Harmonic Oscillation Motion Tracking

Kellen Betts

July 31, 2016

Abstract

In this project, four experiments are used to develop an algorithm for principal component analysis (PCA) with singular value decomposition (SVD). We explore simple harmonic oscillations, pendulum motion, and the effect of noise. The experimental apparatus is a paint can hanging from a spring. The movement of the paint can is recorded by three cameras. The video files from the cameras are processed to track the location coordinates of the paint can over the course of the experiment. PCA is used to extract the dynamical behavior from the data by reducing redundancy and identifying the directions with maximal variance. In the ideal case where the paint can oscillates in a single direction, the PCA algorithm effectively isolates the behavior with a rank one approximation. If noise is introduced by shaking the cameras, the PCA algorithm is still able to isolate the oscillatory behavior but with less accuracy. Finally, if the paint can oscillates in the vertical direction and swings or rotates in the horizontal direction, the PCA algorithm extracts the multidimensional behavior with expected rank and reasonable accuracy.

1 Introduction

Four experiments are conducted using a paint can hanging from a spring. The movement of the paint can is recorded by three independent cameras (iPhones). The four experiments are:

1. **Test 1** (ideal case): A small displacement of the paint can in the z direction and the ensuing oscillations. In this case, the entire motion is in the z directions with simple harmonic motion being observed.

2. **Test 2** (noisy case): Repeat the ideal case experiment, but this time introduce camera shake into the video recording. This should make it more difficult to extract the simple harmonic motion.

3. **Test 3** (horizontal displacement): In this case, the paint can 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.

4. **Test 4** (horizontal rotation): In this case, the mass is released off-center and rotates so as to produce rotation in the $x-y$ plane as well as oscillation in the z direction. Thus, there is both a pendulum motion and a simple harmonic oscillations.

The primary objective in this assignment is to use principal component analysis (PCA), where singular value decomposition (SVD) is a key tool, to extract the dynamics of the paint can from the camera recordings by reducing redundancy and identifying the directions of maximal variance. The location of the paint can is tracked in each video by isolating a target color on the can and logging its coordinates in each frame. The coordinates from each video in a particular experiment are then compiled into a matrix that comprises the data used in the PCA algorithm.

2 Theoretical Background

The governing equation for this system is known,

$$\frac{d^2 f}{dt^2} = -\omega^2 f \tag{1}$$

where $f(t)$ measures the displacement of the mass in the vertical direction. Equation 1 has the analytical solution,

$$f(t) = A\cos(\omega t + \omega_0) \tag{2}$$

The primary objective is to identify the motion of the paint can using PCA by removing redundancy and identifying the maximal variance in the data. In the first experiment there is one degree of freedom with the paint can oscillating in the vertical direction. However, the experimental setup includes three streams of data collecting locations of the paint can in two directions. This means there is a lot of redundancy in the incoming data.

Variance for a dataset $\vec{a} = (a_1, a_2, \ldots, a_n)$ is calculated by,

$$\sigma_a^2 = \frac{1}{n-1}\vec{a}\vec{a}^T \tag{3}$$

and correlation between two datasets $\vec{a}$ and $\vec{b}$ is calculated by the covariance,

$$\sigma_{ab}^2 = \frac{1}{n-1}\vec{a}\vec{b}^T \tag{4}$$

If two vectors are orthogonal then their dot product is zero, and so their covariance is zero. Conversely, if the dot product is large and thus the covariance is large then the two vectors are related. Therefore, covariance provides a measure of statistical dependency between two datasets.

For a matrix,

$$X \in \mathbb{R}^{m\times n} \tag{5}$$

the covariance matrix is calculated,

$$C_x = \frac{1}{n-1}XX^T \tag{6}$$

which can be written,

$$C_x = \begin{bmatrix} \sigma_{x,a}^2 & \sigma_{x,y,a}^2 & \sigma_{x,x,a,b}^2 & \cdots & \\ \vdots & \sigma_{y,a}^2 & & & \\ & & \sigma_{x,b}^2 & & \\ & & & \ddots & \\ & & & & \sigma_{y,c}^2 \end{bmatrix} \in \mathbb{R}^{m\times m} \tag{7}$$

The diagonal of the covariance matrix contains the variance of each data stream and the off-diagonals contain the covariance between pairs of the streams. This means the off-diagonals contain information about redundancy in the data. Moreover, since dynamical behavior is characterized by change and covariance is a measure of change between two data streams, then covariance(s) provides a measure of dynamical behavior in a given dataset. In this assignment, three cameras are capturing data of the same apparatus and so covariance provides a way to identify the dynamics of the paint can from the video data.

For the matrix X (equation 5), the singular value decomposition (SVD) is defined,

$$X = U\Sigma V^* \tag{8}$$

where Σ is a diagonal matrix of singular values σ that correspond to the square root of the eigenvalues $\left(\sigma_i = \sqrt{\lambda_i}\right)$ and are ordered based on size. U and V^* are orthonormal unitary transformations corresponding to the orientation of the incoming data. Using the proper orthogonal modes from the SVD, a low rank approximation can be calculated by the sum,

$$X_N \approx \sum_{j=1}^{N} \sigma_j \vec{u}_j \vec{v^*}_j \tag{9}$$

where N is a given rank. The 2-norm error for the approximation is given by,

$$\|X - X_N\|_2 = \sigma_{n+1} \,. \tag{10}$$

One way to think about SVD is as a least squares fit in higher dimensional space where the elements of Σ are ordered by their L^2 energy. The energy for a given mode can be calculated,

$$\text{energy}_i = \frac{\sum_{i=1}^{N} \sigma_i}{\sum_{i=1}^{n} \sigma_i} \,. \tag{11}$$

Therefore, diagonalization with SVD is a transformation to a new orthonormal basis and provides low rank approximation for a given dataset. To diagonalize the covariance matrix the transformation variable Y is defined,

$$Y = U^* X \tag{12}$$

and the variance in Y (see Appendix C for derivation) is calculated,

$$C_Y = \frac{1}{n-1} \Sigma^2 \,. \tag{13}$$

In this form the system is said to be written in terms of its principal components with the diagonal matrix Σ^2 containing variance measures in Y ordered based by size. This means that the dominant σ_i^2 values reflect directions of maximal variance and allow a low rank approximation of the spatiotemporal dynamics in the data.

3 Algorithm Implementation and Development

3.1 Position Measurements:

After importing the video data and extracting the data for a given frame, the first task is to identify the x, y coordinates of the paint can in the frame. Initially, I attempted to implement a tracking algorithm that got coordinates for the light on the top of the can by brightness. Based on the suggestion from the Catalyst discussion board (megan970), I modified the algorithm to track the yellow color on the paint can. There are five steps in the tracking algorithm that are repeated for each frame in a video:

1. Choosing target color
2. Getting individual frame
3. Building difference image
4. Identifying location of target color

The RGB values for the yellow color on the paint can are manually identified using a separate image editing application similar to Photoshop. Since the color spectrum is different for each of the videos, a subroutine `targetColor.m` is used to match RGB values to video. A given video is then passed into a loop that will process each frame. Inside the loop, the frame is converted to an image and double

precision format. Each color channel is then isolated and the target color value for the channel is subtracted off. The color channels are then combined into a 2-norm difference image where brightness reflects the distance from the target color. As seen in Figure 1, the target color takes a zero intensity value (black) and the rest of the image scales to reflect the difference from the target color. Using this difference image, the paint can location is identified (Figure 2) by finding the coordinates of the minimum value (i.e. the location of the pixel that is closest to the target color). For each camera, the location of the paint can is collected in vectors $(\vec{x}_i\,,\,\vec{y}_i)$. The data is then arranged into the matrix,

$$X = \begin{bmatrix} \vec{x}_a \\ \vec{y}_a \\ \vec{x}_b \\ \vec{y}_b \\ \vec{x}_c \\ \vec{y}_c \end{bmatrix} \epsilon\, \mathbb{R}^{m\times n} \tag{14}$$

where $i = a,\, b,\, c$ are the three cameras, m is the number of "sensors" tracking the paint can location, and n corresponds to the number of measurements in time (meaning the length of the video). In these experiments, the location is tracked in (x, y) for three camera and so the $m = 6$. Once data for the paint can location is obtained for each of the cameras in a given test scenario, the final task is to synchronize the first frames and trim the data to the length of the shortest video so that covariances between the videos can be calculated.

To speed processing, the use of cropping to the region containing the motion of the paint can is explored (Figure 2). The challenge with using cropping is the effort to manually identifying the region in each video that contains the entire range of motion of the paint can. The observed run time using cropped images is approximately twice as fast due to the data reduction, however, nearly identical results are obtained for the paint can location from both cropped and uncropped videos. Therefore, to reduce the upfront manual workload and prevent potential error from cropping out necessary information, all further analyses use uncropped videos.

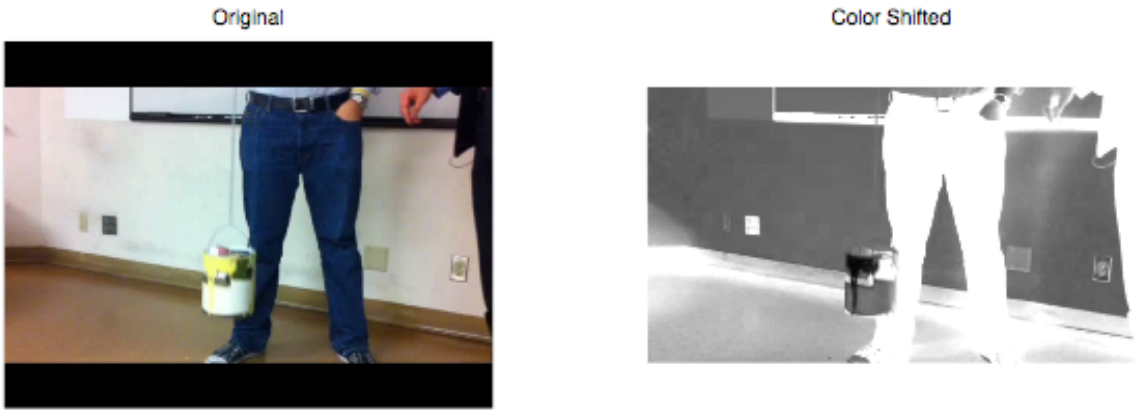

Figure 1. For a specified target color (in this case the yellow on the paint can), a difference image is constructed using the 2-norm of the difference from the target color. This creates an image (right) where the target color takes the minimum value and the rests of the image is scaled by difference from the target color.

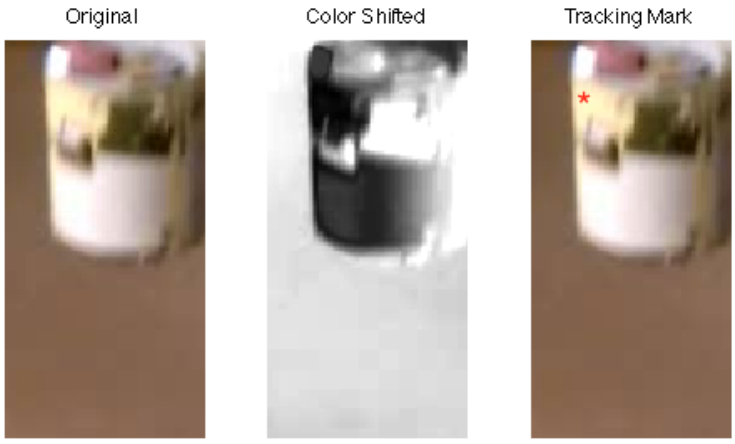

Figure 2. The paint can location (right) is identified by finding the minimum value in the difference image (center) where the scale is based on the difference from the target color (yellow on the can).

3.2 Principal Component Analysis:

There are three steps in the PCA algorithm:

1. Subtracting the mean from the data
2. SVD of the data
3. Output/visualization of results

Implementation is greatly simplified because calculation of the mean and the SVD operation are performed using built in MATLAB functions (see Appendix A). The outputs from the `svd`() function are the U, Σ, V^* matrices. The principal component projections ($Y = U^*X$), diagonal variances $\left(\sigma_i^2 = \lambda_i\right)$, and proper orthogonal modes (equation 9) are plotted in the subroutine `plotProjections.m`. The energy contained at a given mode is calculated using equation 11.

4 Computational Results

4.1 Test 1

In Test 1, the paint can is set to oscillate in a purely vertical direction and so we should expect maximal variance in one component. Plotting the raw location tracking data (Figure 3) clearly shows consistent oscillation in the vertical direction and minimal movement in the horizontal for all three cameras. Energies of the diagonal variances from the PCA were,

σ_i^2	1	2	3	4	5	6
energy	0.9499	0.9744	0.9874	0.9959	0.9986	1.0000

indicating that ~95% of the variance are captured in the first principal component. Figure 4 shows that the variance in the first component is almost 10^2 greater than the others. Plotting of the principal component projections (Figure 5) and the proper orthogonal modes (Figure 6) confirm that the first component or the rank one approximation captures the one dimensional oscillation in this test case.

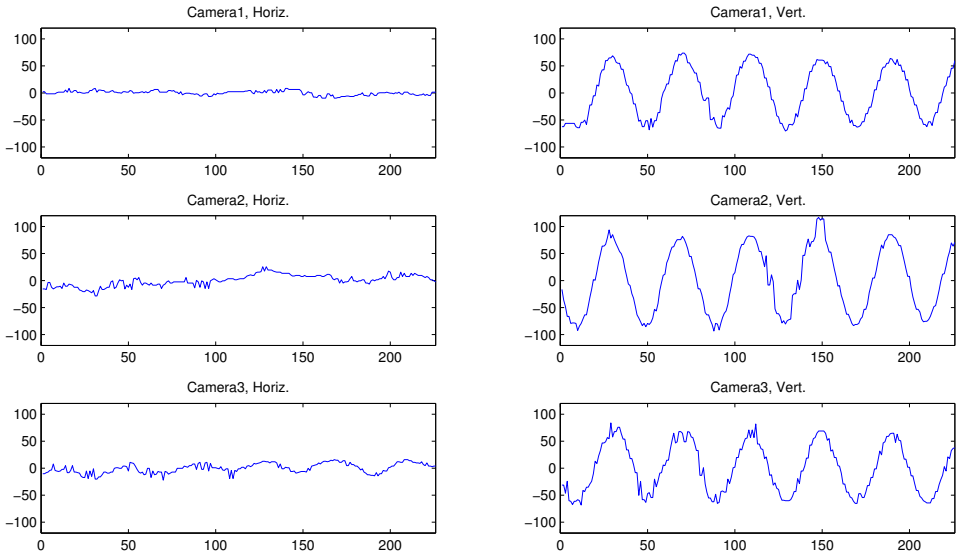

Figure 3. For Test 1, the raw location tracking data for each camera clearly shows oscillation in the vertical (right) direction and minimal movement in the horizontal (left).

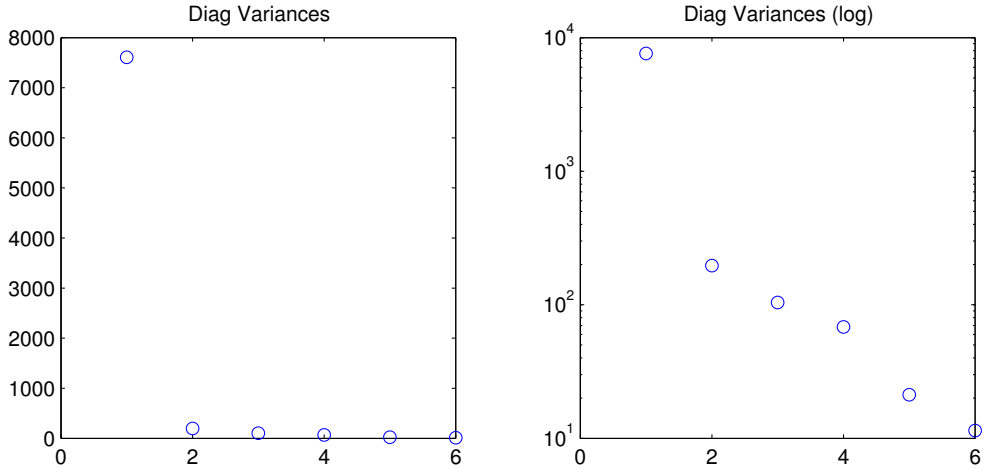

Figure 4. For Test 1, the diagonal variances $\left(\sigma_i^2 = \lambda_i\right)$ show that the first component captures most of the variance with almost 10^2 more variance than the others.

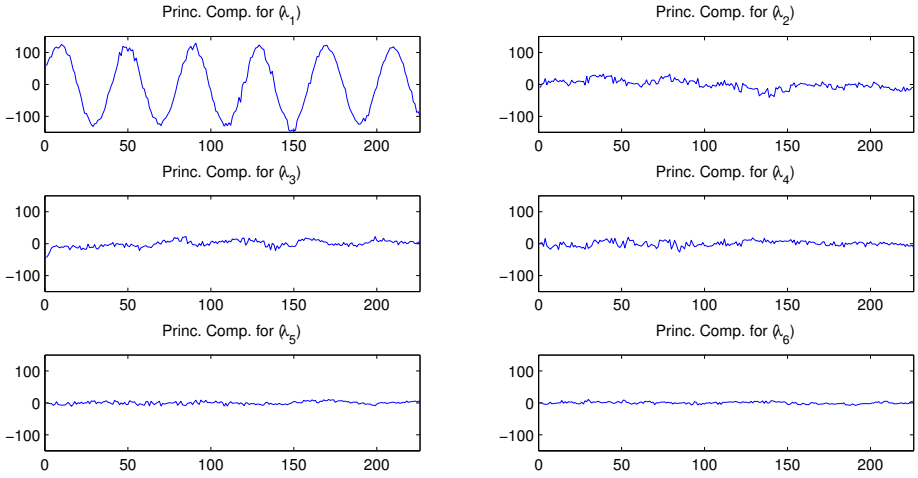

Figure 5. For Test 1, the principal components (Y) shows that the first component (top left) clearly captures the oscillatory behavior of the can.

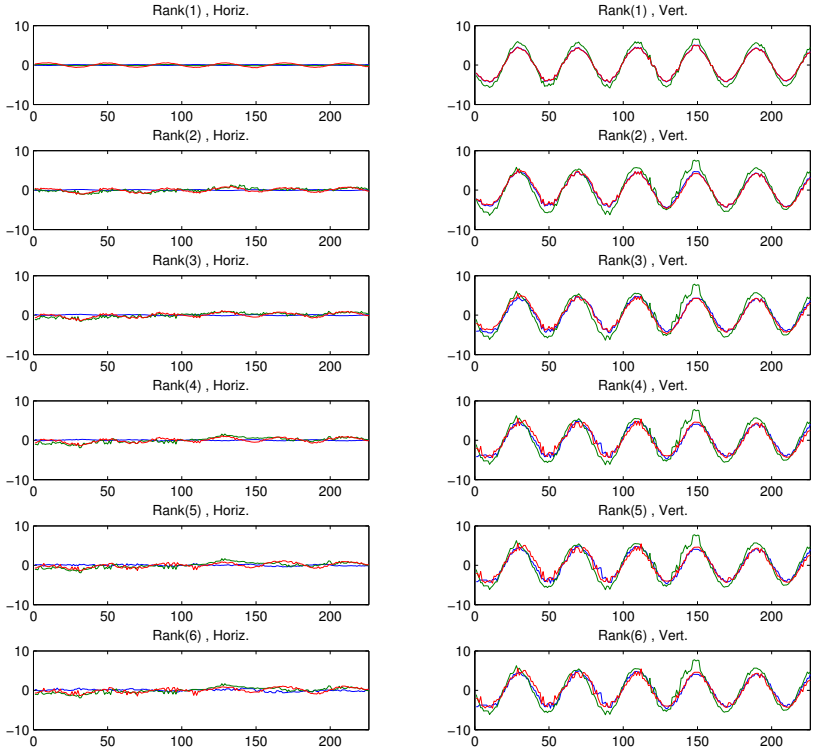

Figure 6. For Test 1, the proper orthogonal modes shows that a rank one approximation (first row) captures that vertical (right) oscillatory behavior and is a close match to the full rank (bottom row) data.

4.2 Test 2

In Test 2, the paint can is set to oscillate in a purely vertical direction just as in Test 1 and so we should expect maximal variance in one component. However, the camera operators intentionally shook the camera while recording introducing noise in the data. The intention of this scenario is to explore how noise affects the PCA algorithm. Plotting the raw location tracking data (Figure 7) shows oscillation in the vertical direction that is consistent but much more erratic than Test 1. The horizontal direction shows some movement but no consistent oscillatory behavior. Therefore, the noise from the camera shake diminishes the quality of the data in both the horizontal and vertical directions. Energies of the diagonal variances from the PCA were,

σ_i^2	1	2	3	4	5	6
energy	0.6527	0.8079	0.8985	0.9468	0.9796	1.0000

indicating that only ~65% of the variance is captured in the first principal component compared to ~95% for Test 1. Figure 8 shows that the variance in the first component is dominant but has less than 10^1 more variance than the others. Plotting the principal component projections (Figure 9) shows that both the first and second components show oscillatory behavior. The proper orthogonal modes (Figure 10) confirm that a rank one approximation does capture the vertical oscillation, but the rank four approximation (corresponding to ~95% energy) is a better match to the full rank (six) data. These results indicate that the noise from the camera shake diminishes performance.

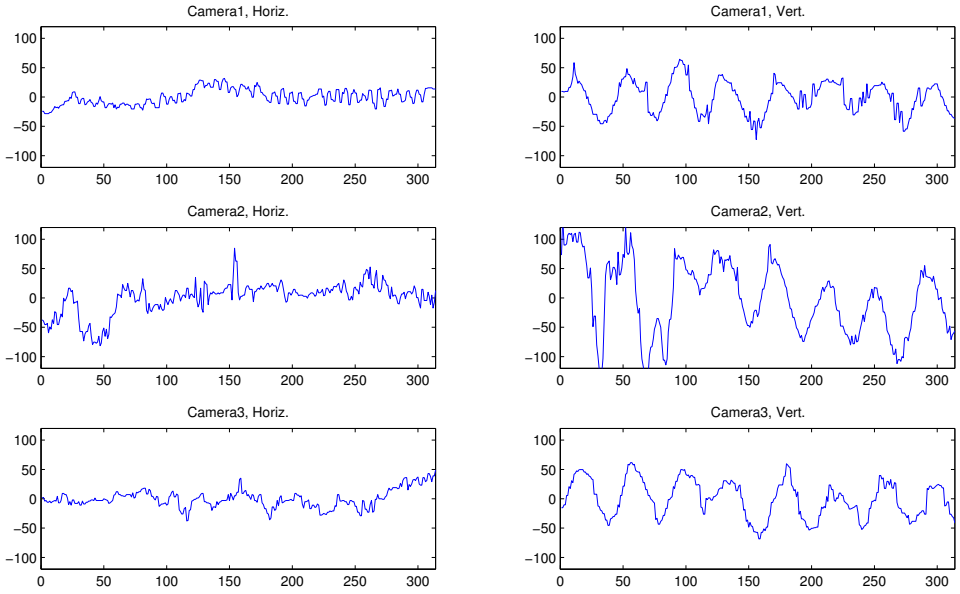

Figure 7. For Test 2, the raw location tracking data for each camera clearly shows oscillation in the vertical (right) direction but the noise-inducing camera shake diminishes its consistency and introduces movement in the horizontal direction (left).

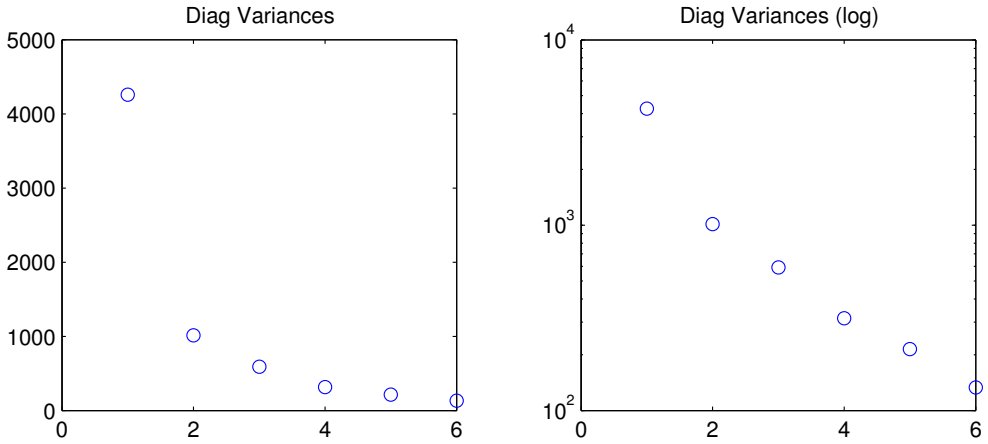

Figure 8. For Test 2, the diagonal variances $\left(\sigma_i^2 = \lambda_i\right)$ show that the first component is dominant but less than 10^1 more than the others.

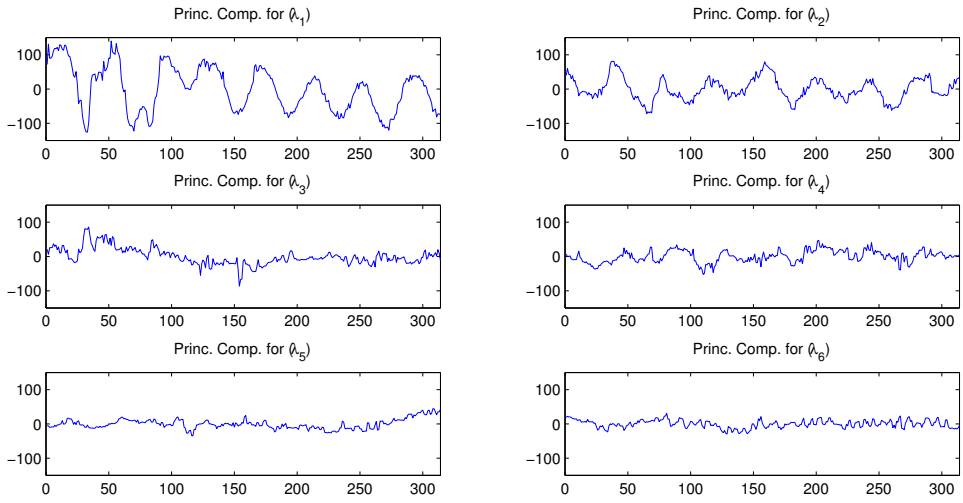

Figure 9. For Test 2, the principal component projections (Y) shows that both the first and second components show oscillatory behavior.

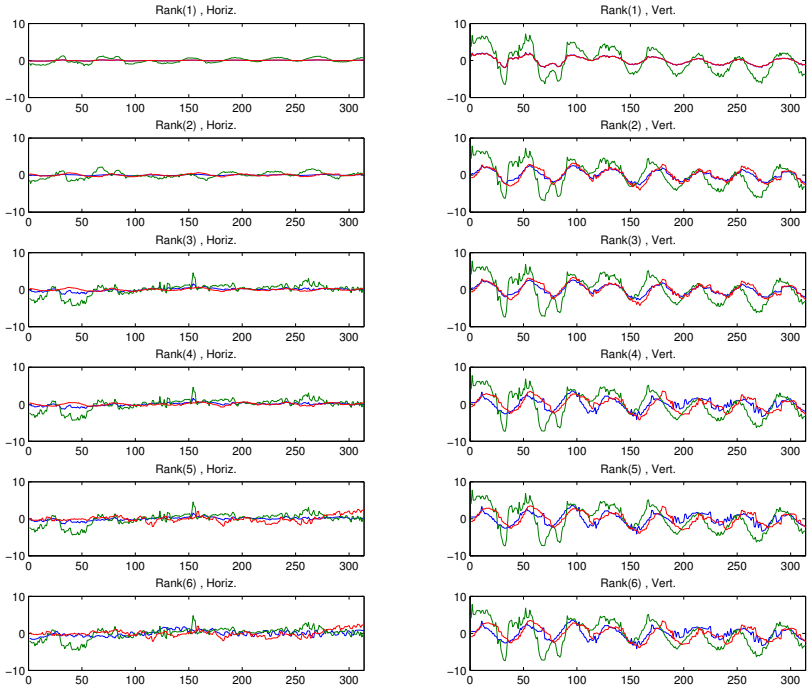

Figure 10. For Test 2, the proper orthogonal modes show that a rank one approximation does capture vertical oscillation, but the rank four approximation (forth from top) is a better match to the full rank (bottom row).

4.3 Test 3

In Test 3, the paint can is set to oscillate in the vertical direction just as in Test 1-2 but is also set to oscillate in the horizontal direction and so we should expect maximal variance for two components. Plotting the raw location tracking data (Figure 11) shows oscillation in both the vertical and horizontal directions for all three cameras.

Energies of the diagonal variances from the PCA were,

σ_i^2	1	2	3	4	5	6
energy	0.4756	0.7226	0.9009	0.9580	0.9957	1.0000

indicating that ~48% of the variance is captured in the first principal component, ~72% in the second, and that it takes four modes to reach 95%. Figure 12 shows that the variance in the first component is dominant but that the trend is approximately linear for the first five modes. Plotting the principal component projections (Figure 13) shows that the first three components contain oscillatory behavior. The proper orthogonal modes (Figure 14) show that a rank two approximation captures the vertical and horizontal oscillation, but the rank four approximation (corresponding to ~95% energy) is a better match to the full rank (six) data. These results indicate that the the PCA algorithm effectively captures the horizontal and vertical oscillation with two components, but better accuracy is achieved with higher rank data.

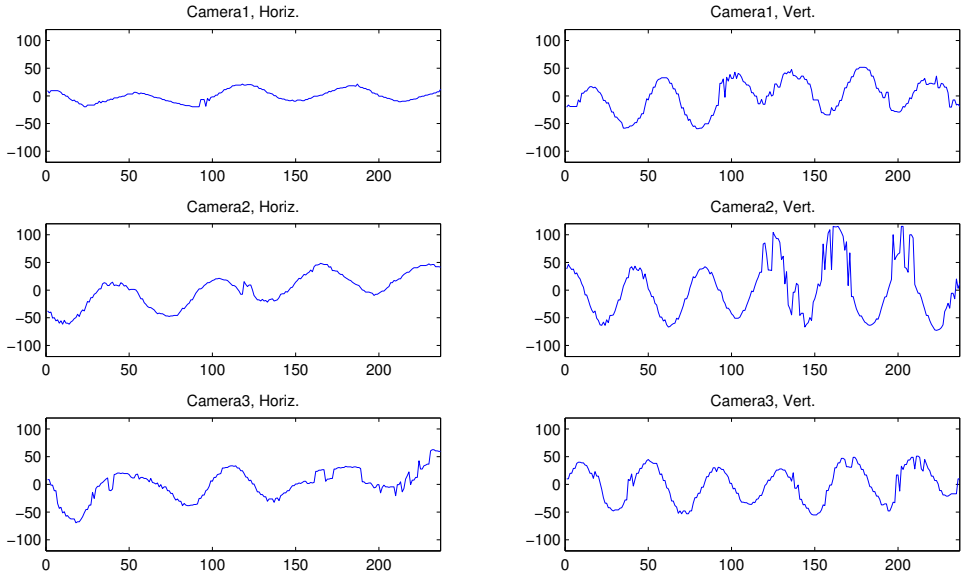

Figure 11. For Test 3, the raw location tracking data clearly shows oscillation in both the vertical (right) and horizontal (left) directions for all three cameras.

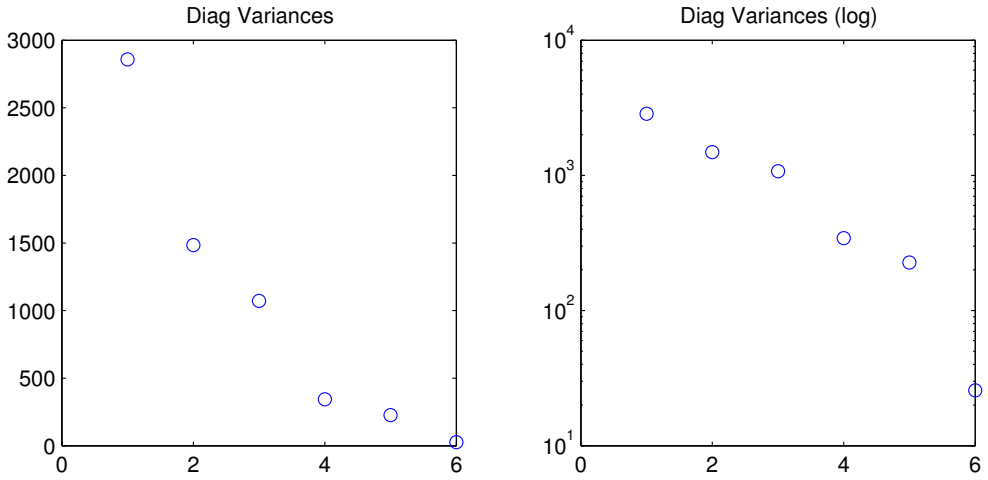

Figure 12. For Test 3, the diagonal variances $\left(\sigma_i^2 = \lambda_i\right)$ show that the variance in the first component is dominant but that the trend is approximately linear for the first five modes.

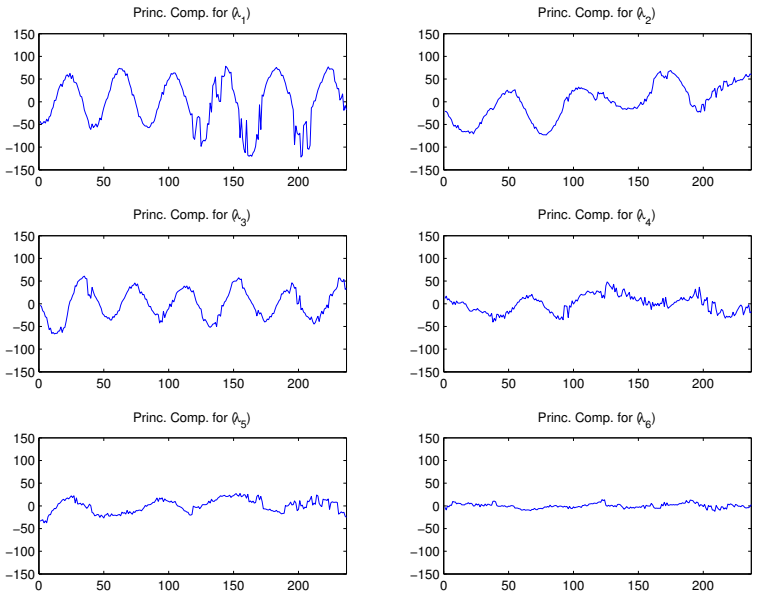

Figure 13. For Test 3, the principal component projections (Y) shows that the first three components contain oscillatory behavior.

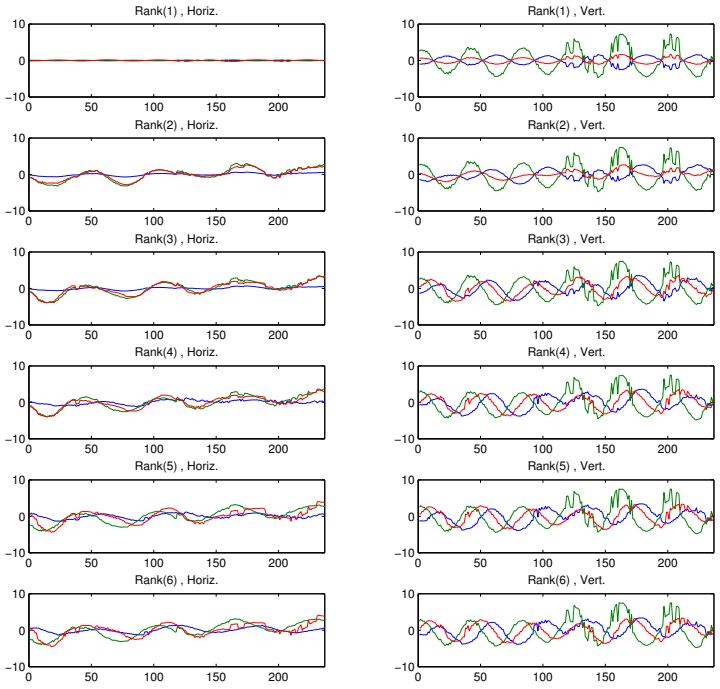

Figure 14. For Test 3, the proper orthogonal modes show that rank two (second from top) captures the vertical and horizontal oscillation, but that rank four (fourth from top) is a better match to the full rank (bottom row).

4.4 Test 4

In Test 4, the paint can is set to oscillate in the vertical and horizontal directions just as in Test 3 but is the horizontal direction is set to rotation and so we should expect maximal variance for two or more components. Plotting the raw location tracking data (Figure 15) shows oscillation in both the vertical and horizontal directions for all three cameras. However, the oscillation in this erratic and at certain time points shows that the location tracking algorithm may have made substantial errors.

Energies of the diagonal variances from the PCA were,

σ_i^2	1	2	3	4	5	6
energy	0.5258	0.8694	0.9297	0.9668	0.9892	1.0000

indicating that ~52% of the variance is captured in the first principal component, ~87% in the second, and that it takes four modes to reach 95%. Figure 16 shows that the variance in the first two components is dominant but that the trend is still approximately linear for the six modes. Plotting the principal component projections (Figure 17) shows that the first four components contain clear oscillatory behavior. The proper orthogonal modes (Figure 18) show the rank two approximation captures the horizontal and vertical behavior and is a decent match to the full rank (six) data. These results indicate that the the PCA algorithm effectively captures the horizontal and vertical oscillation with two components, but better accuracy is achieved with nearly full rank data.

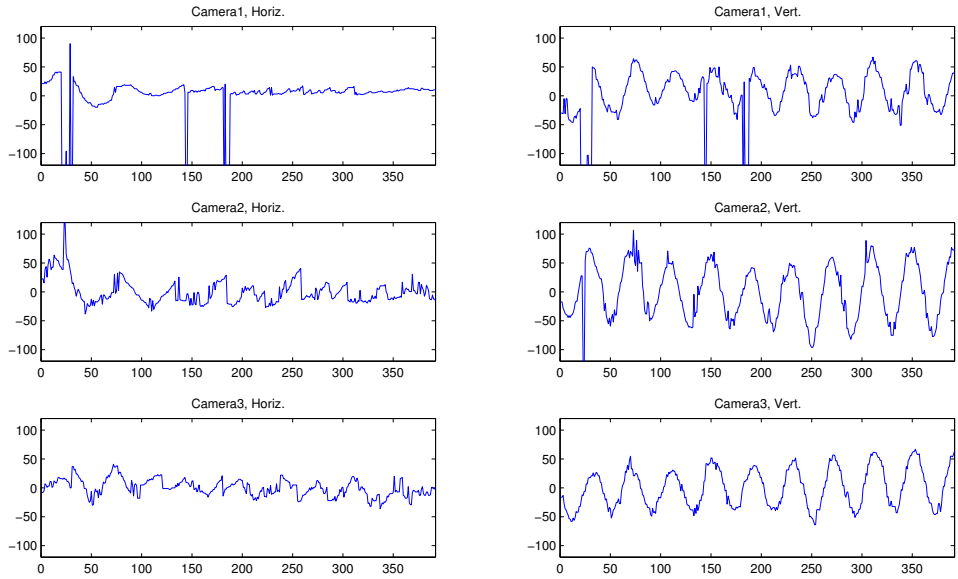

Figure 15. For Test 4, the raw location tracking data shows clear oscillation in the vertical (right) direction and erratic oscillation in the horizontal (left) for all three cameras.

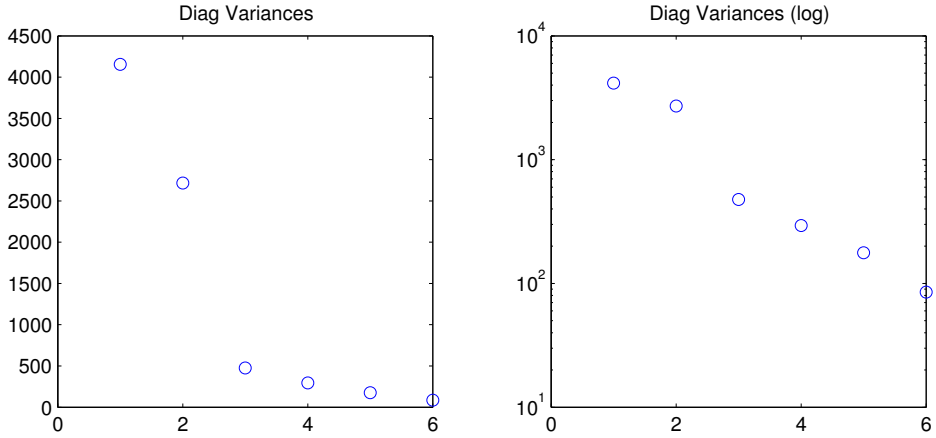

Figure 16. For Test 4, the diagonal variances $\left(\sigma_i^2 = \lambda_i\right)$ show that the variance in the first two components are dominant but that the trend is approximately linear for the six modes.

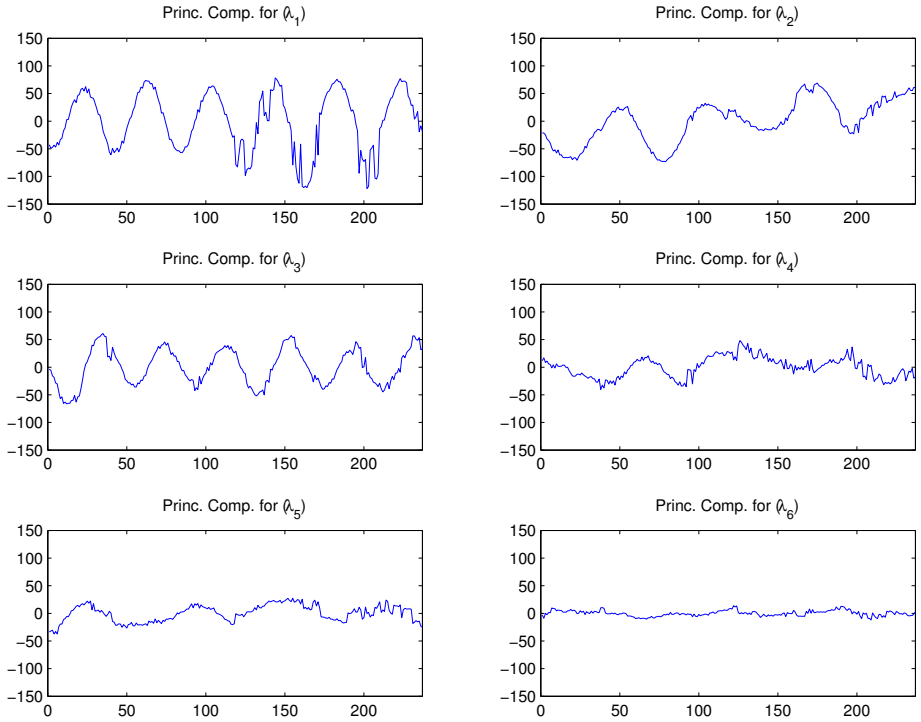

Figure 17. For Test 4, the principal component projections (Y) shows that the first four or five components contain oscillatory behavior.

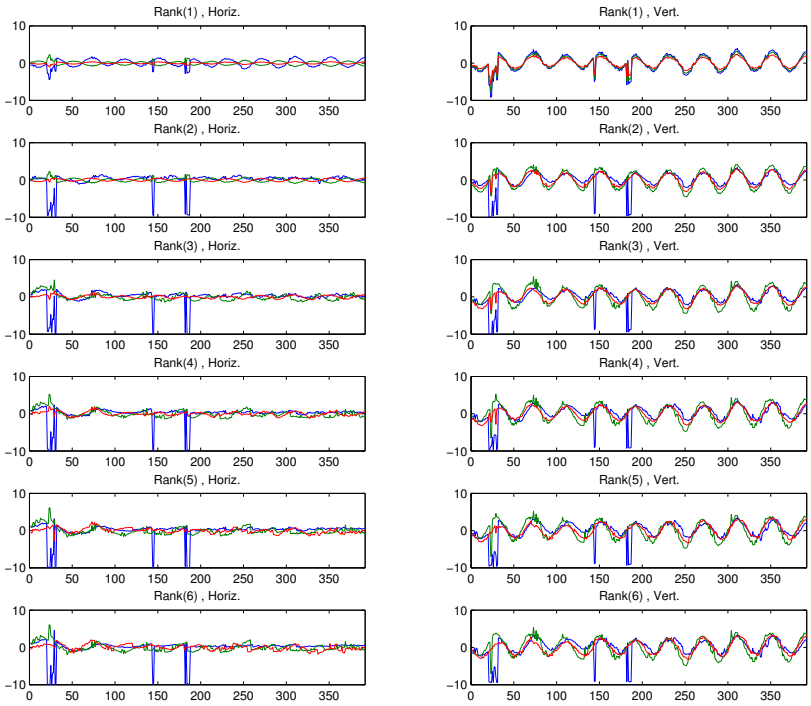

Figure 18. For Test 4, the proper orthogonal modes show that a rank two approximation (second from top) captures the vertical and horizontal oscillation and is a decent match to the full rank (bottom row) data.

5 Summary and Conclusions

Ultimately, this project shows that principal component analysis (PCA) can be used to reduce redundancy and find directions of maximal variance in complex data. This allows low rank approximations of multidimensional dynamical behavior. One of the primary obstacles encountered was the measurement of the paint can location in the video frames. This algorithm adopts a suggestion from a classmate to track the yellow color on the paint can. As with any experiment, experimental precision is critical to minimizing error attributable to the procedure. I suspect some of the erratic behavior, especially in Test 4 with the rotation, may be the result of errors with the location tracking algorithms. However, the paint can apparatus and the videography were not necessarily set up for precision measurement and all of the observed results match expected behavior with reasonable accuracy indicating the tracking algorithm used was sufficient for the purposes of this assignment.

Appendix A

diag Used to extract the diagonals from a matrix.

double Used to convert 8-bit integer image data to double precision format.

frame2im Used to covert a movie frame to an image.

imshow Used to plot image data.

load Loads data from an external file.

ind2sub Used to convert a linear index (in this case the minimum value in the difference image) into corresponding 2D subscripts.

linspace Used to build a linear vector with $n + 1$ points for the spatial domain. The vector is then trimmed to n points due to the periodic boundaries.

mean Used to calculate the mean for each row of the location coordinate data.

min Used to find the minimum value in a given array.

num2str Used to convert a number to a string.

plot Used to plot the various parameters agains time or frequency.

repmat Used to replicate and tile an array for the means.

reshape Reshapes vector or matrix for given dimensions.

semilogy Used to make a plot with the y-axis on a logarithmic scale.

size Used to get the number of rows and columns for a given matrix.

strcat Used to concatenate a string.

subplot Used to produce plot arrays.

sum Used to calculate sums.

switch Conditional structure used to vary code execution based on a specified tag.

tic/toc Used to time the operations.

uint8 Used to convert data to 8-bit integer format.

zeros Used to build vectors and matrices filled with "0".

Appendix B

The main script (MATLAB) for this project `main.m`:

```
clear all; close all; tic

%%=============================================================     initialization

test = 4
rankExpt = 2;
numCameras = 3;

cropBndrys = [220 300; 120 250; 240 280];
cropCheck = 'noCrop';

color = targetColor(test);

numFrames = zeros(numCameras,1);
X = zeros(numCameras*2,500);

%%=================================================================     coordinates

fig = 0;
for camera=1:numCameras

    [mov numFrames(camera)] = loadMovie(test,camera,cropCheck,cropBndrys);
    %playMovie(mov,numFrames(camera));

    for j=1:numFrames(camera)

        % get frame
        F = double(frame2im(mov(j)));
        %figure(1), imshow(uint8(X)), title('Original'), drawnow;

        [y x d] = size(F);

        % shift color spec, floors yellow on can
        Frd = F(:,:,1) - color(camera,1);
        Fgd = F(:,:,2) - color(camera,2);
        Fbd = F(:,:,3) - color(camera,3);
        Fdiff = sqrt( Frd.^2 + Fgd.^2 + Fbd.^2 );
        %plotShifted(F,Fdiff,2);

        % find x,y coords on min value
        [minVal Iind] = min(reshape(Fdiff,1,x*y));
        [Iy Ix] = ind2sub(size(Fdiff),Iind);
        if camera == 3
```

```
            % correct for camera 3 orientation
            X(2*camera-1,j) = -1*Iy;
            X(2*camera,j) = Ix;
        else
            X(2*camera-1,j) = Ix;
            X(2*camera,j) = Iy;
        end

    end

end

% trim camera 2 data to match 1,3
lag = numFrames(2) - numFrames(3);
X(3,1:end-(lag-1)) = X(3,lag:end);
X(4,1:end-(lag-1)) = X(4,lag:end);

% trim all data to shortest video
n = min(numFrames);
X = X(:,1:n);

%%=====================================================================     PCA

[m,n] = size(X);

% subtract off mean (PCA)
mn = mean(X,2);
X = X-repmat(mn,1,n);

% SVD
[U,S,V] = svd(X/sqrt(n-1));
Y = U' * X;
lambda = diag(S).^2;

% energies
energy = zeros(m,1);
for j=1:m
energy(j) = sum(lambda(1:j))/sum(lambda);
end
energy

%%=====================================================================     plotting

plotTracking(X,3);

% plot variances
figure(4);
subplot(1,2,1), plot(lambda,'bo'), title('Diag Variances');
subplot(1,2,2), semilogy(lambda,'bo'), title('Diag Variances (log)');
```

```
plotProjections(Y,U,S,V,'princComp',3,5);

plotProjections(Y,U,S,V,'array',6,6);

%plotProjections(Y,U,S,V,'best',rankExpt,7);

%%=====================================================================     end
toc
```

The MATLAB script `loadMovie.m`:

```
function [mov numFrames] = loadMovie(test,camera,cropCheck,cropLim,dx,dy)

    pathName = '/Volumes/Storage HD/Courses/AMath 582/Homework 4/';
    varNameBase = 'vidFrames';

    fileName = strcat(pathName,'cam',num2str(camera),'_',num2str(test),'.mat');

    load(fileName);

    vidFrames = eval(strcat(varNameBase,num2str(camera),'_',num2str(test)));
    numFrames = size(vidFrames,4);

    switch cropCheck

    case 'crop'
        if camera == 3
            dy = 80;  dx = 160;
        else
            dy = 160;  dx = 80;
        end

        crop = [cropLim(camera,1) cropLim(camera,1)+dy...
        cropLim(camera,2) cropLim(camera,2)+dx];

        for k=1:numFrames
            mov(k).cdata = vidFrames(crop(1):crop(2),crop(3):crop(4),:,k);
            mov(k).colormap = [];
        end

    case 'noCrop'
        for k=1:numFrames
            mov(k).cdata = vidFrames(:,:,:,k);
            mov(k).colormap = [];
        end

    end
end
```

The MATLAB script `playMovie.m`:

```
function playMovie(mov,numFrames)
    for j=1:numFrames
        F = frame2im(mov(j));
        imshow(F), drawnow;
    end
end
```

The MATLAB script `plotProjections.m`:

```
function plotProjections(Y,U,S,V,plotType,rankVal,fig)

    switch plotType

    case 'princComp'
        [m n] = size(Y);
        for j=1:m
            figure(fig), subplot(m/2,2,j), plot(Y(j,:)');
            axis([0 n -150 150]), title(strcat('Princ. Comp. for ...
                (\lambda_',num2str(j),')'));
        end

    case 'array'
        % plot array of modal projections
        for j=1:rankVal

            soln = U(:,1:j)*S(1:j,1:j)*V(:,1:j)';
            solnX = [soln(1,:); soln(3,:); soln(5,:)];
            solnY = [soln(2,:); soln(4,:); soln(6,:)];
            [m n] = size(soln);

            figure(fig), subplot(rankVal,2,(2*j-1)), plot(solnX');
            axis([0 n -10 10]),title(strcat('Rank(',num2str(j),') , Horiz.'));
            figure(fig), subplot(rankVal,2,(2*j)), plot(solnY');
            axis([0 n -10 10]),title(strcat('Rank(',num2str(j),') , Vert.'));

        end

    case 'best'
        % plot best dynamics
        for j=1:rankVal
            soln = U(:,1:j)*S(1:j,1:j)*V(:,1:j)';
            solnX = [soln(1,:); soln(3,:); soln(5,:)];
            solnY = [soln(2,:); soln(4,:); soln(6,:)];
            [m n] = size(soln);
        end
```

```
        if rankVal == 1
            figure(fig), plot(solnY'), axis([0 n -8 8]);
            title(strcat('Rank(',num2str(rankVal),') approx. of dynamics in ...
                Vert.'));
        else
            figure(fig), subplot(1,2,1), plot(solnX'), axis([0 n -10 10]);
            title(strcat('Rank(',num2str(rankVal),') approx. of dynamics in ...
                Horiz.'));
            figure(fig), subplot(1,2,2), plot(solnY'), axis([0 n -10 10]);
            title(strcat('Rank(',num2str(rankVal),') approx. of dynamics in ...
                Vert.'));
        end
    end
    drawnow;
end
```

The MATLAB script `plotShifted.m`:

```
function plotShifted(F,Fdiff,fig)

    figure(fig);
    subplot(1,2,1), imshow(uint8(F)), title('Original');
    subplot(1,2,2), imshow(uint8(Fdiff)), title('Color Shifted');
    %subplot(1,3,3), imshow(uint8(F)), title('Tracking Mark');
    drawnow;

end
```

The MATLAB script `plotTracking.m`:

```
function plotTracking(X,fig)

    [m n] = size(X);

    for j=1:(m/2)

        figure(fig), subplot(m/2,2,(2*j-1)), plot(X((2*j-1),:)');
        axis([0 n -120 120]), title(strcat('Camera ',num2str(j),', Horiz.'));

        figure(fig), subplot(m/2,2,(2*j)), plot(X((2*j),:)');
        axis([0 n -120 120]), title(strcat('Camera ',num2str(j),', Vert.'));

    end

end
```

The MATLAB script `targetColor.m`:

```
function color = targetColor(test)

    if test == 1
        color = [255 249 202; 253 255 134; 230 227 182];
    end
    if test == 2
        color = [253 255 208; 254 255 148; 247 246 177];
    end
    if test == 3
        color = [252 257 207; 250 255 132; 241 243 178];
    end
    if test == 4
        color = [255 249 213; 241 254 142; 207 213 157];
    end

end
```

Appendix C

Diagonalization of the Covariance Matrix:

The SVD for the matrix of the paint can locations is defined,

$$X = U\Sigma V^*$$

with the transformation variable defined,

$$Y = U^*X\,.$$

The covariance matrix of Y is defined,

$$C_Y = \frac{1}{n-1}YY^T = \frac{1}{n-1}\left(U^*X\right)\left(U^*X\right)^T = \frac{1}{n-1}U^*XX^TU$$

and solving the eigenvalue problem,

$$XX^TU = U\Sigma^2$$

gives,

$$C_Y = \frac{1}{n-1}U^*U\Sigma^2 = \frac{1}{n-1}\Sigma^2\,.$$