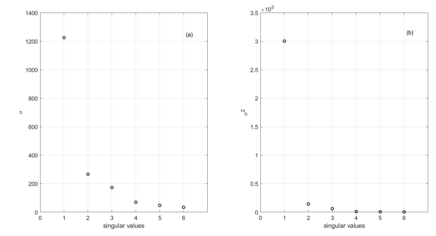


Figure 1: (a) depicts the singular values along the diagonal of $\mathbf{\Sigma}$. (b) is the variance of each of the principal components.

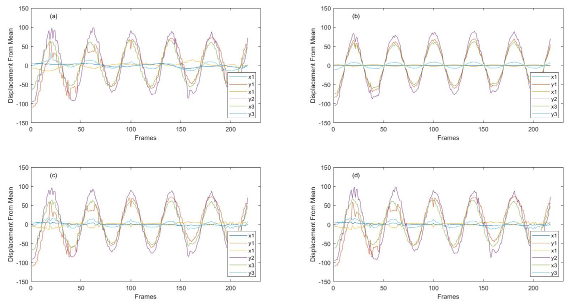
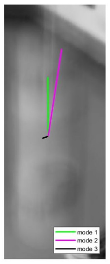


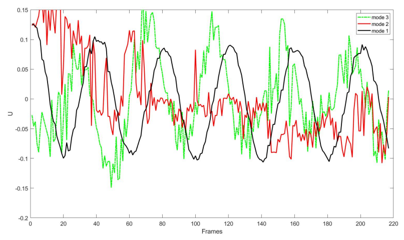
Figure 2: (a) depicts the original measured data $\mathbf{A}$. (b),(c), and (d) depict rank 1,2, and 3 approximations of $\mathbf{A}$, respectively.

4.2 Trial 2

Trial 2 also consisted of a paint can oscillating in the z-direction only, but each camera was shaken vigorously during filming to induce noise. The singular values are shown in Figure 4a with an energy for the rank 1



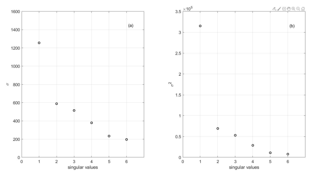
(a) $\mathbf{V}$ applied to the center of the paint can's motion from camera 1's perspective.



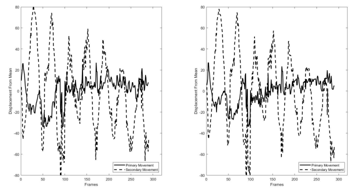
(b) The first three modes of $\mathbf{U}$ as they occur in time measured in video frames

Figure 3: The first three modes of $\mathbf{U}$ and $\mathbf{V}$ for Trial 1

approximation being 40% and the energy of a rank 3 approximation being 92%. The significance lost on σ_1 appears to have been gained mostly by the σ_4, σ_5, and σ_6. Interestingly, because the cameras were shaken in all directions, $\mathbf{V}$ maintained a fairly accurate orientation for the dominant modes, even if $\mathbf{U}$ appears inaccurate as early as mode 2. Because the relationship between Trials 1 and 2 is known, and because I am familiar with the dynamics being recorded, I averaged the measurements for x_1, x_2, y_3 and y_1 ,y_2, x_3, respectively, along with their corresponding rank 3 SVD approximations to produce the plot shown in Figure 4b. This simple filtering technique makes the oscillations more apparent, despite the low energy of each singular value.

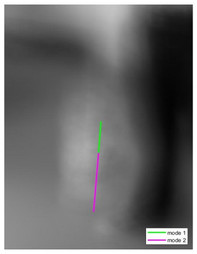


(a) The left panel depicts the singular values along the diagonal of $\mathbf{\Sigma}$. The second panel is the variance of each principal component.

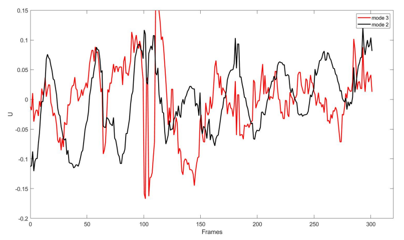


(b) The left panel depicts the average of the original measured data $\mathbf{A}$. The right panel depicts the rank 3 approximation of $\mathbf{A}$ after averaging the same measurements.

Figure 4: The singular values and reproduction of Trial 2



(a) $\mathbf{V}$ applied to the center of the paint can's motion from camera 1's perspective.

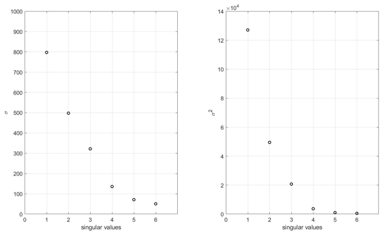


(b) The first 2 modes of $\mathbf{U}$ as they occur in time measured in video frames

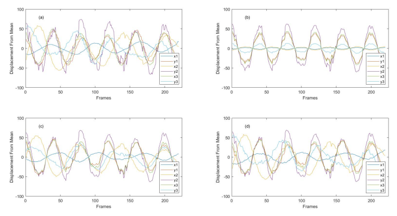
Figure 5: The first 2 modes of $\mathbf{U}$ and $\mathbf{V}$ for Trial 2

4.3 Trial 3

Trial 3 introduced horizontal displacement to the dynamics of Trial 1. The singular values and variances are shown in Figure 6a with an energy for the rank 1 approximation being 43% and the energy of a rank 3 approximation being 86%. The dominant mode as shown in Figure 6 still describes oscillations in the z-direction. Modes 2 and 3 are almost as important, however, and account for horizontal oscillations. They are very closely related, however, and may preserve some redundancy. This is shown fairly clearly in Figure 7b, where the oscillations in modes 2 and 3 both decay in amplitude over time, much like the paint can's pendulum swing decays rapidly. They are however, just slightly out of phase, which is most likely due to the fact that x_2 and y_3 are also slightly out of phase as shown in the top left panel of Figure 6b.

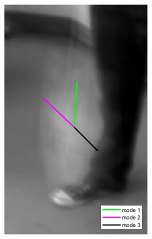


(a) The left panel depicts the singular values along the diagonal of $\mathbf{\Sigma}$. The second panel is the variance of each of the principal components.

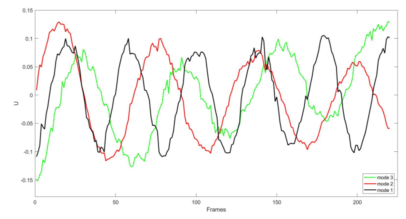


(b) (a) depicts the original measured data $\mathbf{A}$. (b),(c), and (d) depict rank 1,2, and 3 approximations of $\mathbf{A}$, respectively.

Figure 6: The singular values and reproduction of Trial 3



(a) $\mathbf{V}$ applied to the center of the paint can's motion from camera 1's perspective.



(b) The first 3 modes of $\mathbf{U}$ as they occur in time measured in video frames

Figure 7: The first 3 modes of $\mathbf{U}$ and $\mathbf{V}$ for Trial 3. Note mode 1 describing oscillations in the z-direction, and modes 2-3 describing the paint can's horizontal displacement.

4.4 Trial 4

The final trial included oscillations in the z-direction, horizontal displacement, and rotation. The rotation did not effect tracking the motion of the can during any of its oscillations due to the method by which I tracked its position. Because of the orientations of the cameras, however, I could not detect the rotation of the can. The only camera potentially capable of measuring the rotation was camera 1 because of its almost top-down view, but it was not significant enough to change my results. Thus, the singular values and modes closely resemble those of Trial 3, with a rank 1 energy of 47% and a rank 3 energy of 81%. Additionally, modes 2 and 3 seem to describe horizontal oscillations and are out of phase with each other.

5 Summary and Conclusions

This report shows how using singular value decomposition can reveal the principal components, or dynamics of interest, of an system regardless of what that system is or the quantity of measurements taken. The paint

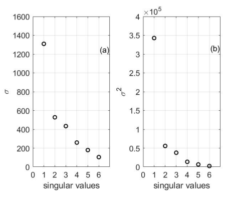


Figure 8: The left panel depicts the singular values along the diagonal of $\mathbf{\Sigma}$. The second panel is the variance of each of the principal components.

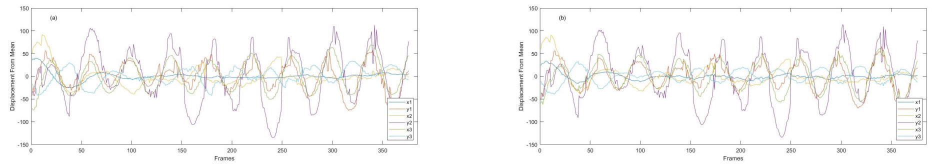


Figure 9: (a) depicts the original measured data $\mathbf{A}$. (b) depicts the rank 3 approximation of $\mathbf{A}$

can is a familiar system, which makes understanding how SVD reveals these components easier because it is then possible to validate the results against what is already known about the system's dynamics. Thus, in each trial I was able to successfully evaluate the singular values and modes to recreate and describe the paint can's principle dynamics. I retained some noise in all trials, which is why modes 4-6 exhibited some energy, but it was not substantial enough to mask the principle dynamics. I would love to recreate the data used in this report with stabilized, synchronized cameras and potentially try other methods of tracking the paint can's motion in pixel space. This should further isolate the principle components found in the SVD analysis.

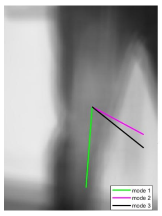
References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

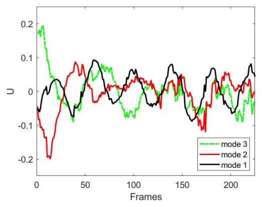
Appendix A MATLAB Functions

What follows is a list of MATLAB functions not covered in the Homework 1 or 2 Reports as they appear in Appendix B. Much of the text below with additional examples can be found in the Mathworks Documentation Library.

- `imshow(x)` displays either a grayscale or full color image in a figure.



(a) $\mathbf{V}$ applied to the center of the paint can's motion from camera 2's perspective.



(b) The first 3 modes of $\mathbf{U}$ as they occur in time measured in video frames

Figure 10: The first 3 modes of $\mathbf{U}$ and $\mathbf{V}$ for Trial 4. Note mode 1 describing oscillations in the z-direction, and modes 2-3 describing the paint can's horizontal displacement.

- `[x,y]=ginput(n)`allows you to identify the coordinates of n points. To choose a point, move your cursor to the desired location and press either a mouse button or a key on the keyboard.
- `implay(x)` opens the Video Viewer app, displaying the content of the file specified by `x`.
- `rgb2gray(x)` converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.
- `[u,s,v]=svd(A,'econ')` produces an economy-size decomposition of m-by-n matrix `A`. The economy-size decomposition removes extra rows or columns of zeros from the diagonal matrix, `S`, along with the columns in either U or V that multiply those zeros.

Appendix B MATLAB Code

This appendix includes the MATLAB code used to produce the results in this report. The code can also be found at `https://github.com/gchase15/AMATH-582`

```
%% Load Data
clear all; close all; clc
addpath('HW3')
load cam1_1
load cam2_1
load cam3_1
load cam1_2
load cam2_2
load cam3_2
load cam1_3
load cam2_3
load cam3_3
load cam1_4
load cam2_4
load cam3_4
implay(vidFrames1_4)
implay(vidFrames2_4)
implay(vidFrames3_4)
%% Cutting the size of the video
%   Find the region that the paint can occupies during the entire video
%   and cut each frame to include only that section.
%   X positions move from left to right.
%   Y positions move from top to bottom.

%Synchronize videos first by cutting initial frames on 2/3 videos
shortFrames1_4=vidFrames1_4(:,:,:,17:end);
shortFrames2_4=vidFrames2_4(:,:,:,20:end);
shortFrames3_4=vidFrames3_4(:,:,:,14:end);
%Find the minimum number of frames to make all videos the same length
F1=size(shortFrames1_4,4);
F2=size(shortFrames2_4,4);
F3=size(shortFrames3_4,4);
FrameLength=[F1 F2 F3];
FrameLength=min(FrameLength);
%Pick the "box" that you want to cut out of each frame so you only have the
%movement of the paint can.
imshow(shortFrames1_4(:,:,:,1));
[x1,y1]=ginput(2);
imshow(shortFrames2_4(:,:,:,1));
[x2,y2]=ginput(2);
imshow(shortFrames3_4(:,:,:,1));
[x3,y3]=ginput(2);  close
%Shrink each frame to fit the box picked above and cut any frames after
%FrameLength ends.
cutFrames1_4=shortFrames1_4(min(y1):max(y1),min(x1):max(x1),:,1:FrameLength);
cutFrames2_4=shortFrames2_4(min(y2):max(y2),min(x2):max(x2),:,1:FrameLength);
cutFrames3_4=shortFrames3_4(min(y3):max(y3),min(x3):max(x3),:,1:FrameLength);
%Confirm you got it right and enjoy the show
implay(cutFrames1_4)
implay(cutFrames2_4)
implay(cutFrames3_4)
```

Listing 1: MATLAB code covering loading the videos, synchronizing them, and reducing them for tracking the paint can

```
 %% Track Video 1
 FrameLength=376;
 %First Part of For loop converts each Frame into grayscale. It preserves a
 %copy of this image as FrameGray1 for comparison later.
 xpos14=[];
 ypos14=[];
 for j=1:FrameLength
 Frame=cutFrames1_4(:,:,:,j);
 FrameGray=rgb2gray(Frame);
 FrameGray1=FrameGray;

%The for and if loops below filter out colors other than bright white.
%The Detection threshold is set to 253 to guard against the rare case where
%the light's color isn't shown at 255.
 for j=1:size(FrameGray,1);
     for i=1:size(FrameGray,2);
         if FrameGray(j,i) > 244;
             FrameGray(j,i)=255;
         else FrameGray(j,i)=0;
         end
     end
 end

 %These figures serve to validate that what remains in the filtered frame
 %are the viable candidates for tracking. They should be commented out once
 %validation is complete.
 figure(1)
 imshow(FrameGray)
 figure(2)
 imshow(FrameGray1)

 %Pick out the location of the first white pixel. Congratulations, you've
%extracted out the x and y positions for a paint can.
 [ylight, xlight]=ind2sub(size(FrameGray),find(FrameGray,1));

 %Alternatively, uncomment the next two lines to pick out the location of
 %the last white pixel. This is occassionally more consistent.
%  white=find(FrameGray);
%  [ylight, xlight]=ind2sub(size(FrameGray),white(end));

 xpos14=[xpos14;
     xlight];
 ypos14=[ypos14;
     ylight];
 end

  %Subtract the mean from each row
 xpos14=xpos14-mean(xpos14);
 ypos14=ypos14-mean(ypos14);
```

Listing 2: MATLAB code covering tracking the paint can.

```
%% The SVD

 %create a matrix of the three video's x and y pos over time
 A4=[xpos14';
     ypos14';
     xpos24';
     ypos24';
     xpos34';
     ypos34'];

  [u4,s4,v4]=svd(A4','econ'); %U and V are rotations, s is the stretch
  A4Cov=cov(A4');

  variance4=(s4.^2)./(size(s4,1)-1);
  var4=diag(variance4);

  sig4=diag(s4); %Collects the sigma values from s into a vector
  energy1=sig4(1)/sum(sig4)%Proportion of first singular value amongst all of them
  energy2=sig4(2)/sum(sig4)
  eng4=sum(sig4(1:3))/sum(sig4)
  %% Describe Motion in mean frame

        meanFrames=double(rgb2gray(cutFrames2_4(:,:,:,1)));
    for j=2:size(cutFrames2_4,4)
        meanFrames=meanFrames + double(rgb2gray(cutFrames2_4(:,:,:,j)));
    end
     meanFrames=uint8(meanFrames/size(cutFrames2_4,4));

    figure(25)
    imshow(meanFrames)
    hold on
    [xcent,ycent]=ginput(1);
    plot([xcent,xcent+100*v4(3,1)],[ycent,ycent+100*v4(4,1)],'g','linewidth',2)
    plot([xcent,xcent+100*v4(3,3)],[ycent,ycent+100*v4(4,2)],'m','linewidth',2)
    plot([xcent,xcent+100*v4(3,3)],[ycent,ycent+100*v4(4,3)],'k','linewidth',2)
    legend('mode 1','mode 2','mode 3','Location','SouthEast')
```

Listing 3: MATLAB code covering computing the SVD and variance. It also covers projecting the V modes onto an averaged image of the paint can's motion.

```
%% Plotting SVD
 x4=1:FrameLength;

 figure(8)
subplot(1,2,1)
plot(sig4,'ko','Linewidth',[1.5])
axis([0 7 0 1600])
set(gca,'Fontsize',[13],'Xtick',[0 1 2 3 4 5 6])
xlabel('singular values'), ylabel('\sigma')
grid on
text(6.1,1.25*10^3,'(a)','Fontsize',[13])
subplot(1,2,2)
plot(var4,'ko','Linewidth',[1.5])
axis([0 7 0 4*10^5])
set(gca,'Fontsize',[13],'Xtick',[0 1 2 3 4 5 6])
xlabel('singular values'), ylabel('\sigma^2')
grid on
text(6.1,3.16*10^5,'(b)','Fontsize',[13])

figure(12)
 plot(x4,u4(:,3),'g-.',x4,u4(:,2),'r',x4,u4(:,1),'k','Linewidth',[2])
 set(gca,'Fontsize',[13])
 legend('mode 3','mode 2','mode 1','Location','SouthEast')
 axis([0 225 -.25 .25])
 xlabel('Frames'), ylabel('U')
```

Listing 4: MATLAB code covering plotting the singular values, variances, and U modes.

```
%% Reproduction

 t=1:length(x4);
 figure(21)
 subplot(1,2,1)
 plot(t,xpos14,t,ypos14,t,xpos24,t,ypos24,t,xpos34,t,ypos34);
 axis([0 385 -150 150])
 legend('x1','y1','x2','y2','x3','y3','Location','SouthEast')

 for j=1:3
 ff4=u4(:,1:j)*s4(1:j,1:j)*v4(:,1:j)'; % modal projections
   subplot(1,2,2)
   plot(x4,ff4)
   legend('x1','y1','x2','y2','x3','y3','Location','SouthEast')
   axis([0 385 -150 150])
 end

 subplot(1,2,1), text(19,130,'(a)','Fontsize',[10]), xlabel('Frames'),
 ylabel('Displacement From Mean')
 subplot(1,2,2), text(19,130,'(b)','Fontsize',[10]), xlabel('Frames'),
 ylabel('Displacement From Mean')
 subplot(2,2,3), text(19,130,'(c)','Fontsize',[10]), xlabel('Frames'),
 ylabel('Displacement From Mean')
 subplot(2,2,4), text(19,130,'(d)','Fontsize',[10]), xlabel('Frames'),
 ylabel('Displacement From Mean')

 motion14=(xpos14(12:end)+xpos24(12:end)+ypos34(12:end))/3;
 motion24=(ypos14(12:end)+ypos24(12:end)+xpos34(12:end))/3;
 PCAmot14=(ff4(12:end,1)+ff4(12:end,3)+ff4(12:end,6))/3;
 PCAmot24=(ff4(12:end,2)+ff4(12:end,4)+ff4(12:end,5))/3;
 tm4=1:length(motion14);

 figure(4)
 subplot(1,2,1)
 plot(tm4,motion14, 'k', tm4,motion24,'k-.', 'Linewidth',[2])
 axis([0 380 -80 80])
 xlabel('Frames'), ylabel('Displacement From Mean')
 legend('Primary Movement', 'Secondary Movement', 'Location','SouthEast')
 subplot(1,2,2)
 plot(tm4,PCAmot14, 'k', tm4,PCAmot24,'k-.', 'Linewidth',[2])
 axis([0 380 -80 80])
 xlabel('Frames'), ylabel('Displacement From Mean')
 legend('Primary Movement', 'Secondary Movement', 'Location','SouthEast')
```

Listing 5: MATLAB code covering reproducing the original measurements using only the principal components from the SVD.