AMATH 482 Homework 3

lewisxy

February 23, 2021

Abstract

In this paper, we demonstrated the possibility of analyzing physical system from multiple video recordings with Principle Component Analysis (PCA).

1 Introduction and Overview

In this paper, we explore the possibly of analyzing well known physics system using multiple camera videos and Principle Component Analysis. By recording the physics system at different angle and tracking the object with in each video. We can "over" sample the motion of system as each camera provides 2 dimensions (x and y coordinates of pixels). With these data, we can run PCA and extract the principle components. By analyze the distribution of singular values, we can deduce the degree of freedom of the system even without any knowledge of the underlying system.

2 Theoretical Background

2.1 Eigenvalue Decomposition (Diagonalization)

Eigenvalue Decomposition (a.k.a. Diagonalization) is a decomposition for square matrices. For a $n \times n$ square matrix A. It can be written as

$$A = PDP^{-1} \tag{1}$$

where columns of P are eigenvectors of A and D is a diagonal matrix containing eigenvalues of the corresponding eigenvectors in P. Both P and D are $n \times n$ matrix. Note that not all square matrices A can be diagonalized. A is diagonalizable if and only if the sum of dimensions of eigenspaces is equal to n (this is automatically true for matrices with n distinct eigenvalues).

2.2 Singular Value Decompositon (SVD)

Singular Value Decomposition is one of the tools to compute PCA. For any $m \times n$ matrix A. We can write

$$A = U\Sigma V^T \tag{2}$$

where U is a $m \times m$ orthonormal matrix, Σ is a $m \times n$ diagonal with non-negative entries, V is a $n \times n$ orthonormal matrix. It worth mention that A can be any matrix in this case, and we can approximate matrix A with a rank k matrix A_k as

$$A \approx A_k = \sum_{i=1}^{k} \sigma_k U_k V_k^T \tag{3}$$

where U_k and V_k^T are the kth column of U and V^T. S_k is the kth singular value.

We can compute SVD using diagonalization. It's trivial to show that $A^TA = V\Sigma^2V^T$ and $AA^T = U\Sigma^2U^T$. Since A^TA and AA^T are symmetric square matrices, we can use diagonalization to solve U and V.

2.3 Principle Component Analysis (PCA)

The core mathematical tool in our methods is Principle Component Analysis (PCA). At a high level, this method allows us to reduce the dimension of data with minimum information loss (measured in L2 norm). To perform PCA for a $d \times n$ data matrix X, we first need to subtract the mean of the data for each dimension. Then, we compute the covariance matrix $\frac{1}{n-1}XX^T$ where its ith row and jth column representing the covariance between dimension i and dimension j (the covariance of a dimension with itself is the variance). We then diagonalize the covariance matrix to compute the eigenvalues and eigenvectors representing the principle components and their corresponding magnitude.

$$\frac{1}{n-1}XX^T = PDP^{-1} \tag{4}$$

Alternatively, we can also use SVD. To see the connection, consider $A = \frac{1}{\sqrt{n-1}}X$. Then $AA^T = \frac{1}{n-1}XX^T = U\Sigma^2U^T$. We can work with data in the orthogonal space with change of basis. Let $Y = U^TX$. Y is the resulted data represented using eigen basis.

2.4 Mass Spring System

In physics, a mass spring system is a one dimension dynamics. In an ideal case, the displacement of the spring over time can be represented by equation (**??**). This can be solved from the differential equation formulated using Newton's second law and Hooke's law (6).

$$x(t) = A\cos(\omega t + \phi) \tag{5}$$

$$F = ma, \qquad F = -kx, \qquad m\frac{d^2x(t)}{dt^2} = -kx(t) \tag{6}$$

2.5 Pattern Recognition

One of the engineering challenge is to extract the position of the object from the video. In this paper, we use a simple brute-force based pattern matching algorithm inspired by convolution and cosine similarity. We first extract an image of the object manually from one of the video frame. Using this image as filter, we compute the convolution of this filter to the image. Then normalize the value by the L2 norm of the filter and corresponding part of the image. Conceptually, if we treat the filter and the corresponding part of the image as unit vectors A, B, the dot product $A \cdot B = cos(\theta)$ where θ is the angle between A and B. This is a nice metric to measure how close is the corresponding part of the image to the filter. After this step, we simply find the maximum position in the convolution, which should give us the center of the object. Of course, when the transformation of the object involves rotation and scaling, the effectiveness of this method is limited. Mathematically, it can be described as the following.

Let B be a $h \times w$ matrix representing the image of object we are look for. A be a $H \times W$ image that we want to find the object ($H > h, W > w$), $A_{i,j}$ be the sub-image of size $h \times w$ started with point (i, j) (typically means top left corner) ($i < H - h, j < W - w$). Define matrix dot product $C \cdot D = \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij}D_{ij}$ for $m \times n$ matrices C, D. (Note: here C_{ij} means the value in ith row and jth column, not to confuse with previous definition of $A_{i,j}$).

$$\text{position} = \arg\max_{(i,j)} \frac{A_{i,j} \cdot B}{||A_{i,j}|| \cdot ||B||}, \quad i < H - h, j < W - w \tag{7}$$

3 Algorithm Implementation and Development

In this section, we describes the different steps to analyze the data from the cameras. Implementations are provided in the form of MATLAB code in Appendix B, explanations of selected MATLAB functions are provided in Appendix A.

3.1 Preparing Data

The firs step is to extract the position of the obeject from the video. Since the object does not change shapes throughout the video, we used a pattern recognition algorithm to extract the position by first manually extracts an image of the object. The detail of this algorithm is described in previous sections. The algorithm itself is implemented in MATLAB code (`convMatch`).

3.2 PCA

After the extraction, we bundle the 2 dimension data from 3 cameras to a 6-dimension data set. Then, as described in previous sections, we first subtract the mean from the data. Then using SVD, we compute the principle components of the data. We compute the low rank approximations of the data using equation (3), and plot the projection to first principle components using the change of basis technique mentioned above.

4 Computational Results

We used the above mentioned method to extract the 4 sets of data from 3 cameras. Set 1 is the baseline data. Set 2 is similar to set 1 but with noticeable noise introduced in the form of camera shake. Set 3 does not have camera shakes, but the mass also swings (like pendulum) in addition to simple harmonic motion. Set 4 is similar to set 1, but also added rotations. Using the above mentioned analysis method, we presented 4 graphs for each set of data. They represent original data, data in first 2 principle components, rank 1 and rank 2 approximations. Unfortunately, the rotation involved in set 4 made it very difficult for the pattern recognition algorithm to extract the position of the mass from the video. Due to limited time, the analysis result of set 4 is not shown in this paper.

In the first set, without noise, we can separate 2 principle components that is significantly larger than the rest. This can also be seen from the obvious circle shape in figure 1b. In rank 1 approximation, we can already observe the sinusoidal shape of the data. With rank 2 approximation, we can observe the phase between these data as well, and the graph looks very similar to the original data. Of course, if we align the video of these 3 camera, we can probably approximate the data well even with 1 principle component, that is, with 1 singular value significantly larger than all the rest.

In the second set, the introduction the noise make the data look a lot messy. This reflected on the first 2 principle components being less significant. Despite the noise, comparing the rank 1 and rank 2 approximations to the original data, the reconstructed data still represents the original data fairly accurately.

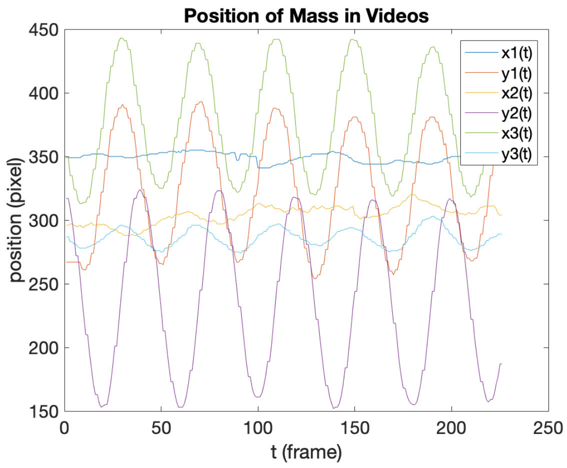
In the third set, due to pendulum motion, both x and y axis resembles sinusoids. We can also see this pattern from the projection to first 2 principle components in figure 3b. There is few data points that are due to a measurement error, and we can observe this from the figure 3b (in the bottom) as well.

5 Summary and Conclusions

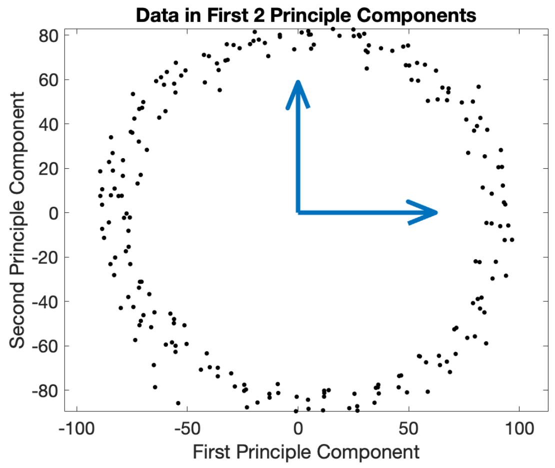
In this paper, we demonstrated the possibility of analyzing physics system with cameras and PCA. Video recording is an excellent way of sampling data as it's essentially projecting a complex system to 2 dimensional screen. With multiple camera recording at different directions, we are able to gather enough information about the system. Using PCA, we can reduce the dimension of the over sampled data from videos, and analyze the underlying system without specific knowledge.

Appendix A MATLAB Functions

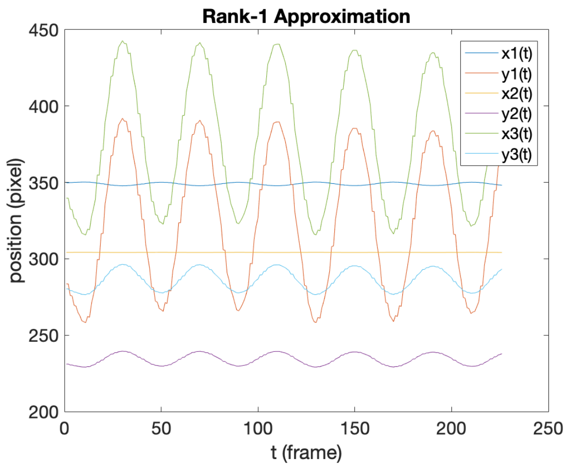
- `convMatch(src, pattern)`: user defined function to match `pattern` image in `src` image.
- `svd(m)`: perform SVD on the given matrix, return `U`, `S`, `V` of the decomposed matrix.



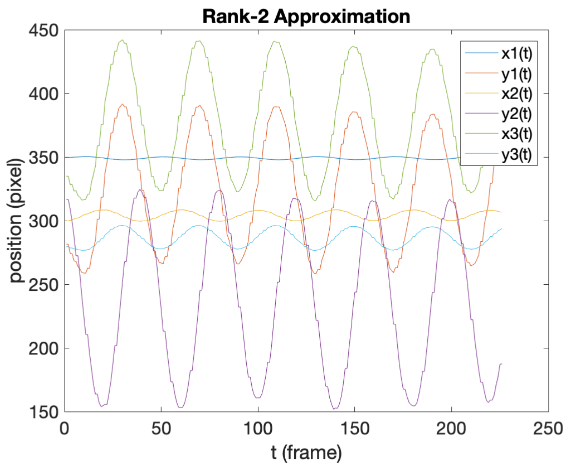
(a) Raw Data for Set 1



(b) Data represented in First 2 priciple components

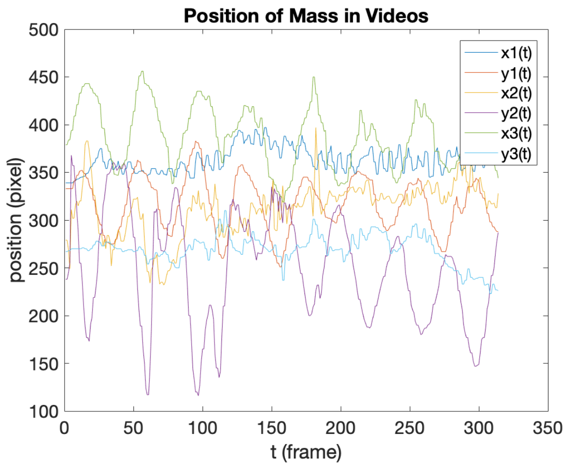


(c) Rank 1 Approximation of the data

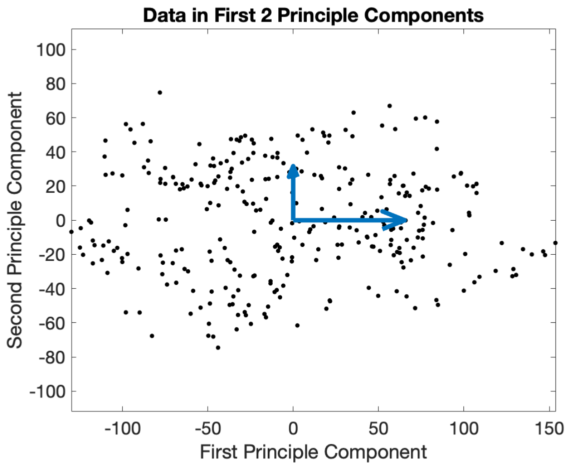


(d) Rank 2 Approximation of the data

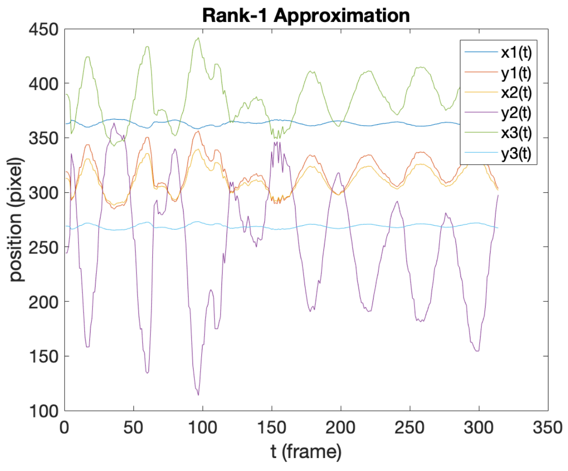
Figure 1: Analysis of Set 1



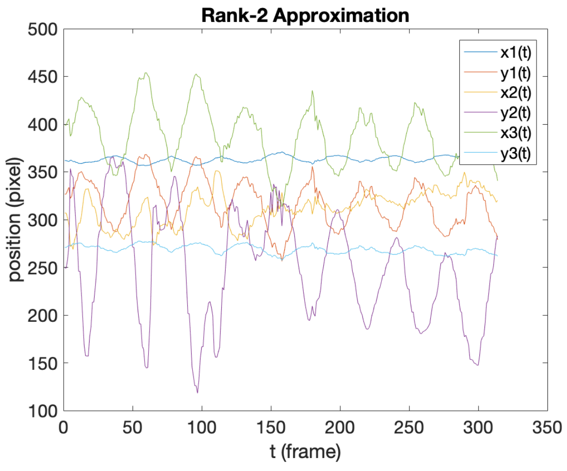
(a) Raw Data for Set 2



(b) Data represented in First 2 priciple components

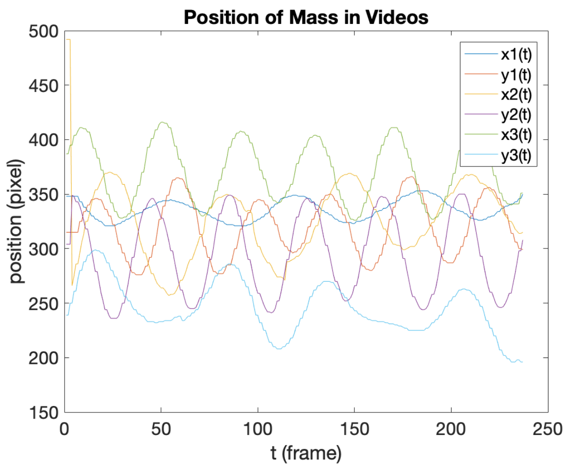


(c) Rank 1 Approximation of the data

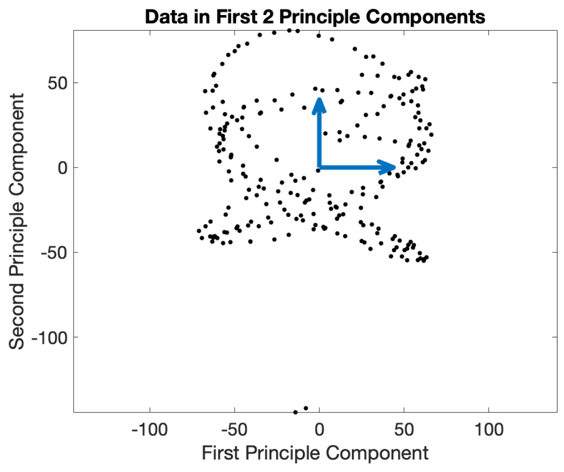


(d) Rank 2 Approximation of the data

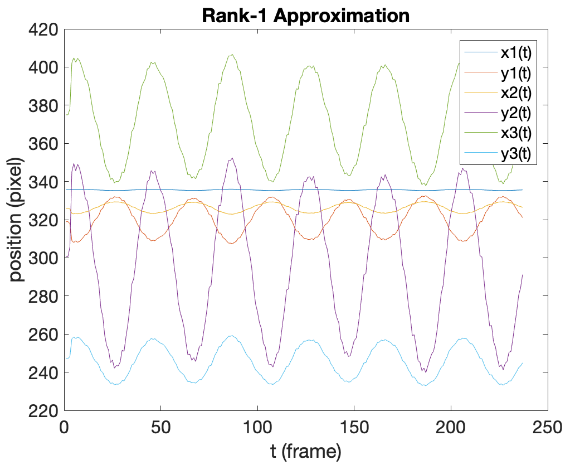
Figure 2: Analysis of Set 2



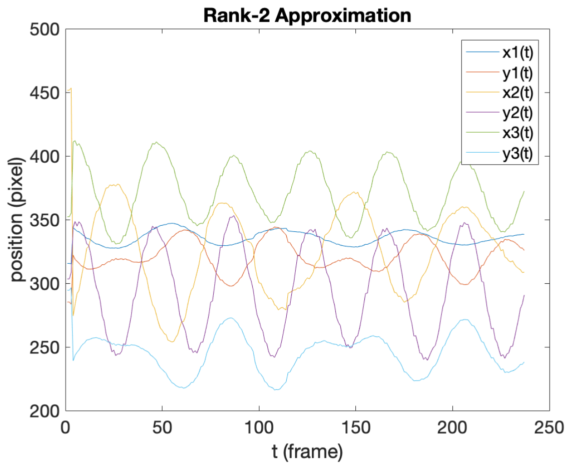
(a) Raw Data for Set 3



(b) Data represented in First 2 priciple components



(c) Rank 1 Approximation of the data



(d) Rank 2 Approximation of the data

Figure 3: Analysis of Set 3

Appendix B MATLAB Code

```
function [ dst_img ] = convMatch(src_img, pattern_img)
%CONVMATCH Summary of this function goes here
%   src_img is expected to be larger than or equal to pattern_img in both
%   dimension
    src_img = double(src_img) / 255;
    pattern_img = double(pattern_img) / 255;
    offset_i = round(size(pattern_img, 1) / 2);
    offset_j = round(size(pattern_img, 2) / 2);
    pattern_normalized = pattern_img / norm(pattern_img(:));
    dst_img = zeros(size(src_img));
    for i=1:(size(src_img, 1) - size(pattern_img, 1))
        for j=1:(size(src_img, 2) - size(pattern_img, 2))
            tmp1 = src_img(i:i+size(pattern_img,1)-1, j:j+size(pattern_img,2)-1);
            tmp1 = tmp1 / norm(tmp1(:));
            dst_img(i+offset_i, j+offset_j) = sum(tmp1(:) .* pattern_normalized(:));
        end
    end
end
```

Listing 1: Algorithm for matching object in image

```
%% Clean workspace
clear all; close all; clc


%% Load Data and Constants
% load one data at a time
%load('data.nosync/cam_1.mat');
%load('data.nosync/cam_2.mat');
load('data.nosync/cam_3.mat');

video_height = 480;
video_width = 640;

data_raw = data;

%% Plot data
data_label = ['x1(t)'; 'y1(t)'; 'x2(t)'; 'y2(t)'; 'x3(t)'; 'y3(t)'];
figure
for i=1:6
    plot(data_raw(i, :));
    hold on;
end
legend(data_label);
set(gca, 'Fontsize',16);
title('Position of Mass in Videos');
xlabel('t (frame)');
ylabel('position (pixel)');

%% PCA (with SVD)
% de-mean data
m_data = mean(data_raw, 2);
```

```
data = data_raw - m_data;

[U,S,V] = svd(data,'econ');
% print singular values
disp(S);

% rank 1 approximation
data_rank1 = S(1,1)*U(:,1)*V(:,1)' + m_data;

% rank 2 approximation
data_rank2 = data_rank1 + S(2,2)*U(:,2)*V(:,2)';

% 2 principle components
n = size(data, 2);
y1 = S(1,1)/sqrt(n-1)*U(:,1);
y2 = S(2,2)/sqrt(n-1)*U(:,2);

% plot apprimations
figure
for i=1:6
    plot(data_rank1(i, :));
    hold on;
end
legend(data_label);
set(gca, 'Fontsize',16);
title('Rank-1 Approximation');
xlabel('t (frame)');
ylabel('position (pixel)');

figure
for i=1:6
    plot(data_rank2(i, :));
    hold on;
end
legend(data_label);
set(gca, 'Fontsize',16);
title('Rank-2 Approximation');
xlabel('t (frame)');
ylabel('position (pixel)');


%% Change of Basis (plot data with first 2 principle components)
data_proj = U'*data;

figure
plot(data_proj(1,:),data_proj(2,:),'k.','MarkerSize',10)
axis equal
hold on
y1_proj = U'*y1;
y2_proj = U'*y2;
c = compass(y1_proj(1),y1_proj(2));
set(c,'Linewidth',4);
c = compass(y2_proj(1),y2_proj(2));
set(c,'Linewidth',4);
set(gca, 'Fontsize',16);
title('Data in First 2 Principle Components');
xlabel('First Principle Component');
ylabel('Second Principle Component');
```

Listing 2: performing PCA analysis on data