HW 4 : Principal Component Analysis

Anamol Pundle

February 26, 2015

Abstract

The process and results of applying Principal Component Analysis (PCA) to videos of an oscillating mass taken by three cameras placed at different locations simultaneously is documented in this report. The oscillating mass is tracked using the flashlight placed on top of it and the position of the mass obtained from each camera is saved. Some of the data is found to be phase shifted, and is brought back into phase. This data is integrated into one matrix and a singular value decomposition(SVD) is performed on it. The data is analyzed in various ways, such as and the effect of PCA on the data is ascertained. The results of the SVD of each set of videos are also compared to each other.

1 Introduction and Overview

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding components. The principal components are orthogonal because they are the eigenvectors of the covariance matrix, which is symmetric.

Four tests are performed on an oscillating mass (can). The can is moved, and is filmed by three cameras simultaneously. In the first test (ideal case), the can executes simple harmonic motion in the z direction. In the second test, the can again undergoes SHM, but noise in introduced into the images due to a slight shake of the camera. In the third test, the mass is released off-center so as to produce motion in the x−y plane as well as the z direction. Thus there is both a pendulum motion and a simple harmonic oscillations. In the fourth test, the mass is released off-center and rotates so as to produce motion in the x−y plane, rotation as well as the z direction. Thus there is both a pendulum motion and a simple harmonic oscillations.

2 Theoretical Background

Singular Value Decomposition is a useful tool using which any matrix can be written as the product of a unitary matrix U, a rectangular diagonal matrix with non-negative diagonal elements S and the conjugate transpose of another unitary matrix V, V^*. The matrices U and V^* are rotational matrices, while S is a stretching matrix. Every matrix has a singular value decomposition, and the singular values are uniquely determined. Several methods can be used to calculate the SVD. The SVD can be used to diagonalize a matrix, if the proper bases for the domain and range are used. Several useful theorems and ways to calculate the SVD are given in reference [1]. The SVD gives a type of least-square fitting algorithm, allowing us to project the matrix onto low-dimensional representations in a formal, algorithmic way.

PCA is the simplest of the true eigenvector-based multivariate analyses. Often, its operation can be thought of as revealing the internal structure of the data in a way that best explains the variance in the data. If a multivariate dataset is visualized as a set of coordinates in a high-dimensional data space (1 axis per variable), PCA can supply the user with a lower-dimensional picture, a projection or "shadow" of this object when viewed from its most informative viewpoint. This is done by using only the first few principal components so that the dimensionality of the transformed data is reduced. Redundant data can also be identified using the covariant matrix. The diagonal terms of the covariant matrix are the measure variance for particular measurements. By assumption, large variances correspond to dynamics of interest, whereas low variances are assumed to correspond to non-interesting dynamics. The off-diagonal terms of the covariant matrix are the covariance between measurements. Indeed, the off-diagonals captures the correlations between all possible pairs of measurements. A large off-diagonal term represents two events that have a high-degree of redundancy, whereas a small off-diagonal coefficient means there is little redundancy in the data, i.e. they are statistically independent.

3 Algorithm Implementation and Development

Similar algorithms are used to extract the position of the can from the videos for each test case. PCA is then performed on the extracted data, and the results are post-processed.

- Load videos from the given .mat files. Each video file is saved in a 4D matrix. Find the size of each frame and the number of frames of each video.
- Convert the videos from RGB to grayscale.
- Isolate the portions of the video where the movement of the can occurs. This is done by setting all pixels except a small window to zero(black).

- Since the oscillating can has a flashlight on top, search the window for the maximum intensity pixel, using the `max` function.
- Convert the index obtained from the previous step to (x, y) using the `ind2sub` function. Save each x and y separately to a X and Y vector.
- Align the position of the can in each video to the same array index. The is done because the data may be phase shifted, and is done by searching for the index i of the minimum position of the can in the first 50 steps, and truncating all data before i and after $i + 200$.
- Clean each X and Y vector using a Shannon window in the frequency domain.
- Create a matrix A where each row is the X and Y vectors.
- Perform a singular value decomposition on A. The u, s and v matrices are obtained in this manner.
- Analyze resulting data, such as plotting mode projections, the relative energy of each mode, predictions of X and Y using an increasing number of modes, and looking at the covariant m

4 Computational Results

4.1 Test 1

Figure 1 shows the raw data extracted from the three different cameras, along with the extracted data smoothed with the Shannon window. The black line shows raw data, while the blue line shows data smoothed using the Shannon window. The can is seen to oscillate along the Y direction in the footage from cameras 1 and 2. In the third camera, the direction of oscillation is in the X direction. The can is mostly steady in the X direction (Y direction for camera 3).

Figure 2 shows the energy of each mode, which is obtained from the diagonal of the s matrix. It is observed that about 97% of the energy is contained in the first two modes, with 87% being contained in the first mode.

Figure 3 shows the position of the can extracted after performing SVD on the A matrix, recreated using the first two modes. Good agreement is observed, compared to Figure 1. Another observation concerns spikes which are observed in one camera's data and not in the others. The first two modes of the SVD show these spikes in the other camera's data too. Hence, the SVD tool may be used as a tool to make data from sensors that are measuring the same quantity more uniform. When the position of the can obtained from the SVD is plotted on top of the original video, the pointer follows the can quite closely.

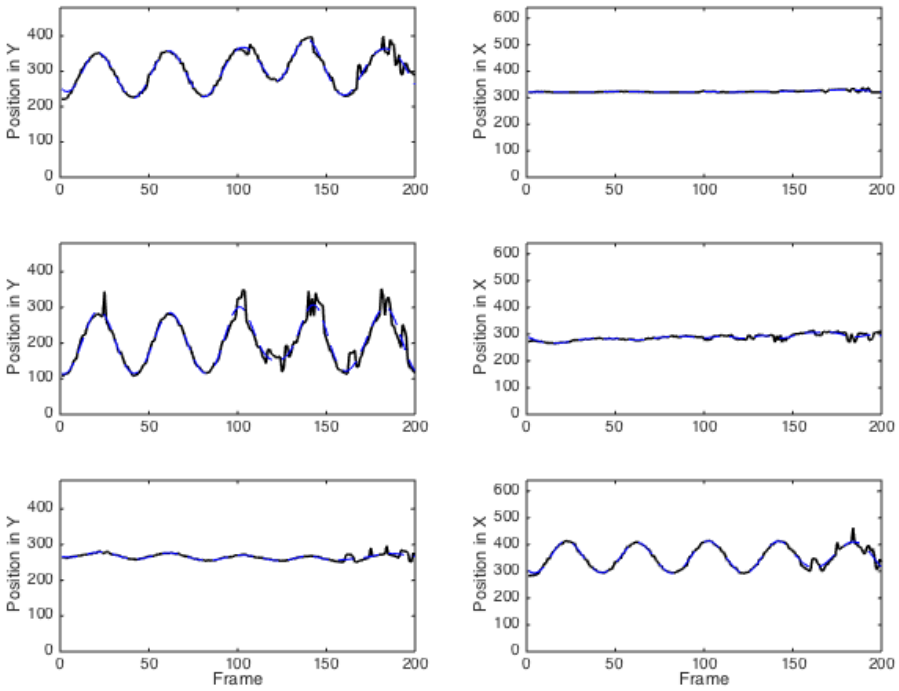

Figure 1: X and Y position of can per frame for Test 1. Black is raw data and blue is smoothed data; (Top Row) camera 1 (Mid Row); camera 2; (Bottom Row) camera 3

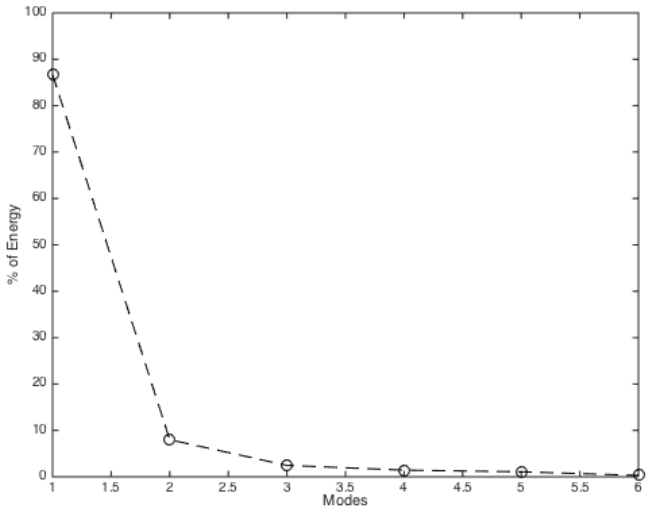

Figure 2: Energy of each mode obtained after SVD for Test 1

4.2 Test 2

Figure 4 shows the raw data extracted from the three different cameras for test 2, along with the extracted data smoothed with the Shannon window. The black line shows raw data, while the blue line shows data smoothed using the Shannon window. All cameras are shaking while capturing the images, due to which the data is noisy. The can is seen to oscillate along the Y direction in the footage from cameras 1 and 2. In the third camera, the direction of oscillation is in the X direction. Spikes seen in the data are the places where the point of maximum intensity shifts from the flashlight to a different area. Hence, this method is not completely foolproof.

Figure 2 shows the energy of each mode, which is obtained from the diagonal of the s matrix. It is observed that about 95% of the energy is contained in the first three modes, with 82% being contained in the first mode. This is smaller compared to the first test case, possibly due to excessive noise in the system.

Figure 3 shows the position of the can extracted after performing SVD on the A matrix, recreated using the first three modes. Good agreement is observed, compared to Figure 4. When the position of the can obtained from the SVD is plotted on top of the original video, the pointer does not quite follow the can closely; hence this method of tracking the can does not work well when there is noise in the system.

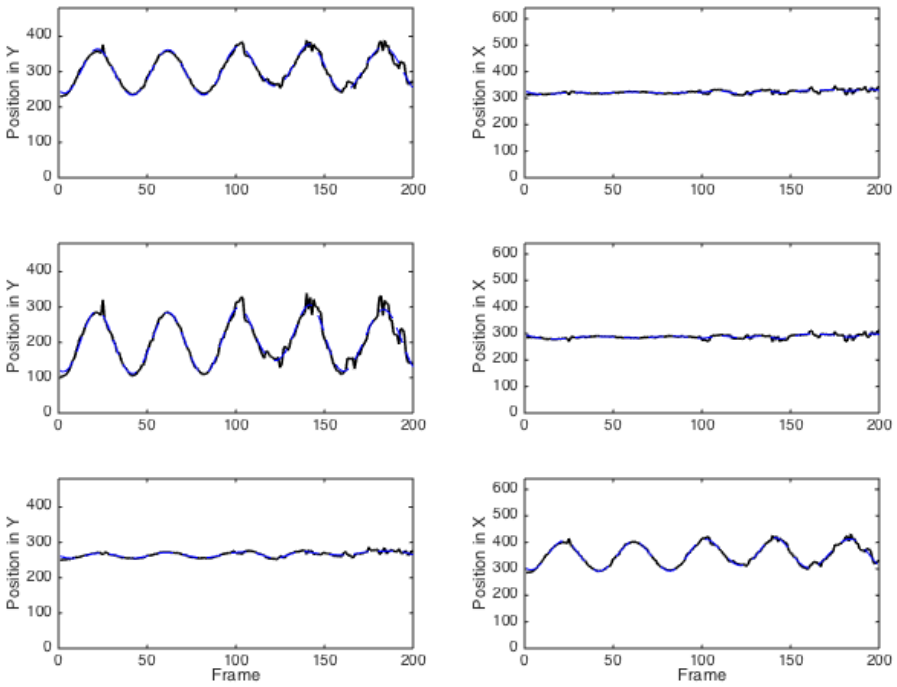

Figure 3: X and Y positions recreated from first 2 modes of SVD for Test 1; (Top Row) camera 1 (Mid Row); camera 2; (Bottom Row) camera 3

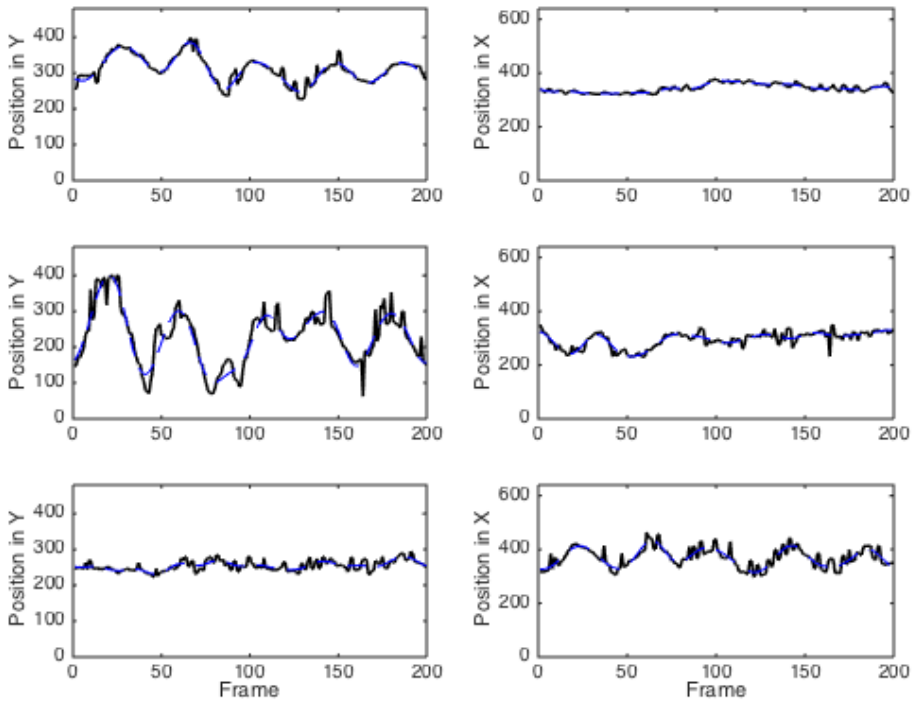

Figure 4: X and Y position of can per frame for Test 2. Black is raw data and blue is smoothed data; (Top Row) camera 1 (Mid Row); camera 2; (Bottom Row) camera 3

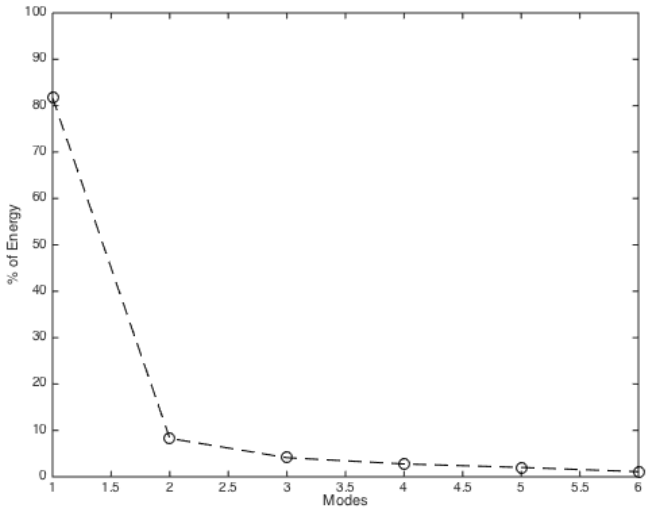

Figure 5: Energy of each mode obtained after SVD for Test 2

4.3 Test 3

Figure 7 shows the raw data extracted from the three different cameras for test 2, along with the extracted data smoothed with the Shannon window. The black line shows raw data, while the blue line shows data smoothed using the Shannon window. In this case, the can is not executing simple harmonic motion in the z direction, but is also moving in the x and $y$directions. The can is mostly steady in the X direction (Y direction for camera 3).

Figure 8 shows the energy of each mode, which is obtained from the diagonal of the s matrix. It is observed that about 97% of the energy is contained in the first three modes, with 85% being contained in the first mode. This is smaller compared to the first test case, possibly due to the can not executing simple harmonic motion, but moving in the $x - y$ plane as well.

Figure 9 shows the position of the can extracted after performing SVD on the A matrix, recreated using the first three modes. Good agreement is observed, compared to Figure 7. When the position of the can obtained from the SVD is plotted on top of the original video, the pointer follows the can closely; hence this method of tracking works well when there the motion is not purely simple harmonic.

4.4 Test 4

Figure 10 shows the raw data extracted from the three different cameras for test 2, along with the extracted data smoothed with the Shannon window. The black line shows raw data, while the blue line shows data smoothed using the

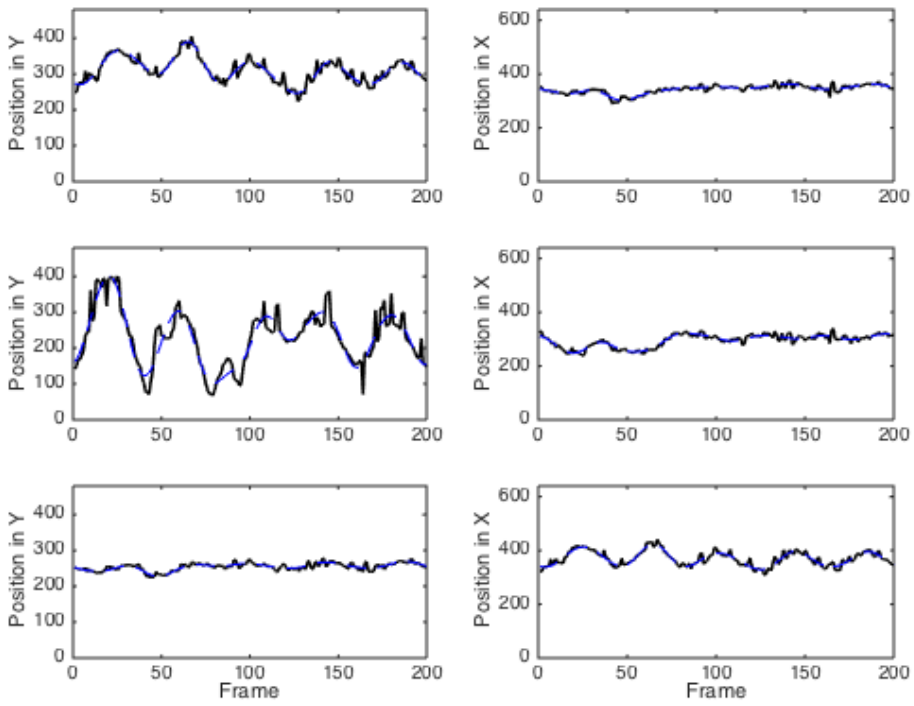

Figure 6: X and Y positions recreated from first 3 modes of SVD for Test 2; (Top Row) camera 1 (Mid Row); camera 2; (Bottom Row) camera 3

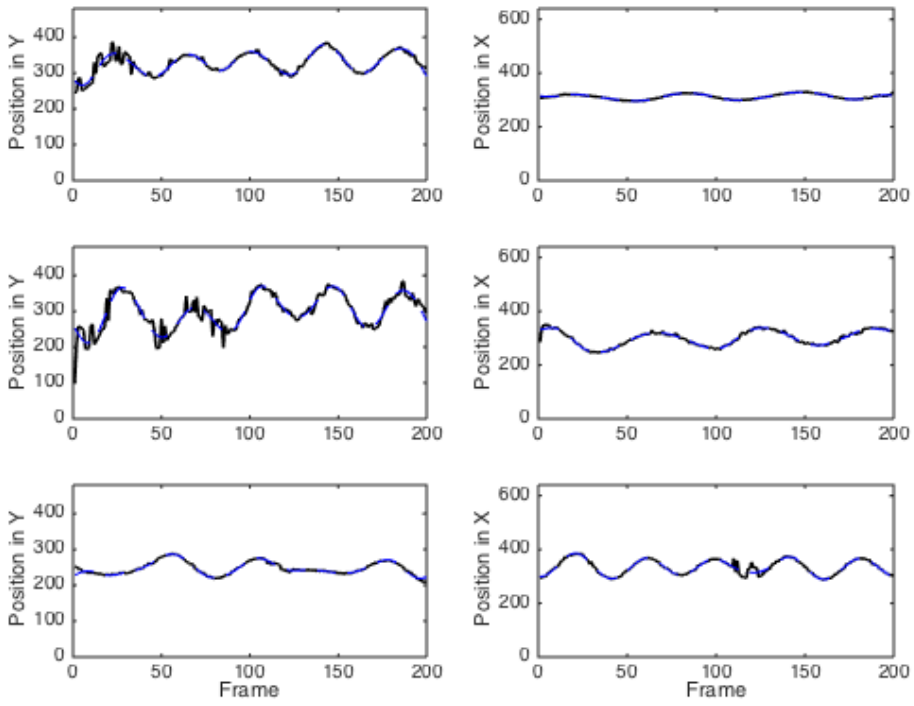

Figure 7: X and Y position of can per frame for Test 3. Black is raw data and blue is smoothed data; (Top Row) camera 1 (Mid Row); camera 2; (Bottom Row) camera 3

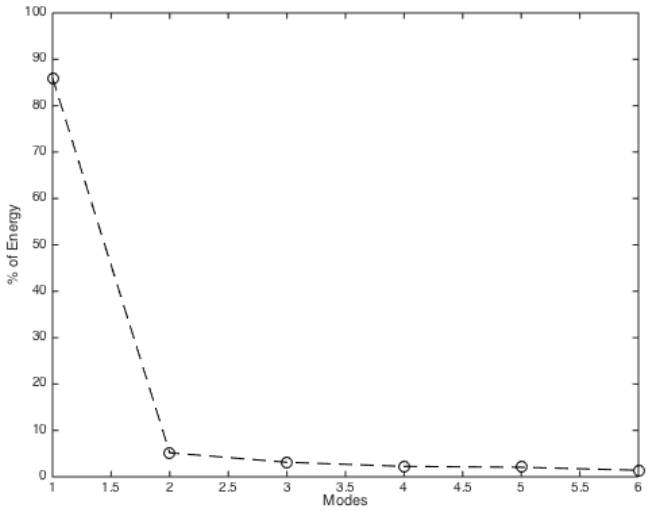

Figure 8: Energy of each mode obtained after SVD for Test 3

Shannon window. In this case, the can is not executing simple harmonic motion in the z direction, but is also moving in the x and $y$directions, as well as rotating along its axis.

Figure 11 shows the energy of each mode, which is obtained from the diagonal of the s matrix. It is observed that about 93% of the energy is contained in the first three modes, with 85% being contained in the first mode. This is smaller compared to the first test case, possibly due to the can not executing simple harmonic motion, but moving in the $x - y$ plane as well as rotating.

Figure 12 shows the position of the can extracted after performing SVD on the A matrix, recreated using the first three modes. Good agreement is observed, compared to Figure 10. Another observation concerns spikes which are observed in one camera's data and not in the others. The first two modes of the SVD show these spikes in the other camera's data too. Hence, the SVD tool may be used as a tool to make data from sensors that are measuring the same quantity more uniform. When the position of the can obtained from the SVD is plotted on top of the original video, the pointer follows the can quite closely.

5 Summary and Conclusion

The technique of Principal Component Analysis is applied to videos of an oscillating mass, with different tests done for the ideal case of pure SHM, noise due to shaking of the camera, motion in the horizontal plane along with oscillations in the vertical plane, and motion in all planes along with rotation of

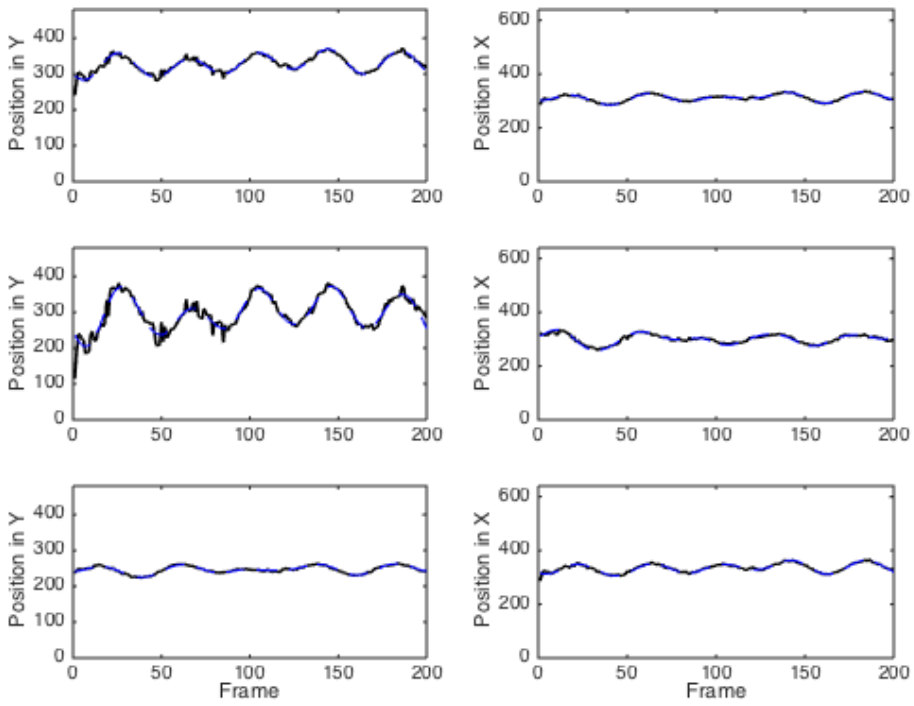

Figure 9: X and Y positions recreated from first 3 modes of SVD for Test 3; (Top Row) camera 1 (Mid Row); camera 2; (Bottom Row) camera 3

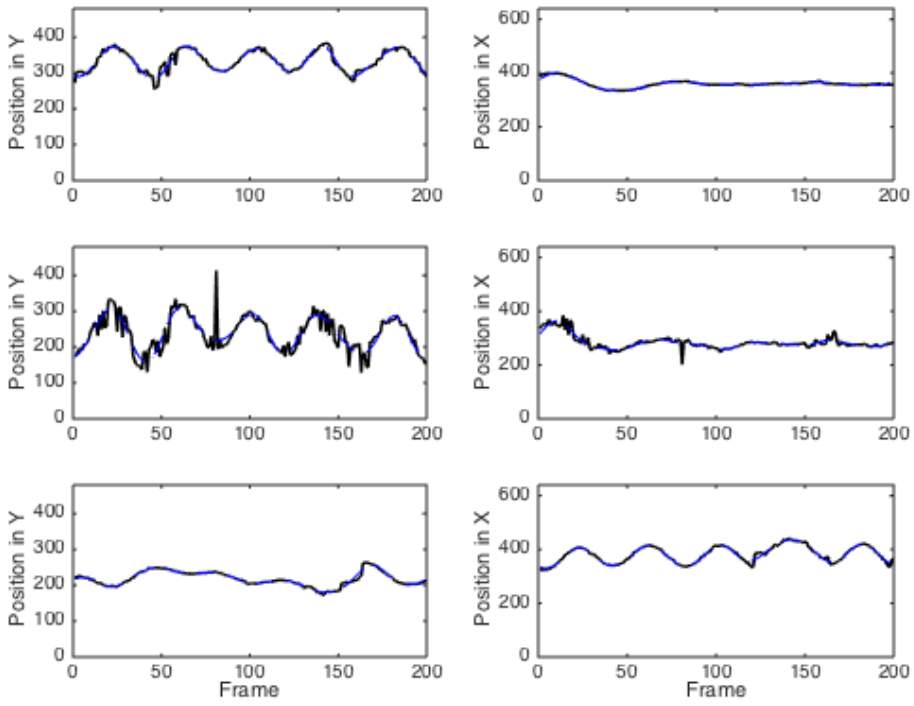

Figure 10: X and Y position of can per frame for Test 4. Black is raw data and blue is smoothed data; (Top Row) camera 1 (Mid Row); camera 2; (Bottom Row) camera 3

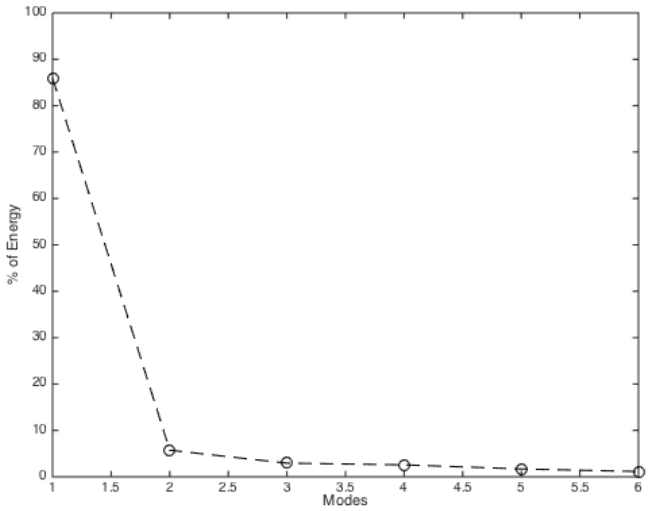

Figure 11: Energy of each mode obtained after SVD for Test 4

the mass along its vertical axis. The mass was tracked by creating a window around the area where the mass moves, and finding the maximum intensity in that window for each frame. The singular value decomposition of data yielded six modes. It was found that the first mode contains greater than 82% of the energy for all tests, and three modes are sufficient to recreate the motion to 92% accuracy in the worst case. The PCA did best in the ideal case, where two modes were sufficient, giving 97% accuracy. The worst case was test 2, when noise was introduced through the random motion of the camera (aka shaking). In this case, the first mode contained 82% of the total energy. An interesting point noted was that spikes which were picked up by one camera and not by the others appeared in other cameras in the two- or three-mode recreation of the motion using SVD.

References

1. Kutz, J. Nathan, Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data, September 2013

2. Matlab help

Appendix I: MATLAB functions used

`load`: Load data from MAT-file into workspace.

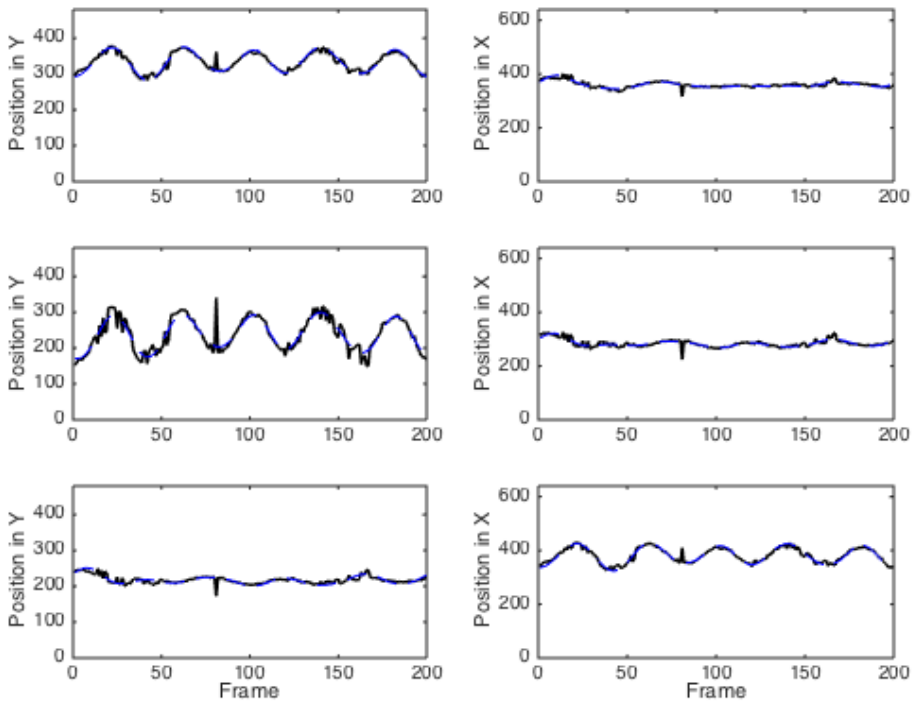

Figure 12: X and Y positions recreated from first 3 modes of SVD for Test 4; (Top Row) camera 1 (Mid Row); camera 2; (Bottom Row) camera 3

- `S = load(FILENAME)` loads the variables from a MAT-file into a structure array, or data from an ASCII file into a double-precision array.

`frame2im`: Return image data associated with movie frame.

- `[X,MAP] = frame2im(F)` returns the indexed image X and associated colormap MAP from the single movie frame F.

`rgb2gray`: Convert RGB image or colormap to grayscale. `rgb2gray` converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

- `I = rgb2gray(RGB)` converts the truecolor image RGB to the grayscale intensity image I.

`ind2sub`: Multiple subscripts from linear index. `ind2sub` is used to determine the equivalent subscript values corresponding to a given single index into an array.

- `[I,J] = ind2sub(SIZ,IND)` returns the arrays I and J containing the equivalent row and column subscripts corresponding to the index matrix IND for a matrix of size SIZ.

`svd`: Singular value decomposition.

- `[U,S,V] = svd(X)` produces a diagonal matrix S, of the same dimension as X and with nonnegative diagonal elements in decreasing order, and unitary matrices U and V so that `X = U*S*V'`.

`diag`: Diagonal matrices and diagonals of a matrix.

- `diag(V,K)` when V is a vector with N components is a square matrix of order N+ABS(K) with the elements of V on the K-th diagonal. $K = 0$ is the main diagonal, $K > 0$ is above the main diagonal and $K < 0$ is below the main diagonal.

`imshow`: displays image

- `imshow(I)` displays the image I in a Handle Graphics figure, where I is a grayscale, RGB (truecolor), or binary image. For binary images, `imshow` displays pixels with the value 0 (zero) as black and 1 as white.

```
% APPENDIX II: MATLAB CODE

% This shows the first test only. The code for all other tests is
%  extremely similar, and availible on request.


close all; clear all; clc;

%load videos
load('cam1_1.mat');
load('cam2_1.mat');
load('cam3_1.mat');

% find size of frame
[m1,n1] = size(vidFrames1_1(:,:,1,1));
[m2,n2] = size(vidFrames2_1(:,:,1,1));
[m3,n3] = size(vidFrames3_1(:,:,1,1));

% find number of frames
numFrames1=size(vidFrames1_1,4);
numFrames2=size(vidFrames2_1,4);
numFrames3=size(vidFrames3_1,4);
maxFrames = max([numFrames1 numFrames2 numFrames3]);

for k = 1:maxFrames
    if k <= numFrames1
        mov1(k).cdata = vidFrames1_1(:,:,:,k);
        mov1(k).colormap = [];

    end
    if k <= numFrames2
        mov2(k).cdata = vidFrames2_1(:,:,:,k);
        mov2(k).colormap = [];
    end
    if k <= numFrames3
         mov3(k).cdata = vidFrames3_1(:,:,:,k);
         mov3(k).colormap = [];
    end
end

X1 = []; X2 = []; X3 = [];
Y1 = []; Y2 = []; Y3 = [];

% convert videos to grayscale
% add window for tracking movement
% find point of maximum intensity
% save x and y coordinates of max intensity in mat X and Y
for jj = 1:maxFrames

    if jj <= numFrames1
        X=frame2im(mov1(jj));
        a = rgb2gray(X);
```

```
        Vid1org(:,:,jj) = a;
        a(:,1:320) = 0; a(:,380:end) = 0;
        a(1:200,:) = 0;
        Vid1(:,:,jj) =  a;
        [Max Ind] = max(a(:));
        [x1 y1] = ind2sub(size(a), Ind);
        X1 = [X1 x1]; Y1 = [Y1 y1];
    end

    if jj <= numFrames2
        X=frame2im(mov2(jj));
        a = rgb2gray(X);
        Vid2org(:,:,jj) = a;
        a(:,1:260) = 0; a(:,330:end) = 0;
        Vid2(:,:,jj) =  a;
        [Max Ind] = max(a(:));
        [x2 y2] = ind2sub(size(a), Ind);
        X2 = [X2 x2]; Y2 = [Y2 y2];
    end

    if jj <= numFrames3
        X=frame2im(mov3(jj));
        a = rgb2gray(X);
        Vid3org(:,:,jj) = a;
        a(1:250,:) = 0; a(310:end,:) = 0;
        a(:, 1:260) = 0;
        Vid3(:,:,jj) =  a;
        [Max Ind] = max(a(:));
        [x3 y3] = ind2sub(size(a), Ind);
        X3 = [X3 x3]; Y3 = [Y3 y3];
    end

end

% bring phase shifted data back in phase
[Min1 I1]=min(X1(1:50)); X1=X1(I1:I1+200); Y1=Y1(I1:I1+200);
[Min2 I2]=min(X2(1:50)); X2=X2(I2:I2+200); Y2=Y2(I2:I2+200);
[Min3 I3]=min(Y3(1:50)); Y3=Y3(I3:I3+200); X3=X3(I3:I3+200);

% smooth data using Shannon window in frequency domain
X1f = fft(X1); X2f = fft(X2); X3f = fft(X3);
Y1f = fft(Y1); Y2f = fft(Y2); Y3f = fft(Y3);
X1f(10:end-10) = 0; X2f(10:end-10) = 0; X3f(10:end-10) = 0;
Y1f(10:end-10) = 0; Y2f(10:end-10) = 0; Y3f(10:end-10) = 0;
Xo1 = abs(ifft(X1f)); Yo1 = abs(ifft(Y1f));
Xo2 = abs(ifft(X2f)); Yo2 = abs(ifft(Y2f));
Xo3 = abs(ifft(X3f)); Yo3 = abs(ifft(Y3f));

% % plot smoothed and unsmoothed data
% figure (1)
% subplot(3,2,1), plot(X1, 'k', 'LineWidth', 1.5); hold on;
% plot(Xo1, 'b--', 'LineWidth', 0.25)
% axis([0 200 0 480]); ylabel('Position in Y');
% subplot(3,2,2), plot(Y1, 'k', 'LineWidth', 1.5); hold on;
```

```
% plot(Yo1, 'b--', 'LineWidth', 0.25)
% axis([0 200 0 640]); ylabel('Position in X');
% subplot(3,2,3), plot(X2, 'k', 'LineWidth', 1.5); hold on;
% plot(Xo2, 'b--', 'LineWidth', 0.25)
% axis([0 200 0 480]); ylabel('Position in Y');
% subplot(3,2,4), plot(Y2, 'k', 'LineWidth', 1.5); hold on;
% plot(Yo2, 'b--', 'LineWidth', 0.25)
% axis([0 200 0 640]); ylabel('Position in X');
% subplot(3,2,5), plot(X3, 'k', 'LineWidth', 1.5); hold on;
% plot(Xo3, 'b--', 'LineWidth', 0.25)
% axis([0 200 0 480]); ylabel('Position in Y'); xlabel('Frame');
% subplot(3,2,6), plot(Y3, 'k', 'LineWidth', 1.5); hold on;
% plot(Yo3, 'b--', 'LineWidth', 0.25)
% axis([0 200 0 640]); ylabel('Position in X'); xlabel('Frame');

% put all data in matrix to do SVD
Amat = [X1; Y1; X2; Y2; X3; Y3];
Aomat = [Xo1; Yo1; Xo2; Yo2; Xo3; Yo3];

% do SVD
[u s v] = svd(Amat);
[uo so vo] = svd(Aomat);

% two-mode recreation of original matrix A
Anew = u(:,1:2)*s(1:2,1:2)*v(:,1:2)';
Aonew = uo(:,1:2)*so(1:2,1:2)*vo(:,1:2)';

% calculate new X and Y matrices from two-mode recreation
Xnew1 = Anew(1,:); Ynew1 = Anew(2,:);
Xnew2 = Anew(3,:); Ynew2 = Anew(4,:);
Xnew3 = Anew(5,:); Ynew3 = Anew(6,:);
Xonew1 = Aonew(1,:); Yonew1 = Aonew(2,:);
Xonew2 = Aonew(3,:); Yonew2 = Aonew(4,:);
Xonew3 = Aonew(5,:); Yonew3 = Aonew(6,:);

% % plot smoothed and unsmoothed two-mode recreation
% figure (2)
% subplot(3,2,1), plot(Xnew1, 'k', 'LineWidth', 1.5); hold on;
% plot(Xonew1, 'b--', 'LineWidth', 0.25)
% axis([0 200 0 480]); ylabel('Position in Y');
% subplot(3,2,2), plot(Ynew1, 'k', 'LineWidth', 1.5); hold on;
% plot(Yonew1, 'b--', 'LineWidth', 0.25)
% axis([0 200 0 640]); ylabel('Position in X');
% subplot(3,2,3), plot(Xnew2, 'k', 'LineWidth', 1.5); hold on;
% plot(Xonew2, 'b--', 'LineWidth', 0.25)
% axis([0 200 0 480]); ylabel('Position in Y');
% subplot(3,2,4), plot(Ynew2, 'k', 'LineWidth', 1.5); hold on;
% plot(Yonew2, 'b--', 'LineWidth', 0.25)
% axis([0 200 0 640]); ylabel('Position in X');
% subplot(3,2,5), plot(Xnew3, 'k', 'LineWidth', 1.5); hold on;
% plot(Xonew3, 'b--', 'LineWidth', 0.25)
% axis([0 200 0 480]); xlabel('Frame'); ylabel('Position in Y');
% subplot(3,2,6), plot(Ynew3, 'k', 'LineWidth', 1.5); hold on;
% plot(Yonew3, 'b--', 'LineWidth', 0.25)
```

```
% axis([0 200 0 640]); xlabel('Frame'); ylabel('Position in X');
%
% figure(1)
% plot(diag(s)*100/sum(diag(s)), 'ko--', 'MarkerSize', 8'); hold on;
% axis([1 6 0 100]); xlabel('Modes'); ylabel('% of Energy');
```

Published with MATLAB® R2014b