AMATH 482 Homework 3

Maxime Didascalou

February 24, 2020

Abstract

The aim of this project was to get spacial information that can be easily interpreted out of videos of an oscillating object. After finding the coordinates of the object from several different cameras, Principle component analysis is used to extract the different dimensions of movement. The results are relatively conclusive, however factors such as noise or errors in tracking add a lot of uncertainty.

1 Introduction and Overview

In most cases, data coming from the real world will not be easily interpretable and thus we have to go through some steps to try and understand what is hidden in it. One thing that you can do is principal component analysis. It is a tool which allows us to only look at certain components of the data. In this case we are given videos of 4 oscillating objects (3 videos per case). Since a video is in 2 dimensions, we cannot extract the full movement of an object with only one. And if we have the coordinates of all three videos, it is hard to understand what is going on without doing anything to these coordinates. So by doing principle component analysis we can take these coordinates and arrange them in a way that makes it easier to understand the movement of the object.
I will first go through the theoretical background necessary to understand why principal component analysis works, and then I will go through the algorithms I implemented, and explain the results.

2 Theoretical Background

As we learned from our textbook [1], using the singular value decomposition (SVD) it is possible to write any matrix $\mathbf{A}$ as:

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

This comes from the fact that you can write

$$\mathbf{A}\mathbf{V} = \mathbf{U}\mathbf{\Sigma} \tag{2}$$

Where $\mathbf{U}$ is a matrix containing all the principle semiaxes (scaled to have a length of 1) that would appear if we applied $\mathbf{A}$ to the unit circle, $\mathbf{\Sigma}$ contains all the lengths of these semiaxes (from biggest to smallest), and $\mathbf{V}$ contains the vectors (from the unit circle) that give these principle semiaxes when multiplied by $\mathbf{A}$. This makes it so that $\mathbf{V}$ and $\mathbf{U}$ are both orthonormal (so their inverse equals their transpose) and $\mathbf{\Sigma}$ is a diagonal matrix where each of the scaling factors are arranged in decreasing order (they are called the singular values). Now consider a matrix:

$$\mathbf{X} = \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} \tag{3}$$

Then, using the fact that sample variance and covariance of samples with mean 0 are defined as:

$$\sigma_a^2 = \frac{1}{n-1}\mathbf{a}\mathbf{a}^T, \quad \sigma_{ab}^2 = \frac{1}{n-1}\mathbf{a}\mathbf{b}^T, \tag{4}$$

it is easy to show (assuming the mean of each row is 0) that:

$$\frac{1}{n-1}\mathbf{X}\mathbf{X}^T = \mathbf{C}_\mathbf{X}, \tag{5}$$

Where $\mathbf{C}_\mathbf{X}$ is the covariance matrix of the rows of $\mathbf{X}$, and n is the length of the rows.
Now we want a change of basis so that there is no covariance between the rows. Therefore we diagonalize the covariance matrix, so that all off-diagonal elements are zero, and we get:

$$\mathbf{C}_\mathbf{X} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1} \tag{6}$$

The eigenvectors in $\mathbf{V}$ are the principle components, and they are uncorrelated since $\mathbf{C}_\mathbf{X}$ is symmetric. The entries of $\mathbf{\Lambda}$ are the variances of these new variables. To connect this to the SVD we simply say

$$\mathbf{A} = \frac{1}{\sqrt{n-1}}\mathbf{X} \Rightarrow \mathbf{C}_\mathbf{X} = \mathbf{A}\mathbf{A}^T = (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^*)(\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T) = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^T \tag{7}$$

Note that we first need to set the mean of each row of $\mathbf{A}$ to 0.
Then to work in the basis of the principle components we just need to multiply by $\mathbf{U}^T$:

$$\mathbf{Y} = \mathbf{U}^T\mathbf{X} \tag{8}$$

This process is called principal component analysis (PCA). It is useful because since there is no covariance between the variables, you can choose to only look at the ones with the most variance since they are the ones that contain the most information, and you could possibly plot data that would otherwise be impossible to plot due to it's high dimensions.

3 Algorithm Implementation and Development

These are the main algorithms I used in this project.

1. Loading the data and constructing a matrix where the rows are the coordinates of the matrix from the three cameras (6 rows since we take x and y coordinates)
2. Using SVD/PCA to extract the principle components and analyse the different components of the movement of the object.

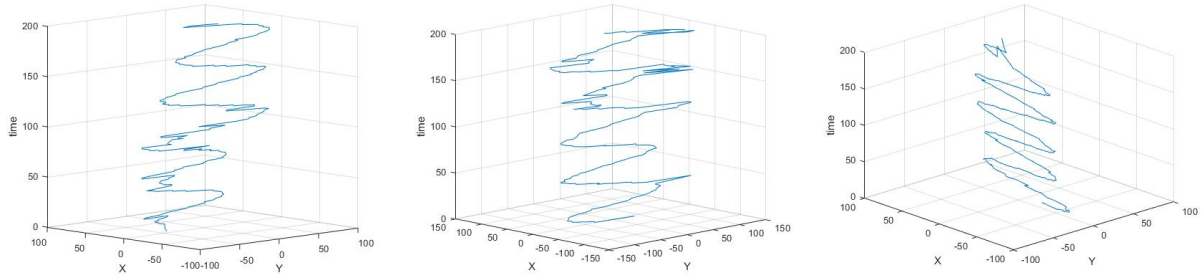
Algorithm 1: Finding Coordinates

Import videos to MATLAB
Cut them so the object starts at the same position in all of them and so they have the same number of frames n
for each of the three videos **do**
  for $j = 1 : n$ **do**
    Convert frame j to grayscale
    Find the index (x,y) of the maximum since the object has a bright light on it (in a cropped area of the frame for greater accuracy)
  end for
end for

Note: In Cartesian coordinates, low values of y are at the bottom, but in this case I found indexes in a matrix, so a higher value of y means that it's lower in the picture. Values of x behave like usual.

Algorithm 2: Finding principal components

Put all coordinates in a 6 by n matrix $\mathbf{A}$
Subtract each row of $\mathbf{A}$ by its mean
Find the SVD decomposition of $\mathbf{A}$ using `svd`
Compute $\mathbf{Y} = \mathbf{U}^T\mathbf{X}$



(a) Video 1 (b) Video 2 (c) Video 3

Figure 1: Non-transformed coordinates for the first test

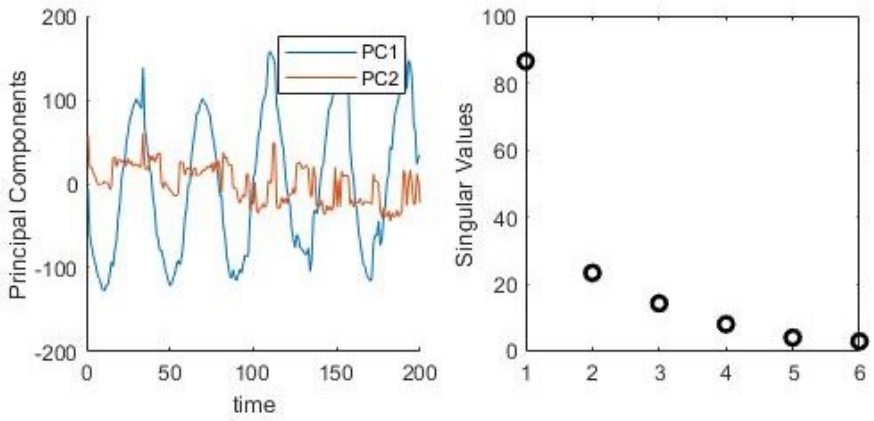


Figure 2: Important principal components (left) based on singular values (right)

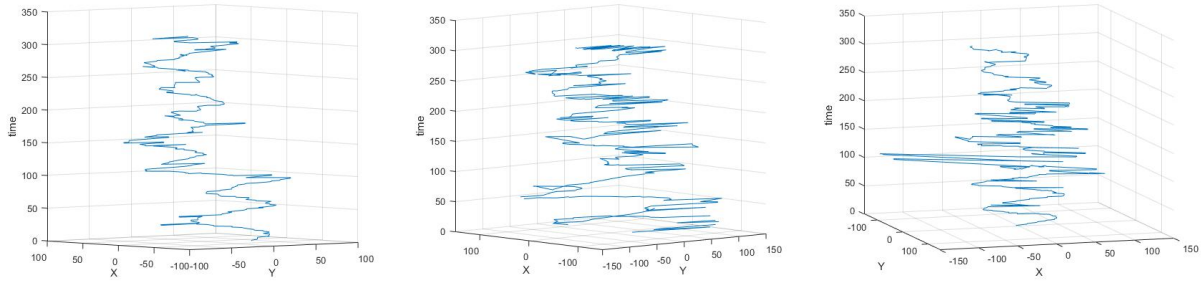
4 Computational Results

4.1 Test 1: Ideal Case

In this case, we know that the object was only moving up and down, as you can see from Figure 1. So theoretically, we should only get one non zero singular value since the movement is only one dimensional. In practice, we see in Figure 2 that one singular value dominates the rest, but they are not zero, probably due to noise in the data or errors in the tracking. But the principal components are what we expect: oscillatory motion for PC1 and the rest don't have useful information.

4.2 Test 2: Noisy Case

This case also only has one dimensional motion, but it is however mixed with noise as you can see from Figure 2. As you can see in Figure 4, the first principal component still manages to somewhat capture the oscillatory motion of the object, but the singular values 2 through 6 are closer to singular value 1 than in Test 1. So PCA was not able to completely identify that the movement was one dimensional.



(a) Video 1 (b) Video 2 (c) Video 3

Figure 3: Non-transformed coordinates for the second test

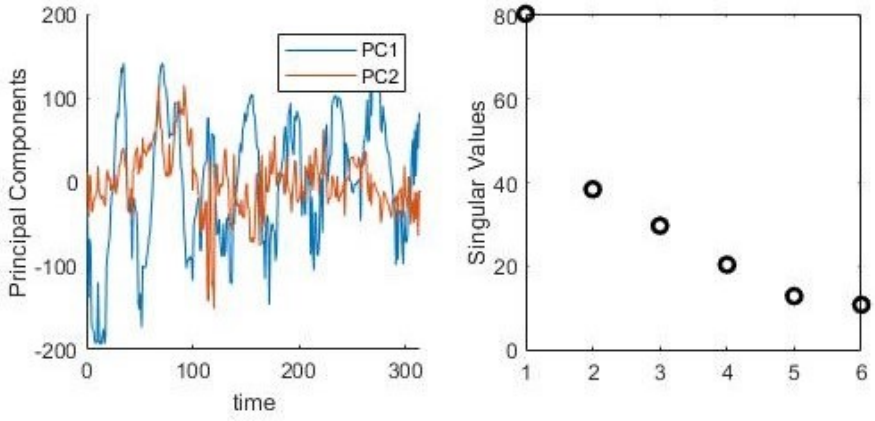
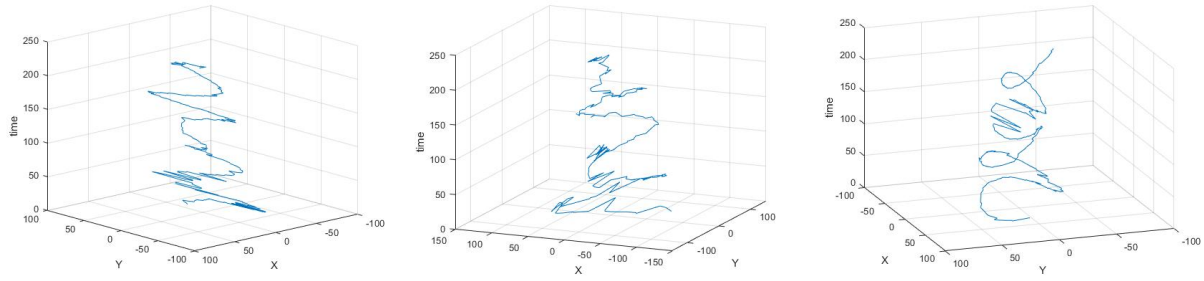


Figure 4: Important principal components (left) based on singular values (right)

4.3 Test 3: Horizontal Displacement

In this case, the object is released off centre with movement in the x-y plane (as can be seen in Figure 5) so there should be more than one principal component. Indeed we can see that PC1 and PC2 in Figure 6 both show oscillatory motion, capturing the 2 dimensional movement of the object. However one caveat is that the third singular value is still relatively big compared to the first 2. This is probably due again to small errors in the tracking of the object in the video, or the object might have moving slightly in the third dimension.



(a) Video 1 (b) Video 2 (c) Video 3

Figure 5: Non-transformed coordinates for the third test

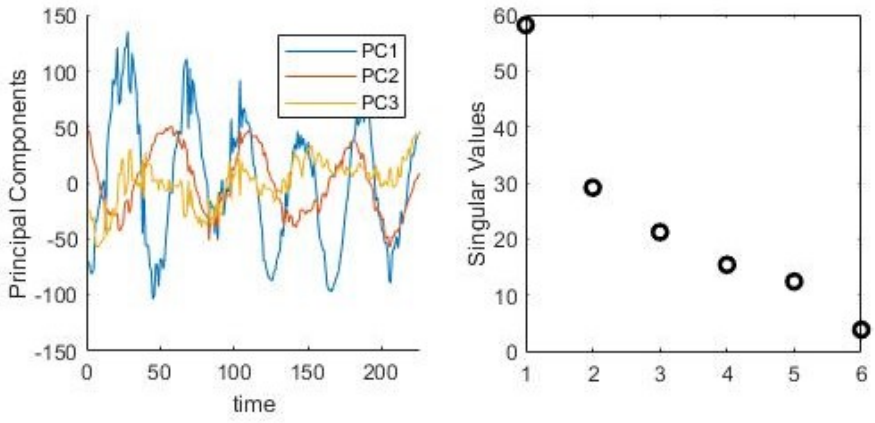
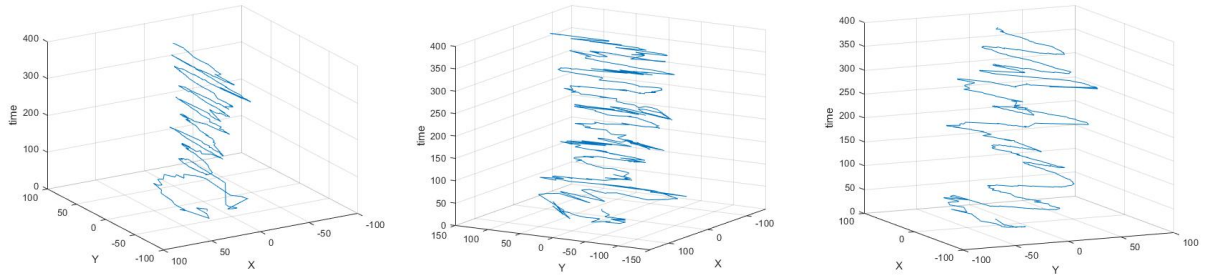


Figure 6: Important principal components (left) based on singular values (right)

4.4 Test 4: Horizontal Displacement and Rotation

This case had the object moving up and down like the others, but also slightly rotating in the x-y plane. Theoretically, this should mean that there would be 3 significant principal components. In practice, from Figure 8 we get that the first principal component is much bigger than the two following it. This is probably due to the fact that the original coordinates from the tracking algorithm (Figure 7), seem to mainly describe one dimensional motion except for a time close to 0 where we can see some rotation. So the vertical motion is much bigger than the horizontal one. But when we look at the principal components, the initial rotation close to time = 0 seems to be captured by PC1 and PC2 since they both have a spike near that point.



(a) Video 1 (b) Video 2 (c) Video 3

Figure 7: Non-transformed coordinates for the fourth test

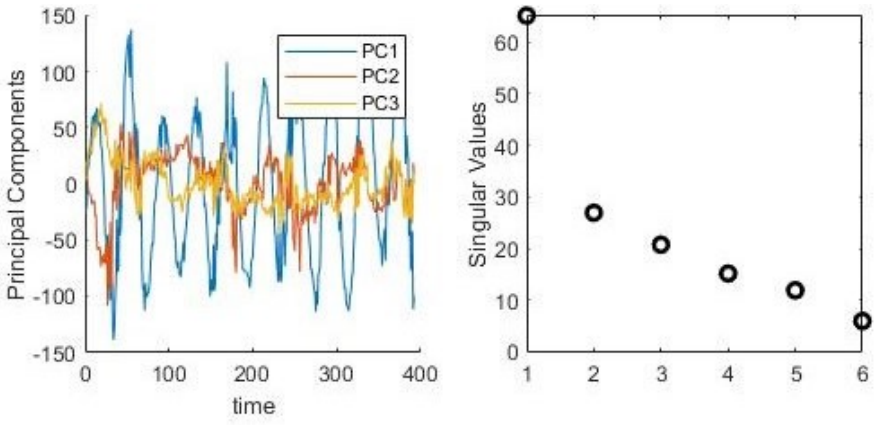


Figure 8: Important principal components (left) based on singular values (right)

5 Summary and Conclusions

After finding the coordinates of the location of an object for each frame of three videos showing the movement of the same object, I was able to perform principal component analysis on these coordinates in order to identify the different components of the objects movement.
The results mostly indicate what we would expect from each case. It is also important to note that there is a lot of uncertainty in the coordinates since the light on top of the object that was used to locate it in the frame sometimes wasn't the lightest point. So that influenced the results, but it is still possible to understand the main components of the movement since PCA allows us to get rid of some of those uncertainties. PCA starts to fall apart however when a lot of noise is added in the data. Although test 2 showed some signs of oscillation in PC1, the singular values were too close to each other to be able to conclusively say that the movement was only 1-dimensional.

References

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

Appendix A MATLAB Functions

- `I = rgb2gray(RGB)` converts the truecolor image `RGB` to the grayscale image `I`.
- `[M,I] = max(A,[],_,'linear')` returns the linear index into `A` that corresponds to the maximum value in `A`.
- `[I1,I2,...,In] = ind2sub(sz,ind)` returns `n` arrays `I1,I2,...,In` containing the equivalent multidimensional subscripts corresponding to the linear indices `ind` for a multidimensional array of size `sz`. Here `sz` is a vector with `n` elements that specifies the size of each array dimension.
- `[U,S,V] = svd(A,'econ')` performs a singular value decomposition of matrix `A`, such that `A = U*S*V'` and returns the reduced forms of `U`, `S` and `V`.

Appendix B MATLAB Code

```
% -- Load Movies
load cam1_2.mat;
load cam2_2.mat;
load cam3_2.mat;
vidFrames2_2 = vidFrames2_2(:,:,:,27:end);
vidFrames3_2 = vidFrames3_2(:,:,:,5:end);
[height1 width1 rgb1 num_frames1] = size(vidFrames1_2);
[height2 width2 rgb2 num_frames2] = size(vidFrames2_2);
[height3 width3 rgb3 num_frames3] = size(vidFrames3_2);
num_frames = min([num_frames1 num_frames2 num_frames3]);

% -- Watch Movie 1
for j=1:num_frames
  X=vidFrames1_2(:,:,:,j);
  X=rgb2gray(X);
  imshow(X); drawnow
end
```

Listing 1: Loading data and watching movies

```
% -- Get coordinates for 1st vid
coords1 = zeros([2 num_frames]);
for j=1:num_frames
    X=rgb2gray(vidFrames1_2(:,:,:,j));
    X=im2double(X);
    X_temp=X(192:end,300:500);
    [M,I] = max(X_temp(:));
    dims = size(X_temp)
    [y,x] = ind2sub([dims(1) dims(2)],I);
    coords1(:,j) = [x y];
end
% -- Plotting coordinates
coords1(1,:) = coords1(1,:) - mean(coords1(1,:));
coords1(2,:) = coords1(2,:) - mean(coords1(2,:));
figure(1)
plot(1:num_frames,coords1(2,:))
figure(2)
plot3(coords1(2,:),coords1(1,:),1:num_frames)
xlim([-100,100])
ylim([-100,100])
xlabel('Y')
ylabel('X')
zlabel('time')
grid on
```

Listing 2: Finding coordinates and plotting them

```
A = zeros([6 num_frames]);
A(1:2,:) = coords1;
A(3:4,:) = coords2;
A(5:6,:) = coords3;

[U,S,V]=svd(A/sqrt(num_frames-1),'econ'); % perform the SVD
Y=U'*A; % produce the principal components projection

subplot(1,2,1)
hold on
plot(1:num_frames,Y(1,:))
plot(1:num_frames,Y(2,:))
xlabel('time')
ylabel('Principal Components')
legend('PC1','PC2')
pbaspect([1 1 1])
subplot(1,2,2)
plot(1:6,diag(S),'ko','Linewidth', 2)
ylabel('Singular Values')
xlim([1,6])
pbaspect([1 1 1])
```

Listing 3: Using SVD to find principle components, plotting.