AMATH 482 Homework 3

Shane Fretwell

February 23, 2021

Abstract

Many of the physical phenomena we encounter have well-known formulas underlying their behavior; voltage is the product of the current across a resistor and its resistance, buoyancy is determined by the ratio of densities, and the position of a mass on a spring over time is given by Hooke's Law. However, we do not always have the equations for modelling a phenomenon, and in these cases, it is often most effective to take a statistical approach. Here, we will analyze the movement of a mass on a spring without the related physical laws and compare the results with what we know about simple harmonic motion.

1 Introduction and Overview

We are all familiar with the concept of correlation, but there are some ways to think about this concept that are still unexpected. Singular Value Decomposition is an example of this, describing correlation as variance along a basis of orthogonal vectors. We will explore this concept by retrieving the one-dimensional movement of a paint bucket on a spring from a series of videos that introduce different types of noise to the linear movement. These types include shaking of the camera, off-axis movement of the mass, and rotation. Even in the presence of this noise, we can use SVD to determine the vector along which variation is maximized, which represents the direction of correlation between the movement of the bucket in each of the recordings. We will see that this is the direction of the mass's linear oscillation.

2 Theoretical Background

2.1 Color Tracking

Given a picture of an object that is colored sufficiently differently from its background, we can easily mark the location of that object by color tracking. First, we compare the color values of every pixel in the video, keeping track of the pixels whose color matches that of the object. With this alone, we have a general idea of where the object is in the frame, but to get a more precise location (x, y), there is one more step. We simply take the median of the x coordinates of the matching pixels, and the same for the y coordinates, and call these media (x, y), respectively. We use the median because this measure is less swayed by outliers in the data than the mean, and outliers are likely to have nothing to do with the location of the object.

2.2 Singular Value Decomposition

The geometric interpretation of a linear transformation matrix A is that the vector multiplied by A is stretched and

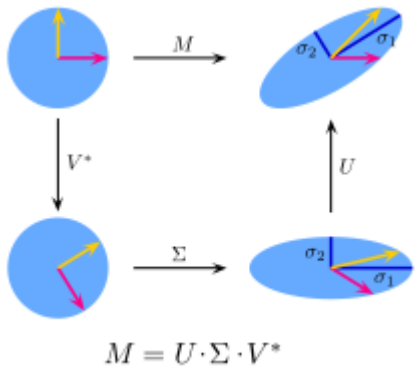

Figure 1: Geometric interpretation of SVD

rotated. There are matrices whose images are simple rotations of their pre-images, and there are matrices whose n image coordinates are the product of n positive scalars with each of the pre-image's n coordinates. The premise of the SVD is to separate any invertible transformation into three simpler components: a rotation, a positive coordinate-wise scaling, and another rotation. Figure 1 illustrates this decomposition of a transformation matrix M on the unit disc. Note that the unit disc is a set of vectors, and thus can be represented as a matrix D whose columns are all of the vectors that comprise the disk. The matrix M directly transforms the unit disc and the canonical unit vectors into their image under M in one matrix multiplication MD. Multiplying the unit disc by three separate matrices in sequence and arriving at the same image under M demonstrates the principle of SVD, since $MD = U\Sigma V^*D$ implies $M = U\Sigma V^*$. Multiplication by V^* rotates the unit disk and Σ stretches the disk along each of the vectors that form the new basis, creating an ellipse whose principal axes are of length σ_1 and σ_2, the values on the diagonal of Σ. Finally, this ellipse is rotated, through multiplication by U, into the orientation of the image of D under M (1).

2.2.1 The matrices U, ∑, and V*

The individual matrices U, Σ, and V have meaning beyond their transformative properties of scaling and rotation. Importantly, the columns of U are the unit basis vectors of V^*D, which also makes them the vectors that describe the orientation of the principal axes of the ellipse MD. Σ is a diagonal matrix, and the values on its diagonal are half of the width of the principal axes. By convention, the columns of the three matrices U, Σ, and V are organized such that the diagonal entries of Σ are in decreasing order, and ensuring that the columns of U and V are similarly permuted such that the equation $M = U\Sigma V^*$ still holds.

2.2.2 SVD in the context of Statistics

To illustrate the significance of SVD in statistics, we think of the columns of A as a 'cloud' of data points that looks like MD – call it a scatter plot of weight vs. height. Then, we can see that these measurements of weight and height are strongly correlated. This strong correlation means that we can describe each data point with fair accuracy by simply noting its distance along the line of best fit, or more precisely, its projection onto the line of best fit. This is a one-dimensional approximation of the 2-D data. From lecture 11, we have the theorem of equation 1 where r is the rank of A.

$$A = \sum_{k=1}^{r} \sigma_k u_k v_k^* \tag{1}$$

This theorem also states that partial sum up to i is the best i-rank approximation of A. So, when we project onto the line of best fit, we are taking only the first term of this sum. Since this would be a rank-1 approximation, the new data matrix, call it $\bar{A}$, has a single spanning vector – the first column of the matrix U where $A = U\Sigma V^*$. Thus, the projections of the data onto the line of best fit are easily read off from the vector $u_1 * A$. For simplicity of explanation, the data matrix A has been in $\mathbb{R}^{2xm}$, but this generalizes to $\mathbb{R}^{nxm}$ for higher dimensional data and left-multiplying by $u_1 *$ will still make for this easy reading of projections onto the line of best fit (1).

3 Algorithm Implementation and Development

3.1 Color Tracking of the Bucket

The bucket in the videos is a used paint bucket that has bright yellow paint along its sides, such that the paint is visible to an observer from nearly any angle. There are in fact only four frames among a total of 3414 frames analyzed wherein there was not enough yellow paint visible to track the bucket's position in the frame. Thus, by tracking the exact colors of the yellow paint as described in section 2.1, we can mark the bucket's location as (x, y) within each of the 3410 tractable frames.

Algorithm 1: Color Tracking – corresponds to trackByColor in Appendix A

for each frame

% Gives a vector of x coordinates and a vector of y coordinates of matching pixels

```
    % colormask is one of four automatically generated color masking functions – see Appendix A
    [y,x] = find(colormask(frame));
    trackedPosition(i) = [median(x), median(y)];
end
```

3.1.1 Color Masks

For determining which pixels in the image were within the right color range, MATLAB provides a package called Color Thresholder. This package allows you to export automatically generated functions that take in an image and return an array with entries for each pixel in the image. If the entry for a pixel is 1, then that pixel's color matched the colors that the given function is searching for. You can load an image into this package and select colors from the image's color space that you want a function to search for – the selection process is shown in figure 2. This is how the color masking functions for tracking the bucket were generated. Indeed, it did require different masking functions for different videos due to apparent differences in lighting and cameras.

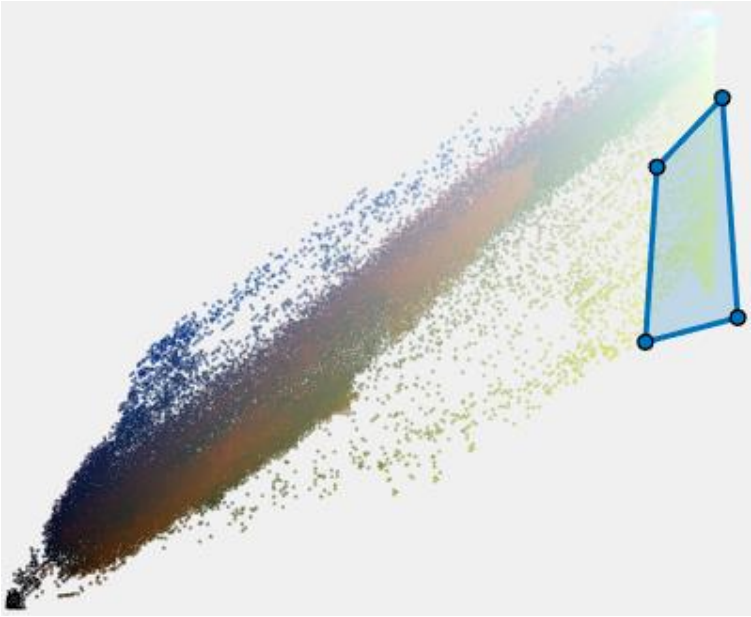

Figure 2: The color selection tool within MATLAB's color thresholding tool.

3.2 Applying the Concepts of SVD

Since the position of the bucket over time is largely one-dimensional, we know that the position of the bucket in each of the videos at a given point in time should be heavily correlated. The x-y axes of each video were oriented differently due to the different camera positions, but they are all oriented such that they can perceive the movement of the bucket towards and away from the floor, i.e., none of them are oriented directly above or below the bucket. This means that a movement of the bucket towards the floor might be a movement in the negative x direction of camera one, along the x-y line in camera two, and in the positive y direction for camera three. Regardless, these movements as perceived by the cameras are clearly going to be correlated. So, we take observations of the position of the bucket in each camera over time – the goal is to have a position vector for each frame of the videos. This vector contains the x and y coordinates of the bucket in each of the three cameras synchronized in time. Then, we have data that we can put into a matrix A and decompose to analyze.

3.2.1 Synchronizing in Time

The videos from each camera were of different durations and started at different times during the movement of the bucket. To motivate synchronizing the videos, think of the height vs. weight example. To get an accurate correlation between the two, you need to measure the same person's height and weight; if you measure the height of one person and plot it against the weight of his brother, then genetics may allow the correlation to show through, but the correlation will be much clearer when each data point is measurements entirely from the same person. The same problem arises if we do not ensure that the position vectors are synchronized in time. The coordinates from each video must be measuring the position of the bucket at the same point in time. To accomplish this, it was useful to observe the moment at which the bucket first switched directions in each of the videos and line up this point in time within the position vectors.

3.2.2 Obtaining the Linear Movement

Once we have the position vectors, we decompose the data matrix to retrieve the matrix U and multiply the data by $u_1 *$. To

Algorithm 2: Decomposition

```
[U,~,~] = svd(A, 'econ');
Displacement = transpose(U(:,1))*A;
```

4 Computational Results

In figure 3 we have plots of the linear displacement of the bucket over time from each of the four cases. We can see clear harmonic motion here, and while the different types of noise did affect the smoothness and symmetry of the observed motion, these are very promising results. It is clear that what is being measured in these plots, for the most part, is the true linear movement of the bucket as it oscillates on the spring. Even with the differing height of the peaks of the waveforms, we see a consistent period of 20 frames that aligns with the theories from physics about how a mass on a spring should behave.

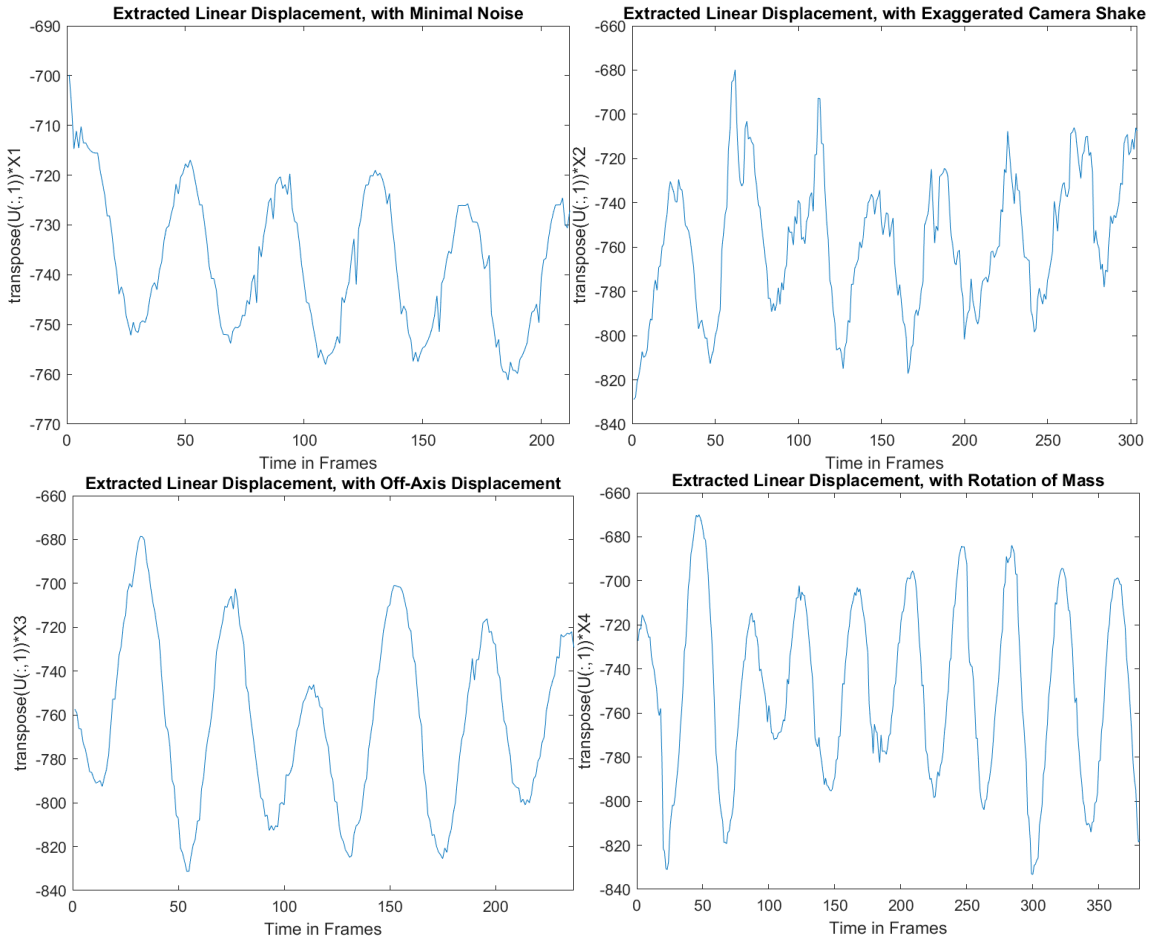

Figure 3: Oscillation of the position of the bucket as extracted via color tracking and SVD.

5 Summary and Conclusions

Here, we have a clear example of the power of data and its proper analysis. The SVD has allowed us to extract a linear oscillation from video of a bucket that was swinging, rotating, and blurred. It is interesting to think of using this sort of analysis on systems that are not as well-described by physical laws, in order to reveal these systems' inner workings.

References

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

[2] Weisstein, Eric W. "L^2-Norm." From *MathWorld*--A Wolfram Web Resource. https://mathworld.wolfram.com/L2-Norm.html

Appendix A MATLAB Functions

- trackPoints = trackByColor(frames,start,duration,colorMask) Tracks the median location of the colors specified by the given colorMask function.

- mask = colorMask(RGB) an automatically generated function that returns an array with entries for each pixel in the image RGB. If the entry for a pixel is 1, then that pixel's color matched the colors that this function is searching for.
- [U, ~, ~] = svd(A) returns the matrix U from the SVD decomposition of A.

Appendix B MATLAB Code

main.m

```
%{
02-20-2021
Shane Fretwell
AMATH 482 Assignment 3, Spring-Mass Systems
%}
%%
clear; close all;

%%
GRAPHICS = false;
w = 640;
h = 480;

%%
load('C:/Users/sear/Documents/AMATH482_data/Ass3/cam1_1.mat')
load('C:/Users/sear/Documents/AMATH482_data/Ass3/cam2_1.mat')
load('C:/Users/sear/Documents/AMATH482_data/Ass3/cam3_1.mat')

%%
if GRAPHICS
    implay(vidFrames1_1)
end
%%
if GRAPHICS
    implay(vidFrames2_1)
end
%%
if GRAPHICS
    implay(vidFrames3_1)
end

%% Show color masked video
if GRAPHICS
    figure(2)
    for k=1:T3
    [BW,maskedRGBImage] = cam3_1Mask(vidFrames3_1(:,:,:,k));
    imshow(maskedRGBImage);
    end
end

%% Tracking cam1_1 by color
T = 212;
track1_1 = trackByColor(vidFrames1_1,1,T,@cam1_1Mask);
```

```
%% Tracking cam2_1 by color
start = 10;
track2_1 = trackByColor(vidFrames2_1,start,T,@cam2_1Mask);

%% Tracking cam3_1 by color
start = 20;
track3_1 = trackByColor(vidFrames3_1,start,T,@cam3_1Mask);

%% Plotting tracked points
if GRAPHICS
    figure(1)
    for k=1:T
        scatter(track1_1(k, 1), track1_1(k, 2), 25);
        axis([0 w 0 h]);
        hold on;
        scatter(track2_1(k, 1), track2_1(k, 2), 25);
        scatter(track3_1(k, 1), track3_1(k, 2), 25);
        hold off;

        drawnow
    end
end

%% Move on to Case 2
clear vidFrames1_1 vidFrames2_1 vidFrames3_1
load('C:/Users/sear/Documents/AMATH482_data/Ass3/cam1_2.mat')
load('C:/Users/sear/Documents/AMATH482_data/Ass3/cam2_2.mat')
load('C:/Users/sear/Documents/AMATH482_data/Ass3/cam3_2.mat')

%%
if GRAPHICS
    implay(vidFrames1_2)
end
%%
if GRAPHICS
    implay(vidFrames2_2)
end
%%
if GRAPHICS
    implay(vidFrames3_2)
end

%% Show color masked video
if GRAPHICS
    figure(5)
    T = length(vidFrames3_2(1,1,1,:));
    for k=1:T
        [BW,maskedRGBImage] = cam3_2Mask(vidFrames3_2(:,:,:,k));
        imshow(maskedRGBImage);
    end
end
%% Tracking cam1_2 by color
start = 10;
T = length(vidFrames1_2(1,1,1,:)) - start;
track1_2 = trackByColor(vidFrames1_2,1,T,@cam1_1Mask);
```

```
%% Tracking cam2_2 by color
start = 1;
track2_2 = trackByColor(vidFrames2_2,start,T,@cam2_1Mask);

%% Tracking cam3_2 by color
start = 15;
track3_2 = trackByColor(vidFrames3_2,start,T,@cam3_2Mask);
%% Plotting tracked points
if GRAPHICS
    figure(1)
    for k=1:T
        scatter(track1_2(k, 1), track1_2(k, 2), 25);
        axis([0 w 0 h]);
        hold on;
        scatter(track2_2(k, 1), track2_2(k, 2), 25);
        scatter(track3_2(k, 1), track3_2(k, 2), 25);
        hold off;
        title(string(k))
        drawnow
        pause(0.2)
    end
end
%% Move on to Case 3
clear vidFrames1_2 vidFrames2_2 vidFrames3_2
load('C:/Users/sear/Documents/AMATH482_data/Ass3/cam1_3.mat')
load('C:/Users/sear/Documents/AMATH482_data/Ass3/cam2_3.mat')
load('C:/Users/sear/Documents/AMATH482_data/Ass3/cam3_3.mat')

%%
if GRAPHICS
    implay(vidFrames1_3)
end
%%
if GRAPHICS
    implay(vidFrames2_3)
end
%%
if GRAPHICS
    implay(vidFrames3_3)
end

%% Show color masked video
if GRAPHICS
    figure(5)
    T = length(vidFrames3_3(1,1,1,:));
    for k=1:T
        [BW,maskedRGBImage] = cam3_2Mask(vidFrames3_3(:,:,:,k));
        imshow(maskedRGBImage);
    end
end
%% Tracking cam1_3 by color
start = 3;
T3 = length(vidFrames3_3(1,1,1,:));
track1_3 = trackByColor(vidFrames1_3,start,T3,@cam1_1Mask);
```

```
%% Tracking cam2_3 by color
start = 32;
track2_3 = trackByColor(vidFrames2_3,start,T3,@cam2_1Mask);

%% Tracking cam3_3 by color
start = 1;
track3_3 = trackByColor(vidFrames3_3,start,T3,@cam3_2Mask);

%% Plotting tracked points
if GRAPHICS
    figure(1)
    for k=1:T
        scatter(track1_3(k, 1), track1_3(k, 2), 25);
        axis([0 w 0 h]);
        hold on;
        scatter(track2_3(k, 1), track2_3(k, 2), 25);
        scatter(track3_3(k, 1), track3_3(k, 2), 25);
        hold off;
        title(string(k))
        drawnow
        pause(0.2)
    end
end

%% Move on to Case 4
clear vidFrames1_3 vidFrames2_3 vidFrames3_3
load('C:/Users/sear/Documents/AMATH482_data/Ass3/cam1_4.mat')
T1 = length(vidFrames1_4(1,1,1,:));

load('C:/Users/sear/Documents/AMATH482_data/Ass3/cam2_4.mat')
T2 = length(vidFrames2_4(1,1,1,:));

load('C:/Users/sear/Documents/AMATH482_data/Ass3/cam3_4.mat')
T3 = length(vidFrames3_4(1,1,1,:));

%%
if GRAPHICS
    implay(vidFrames1_4)
end
%%
if GRAPHICS
    implay(vidFrames2_4)
end
%%
if GRAPHICS
    implay(vidFrames3_4)
end

%% Show color masked video
if GRAPHICS
    figure(5)
    T = length(vidFrames2_4(1,1,1,:));
    for k=1:T
        [BW,maskedRGBImage] = cam2_1Mask(vidFrames2_4(:,:,:,k));
        imshow(maskedRGBImage);
```

```
    end
end
%% Tracking cam1_4 by color
start = 7;
T = T1 - start;
track1_4 = trackByColor(vidFrames1_4,start,T,@cam1_1Mask);

%% Tracking cam2_4 by color
start = 15;
track2_4 = trackByColor(vidFrames2_4,start,T,@cam2_1Mask);

%% Tracking cam3_4 by color

start = 7;
track3_4 = trackByColor(vidFrames3_4,start,T,@cam3_2Mask);

%% Plotting tracked points
if GRAPHICS
    figure(1)
    for k=1:T
        scatter(track1_4(k, 1), track1_4(k, 2), 25);
        axis([0 w 0 h]);
        hold on;
        scatter(track2_4(k, 1), track2_4(k, 2), 25);
        scatter(track3_4(k, 1), track3_4(k, 2), 25);
        hold off;
        title(string(k))
        drawnow
        pause(0.1)
    end
end

%%
clear vidFrames1_4 vidFrames2_4 vidFrames3_4
```

projectedSpacePlots.m

```
%% SVD on tracked movement

%% Case 1

X1 = double([
    transpose(track1_1);
    transpose(track2_1);
    transpose(track3_1)
]);

%%
[U,~,~] = svd(X1, 'econ');
displacement = transpose(U(:, 1))*X1;

%% Plotting the tracked points projected onto the 1-dimensional basis formed by the
main component

figure(1)
plot(displacement);
title("Extracted Linear Displacement, with Minimal Noise");
xlabel("Time in Frames");
ylabel("transpose(U(:,1))*X1");
axis([0 length(displacement) -inf inf]);
axis 'auto y'

%% Case 2

X2 = double([
    transpose(track1_2);
    transpose(track2_2);
    transpose(track3_2)
]);

%%
[U,~,~] = svd(X2, 'econ');
displacement = transpose(U(:, 1))*X2;

%% Plotting the tracked points projected onto the 1-dimensional basis formed by the
main component

figure(2)
plot(displacement);
title("Extracted Linear Displacement, with Exaggerated Camera Shake");
xlabel("Time in Frames");
ylabel("transpose(U(:,1))*X2");
axis([0 length(displacement) -inf inf]);
axis 'auto y'

%% Case 3

X3 = double([
    transpose(track1_3);
    transpose(track2_3);
```

```
    transpose(track3_3)
]);

%%
[U,~,~] = svd(X3, 'econ');
displacement = transpose(U(:, 1))*X3;

%% Plotting the tracked points projected onto the 1-dimensional basis formed by the
main component

figure(3)
plot(displacement);
title("Extracted Linear Displacement, with Off-Axis Displacement");
xlabel("Time in Frames");
ylabel("transpose(U(:,1))*X3");
axis([0 length(displacement) -inf inf]);
axis 'auto y'

%% Case 4

X4 = double([
    transpose(track1_4);
    transpose(track2_4);
    transpose(track3_4)
]);

%%
[U,~,~] = svd(X4, 'econ');
displacement = transpose(U(:, 1))*X4;
% Remove data from four frames that did not show a sufficient amount of
% yellow paint on the bucket for tracking the bucket's position.
displacement = rmoutliers(displacement);

%% Plotting the tracked points projected onto the 1-dimensional basis formed by the
main component

figure(4)
plot(displacement);
title("Extracted Linear Displacement, with Rotation of Mass");
xlabel("Time in Frames");
ylabel("transpose(U(:,1))*X4");
axis([0 length(displacement) -inf inf]);
axis 'auto y'
```