Video object tracking and dimensional reduction with Singular Value Decomposition & Principle Component Analysis

Joe Henthorn
University of Washington

Winter 2020

Abstract: The goal of this paper is to demonstrate how to use the singular value decomposition (SVD) method to obtain the Principle components of a system and to preform an analysis from these results. This is called the Proper Orthogonal Decomposition (POD) or the Principle Component Analysis (PCA). The example in this paper is an attempt to track the motion of a mass attached to a spring with the spring fixed to a "solid" support. Three distinct cameras are used to record the oscillating can-spring system. With this video data, we track the object and acquire positional data in each camera reference frame. Then we use the SVD method to determine the Principle components and to produce a lower dimensional approximation for the can-spring system.

1 Introduction

This document is a guide for Proper Orthogonal Decomposition, also known as PCA. In this example we seek to The follow the motion of a paint can on a spring with a vertical oscillatory displacement. In four different experiments we track the motion of the paint can and determine its position with a simple tracking algorithm. First we demonstrates how to track an object in 2D space and time with a simple binary conversion method with a dynamic threshold. This document also demonstrates how to use the SVD algorithm to find a lower dimensional or lower rank approximation for the system in motion. Each experiment uses three cameras with unique reference frames to track the object. The first experiment is an ideal case with stable recordings and a low noise signal. The second experiment is a noisy signal case in which the cameras wobble as they record. In the third experiment the paint can is perturbed such that is has a horizontal displacement component giving it a pendulum like motion, and the fourth experiment has a horizontal displacement and rotation. The motion of the spring mass system solution can be determined by its physical governing equation.

$$\frac{d^2 f(t)}{dt^2} = -\omega^2 f(t) \tag{EQ:1}$$

Therefore this example is essentially overkill for determining the motion of the object. However, not all systems are as nice and simple with pre-existing solutions. The SVD algorithm is a powerful method for approximating the behavior of a system from data alone. Herein, we apply this technique to recreate the motion of the system.

2 Theoretical Background

The core of this method is to eliminate redundancies in our data, and involves computation of the covariance of a system. By using the dot product of the data vector with its transpose we project the data set onto itself resulting in the covariance matrix.

$$X = \begin{bmatrix} x_a \\ y_a \\ x_b \\ y_b \\ \vdots \end{bmatrix} \tag{EQ:2}$$

$$C_x = \frac{1}{n-1} X X^T \tag{EQ:3}$$

where C_x is a square matrix (n x n)

The covariance matrix shows correlations in the data and which values in the data set are statistically independent or dependant. Large variance or covariance values are indicative of large "overlaps" in the projections of these values and therefore are statistically dependant, while small values show minimal overlap in the projection of the data. Variance and covariance values of zero are indicative of orthogonal components, which are interpreted as statistically independent.

The SVD algorithm is a matrix algebra technique that can be used to diagonalize a matrix such that its off diagonals are zero. The U matrix defines the new basis of the system and the V matrix is the projection of U and Σ. This allows us to decompose a system into its Eigenvectors and Eigenvalues.

$$A = U \Sigma V^{\star} \tag{EQ:4}$$

Where U and V are unitary matrices and Σ is a diagonal matrix. The diagonal values of Σ are the singular values σ_j ordered from largest to smallest.
Here, the U matrix represents the rotation matrix with eigenvectors as columns. The S matrix represents the diagonal matrix of squared singular values, which are also eigenvalues. Conversely, the square root of the eigenvalues are the singular values.

$$\lambda = \Sigma^2$$

$$C_x = V \Lambda V^{-1}$$

$$C_x = \begin{pmatrix} \sigma_1^2 & \cdots & \cdots \\ \cdots & \sigma_2^2 & \cdots \\ \cdots & \cdots & \ddots \end{pmatrix} \tag{EQ:5}$$

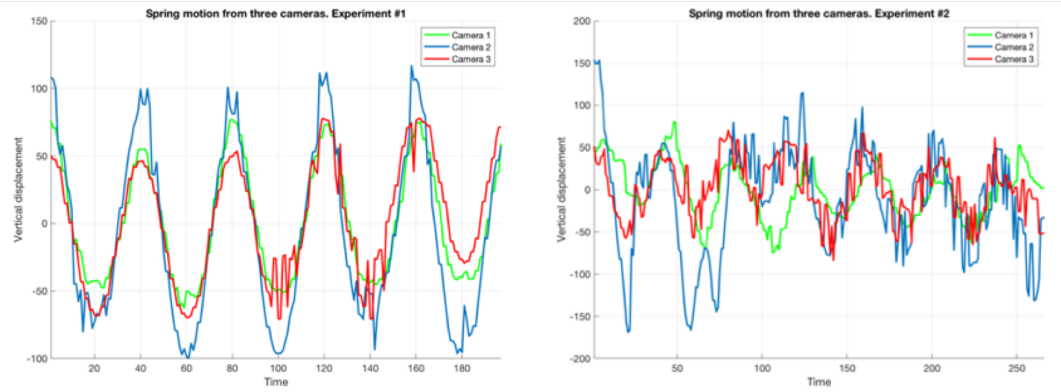
We can also determine the "energy" of the system by computing a ratio of Frobenius norms.

$$Energy = \frac{\Sigma_{j=1}^{N} \sigma_j}{\Sigma \sigma} \tag{EQ:6}$$

3 Methods & Algorithm Implementation

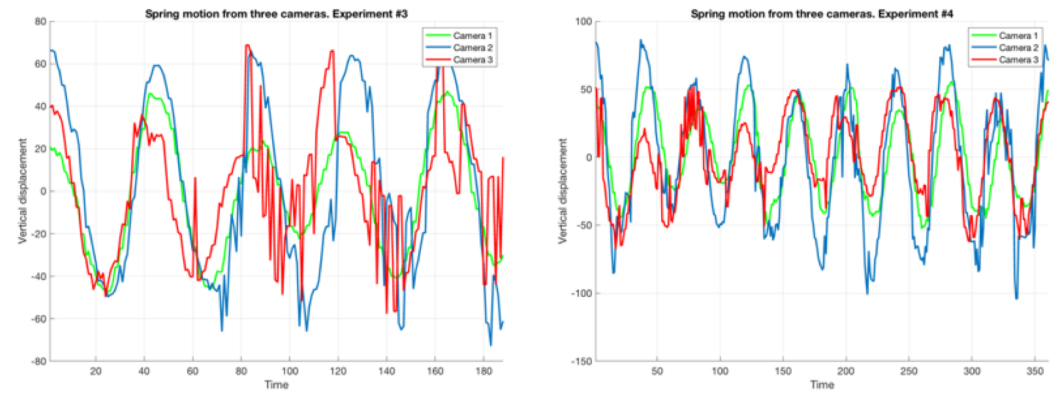
For this example, we first need to process our video data to track the moving object. We convert each RGB frame into a gray scale representation with the **rgb2gray()**. Next, we crop and binarize the image by using **imbinarize(frame, threshold)**. This allows us to view the frame as a logical array of 1's and 0's. The binary threshold is dynamic in that its threshold is set to 0.99 unless the object is not detected. The threshold is subsequently decrease until the object is detected in that specific frame. Next, we find the true

values (1's) in each frame and determine their index position over a for loop that cycles through all pixels in the frame (n x m). The index values are then used to average the true values resulting in the average position of the can. Lastly, the mean of the data is subtracted from the position data to isolate the can object.
The attached code also uses an nearest-neighbor interpolation method to preserve the position of the can when the object cannot be found, or if the position value is NaN.
Once the position data is collected for all three camera recordings, they are put into a matrix with the rows as the x and y coordinate data for each camera.
Next, we align each vertical position trace by their first peak, and then truncate the ends of the vectors such that the are all the same length.



.

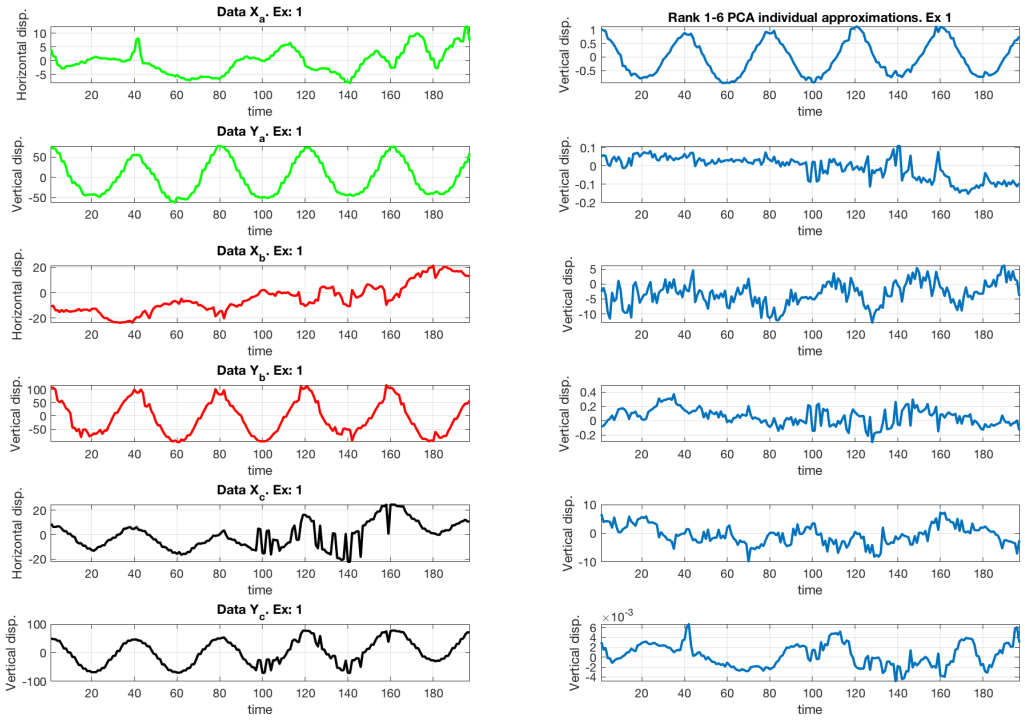
Figure 1: Plot of camera data for all three cameras for experiments 1-2 . Data shows vertical displacement. All data is aligned in time.



.

Figure 2: Plot of camera data for all three cameras for experiments 3-4 . Data shows vertical displacement. All data is aligned in time.

Then we preform the SVD and find the singular values and energy of the each system, and we compare the rank 1, rank 2, and rank 3 approximations with the average paint can location.

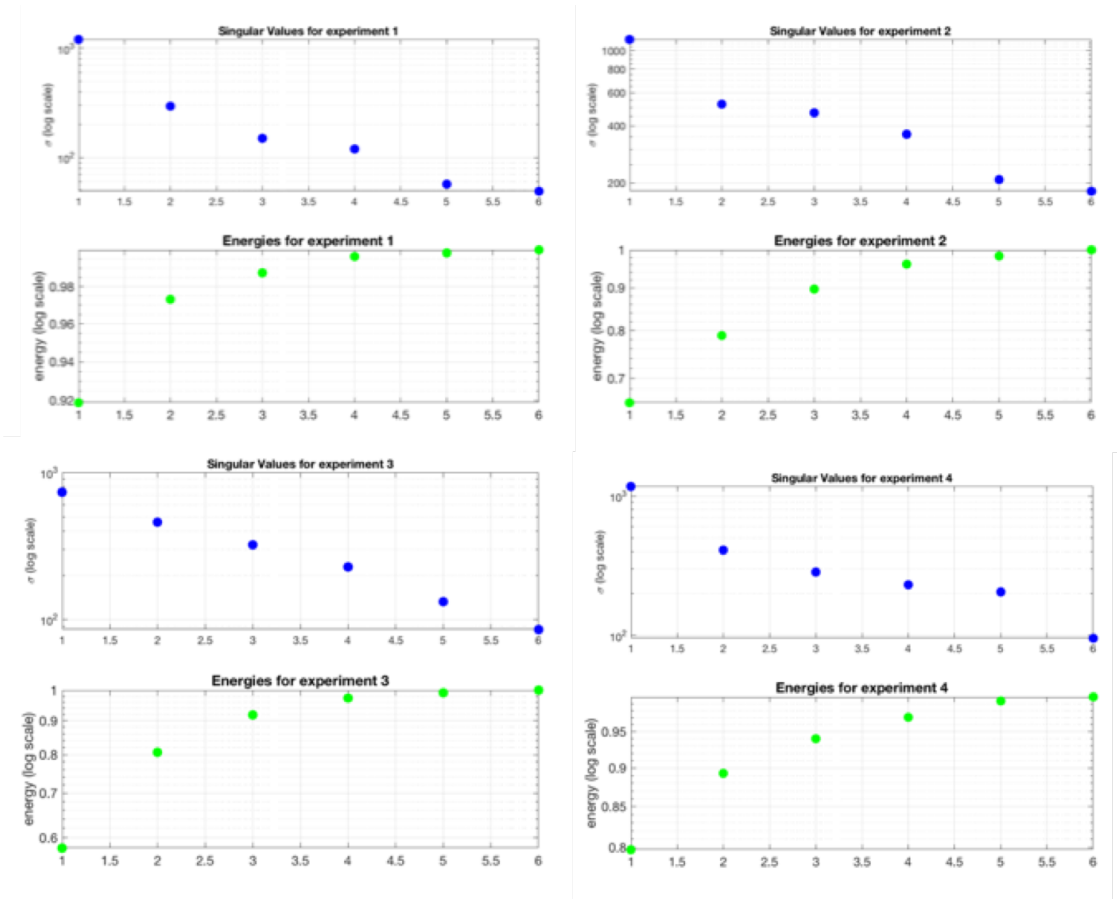


.

Figure 3: Both plots are from experiment 1. Left side of figure shows x and y camera data vs time for all three cameras. Right side of figure shows the SVD approximations with rank 1 approximation as very similar to the original y data vectors

4 Computational Results

In this example we computed the singular values and cumulative energies of each system.



.

Figure 4: Figure shows the singular values and energies for each experiment. All plots are log plots in the y direction.

Table 1. Singular values (1-3) for experiments 1-4

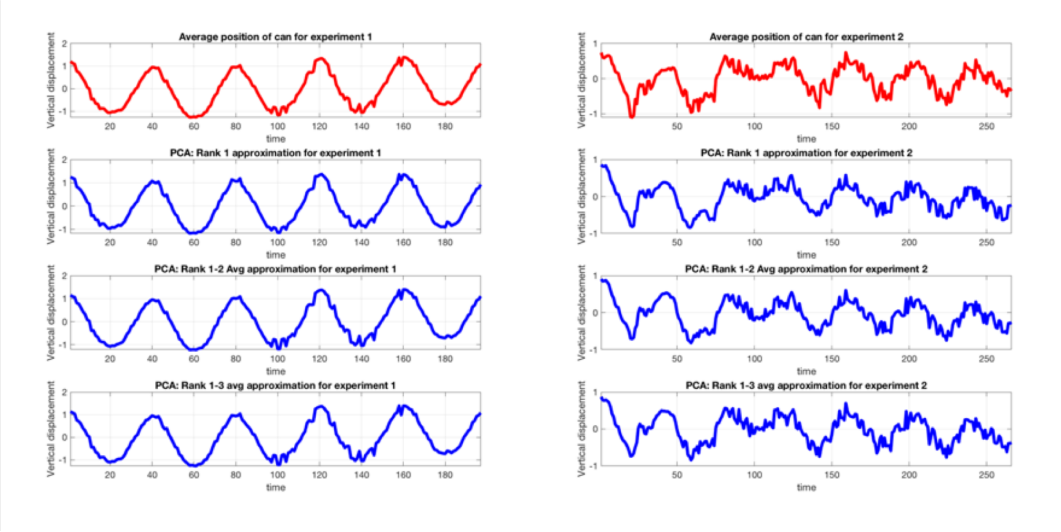
Experiment	Singular values 1	Singular values 2	Singular values 3
1	1207.88	293.31	150.253
2	1151.28	521.47	468.703
3	735.48	460.6	323.0
4	1175.375	407.191	284.83

Table 2. Energies values for experiments 1-4

Experiment	Energy 1	Energy 2	Energy 3	Energy 4	Energy 5	Energy 6
1	0.919	0.054	0.014	0.0089	0.002	0.0015
2	0.6548	0.1343	0.1085	0.0646	0.0213	0.0162
3	0.5790	0.2271	0.1117	0.0555	0.0187	0.0078
4	0.797583	0.09572	0.0468	0.0304	0.0240	0.0052

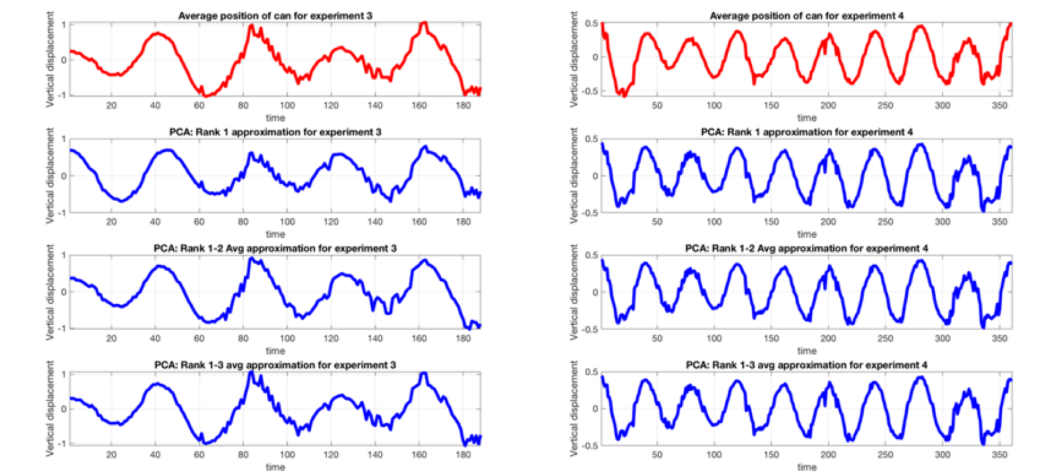
.

Figure 5: Table shows first three singular values and all energy values for all four experiments. Notes: Most of the energy is contained in the rank 1 approximation for all cases.



.

Figure 6: Top plot shows the average y displacement. Bottom plots show the rank 1-3 approximations.



.

Figure 7: Top plot shows the average y displacement. Bottom plots show the rank 1-3 approximations

5 Conclusion

In this example we used camera data to acquire position data and then used the SVD to reconstruct the position data with a lower dimensional approximation. We started with six dimensions per experiment, and found that the majority of the information of each system was captured by the first singular value and rank 1 approximation.

Here we found that the ideal case recording produced the best looking data, and the best rank 1 approximation. We also found the the rank 1 approximation of the noisy experiment was less good than the ideal case, but still possessed the majority of the system information. The results from the first two experiments show that this system is essentially a 1D system with mostly vertical displacement.

The last two experiments had much less information in the first singular values and rank 1 approximation

due to the horizontal displacement components. The results from the last two experiments show that these systems are essentially a 2D system with mostly vertical displacement, but also horizontal displacement. In this case it was found that a rank 1-3 averaged approximation could fully capture the vertical and horizontal displacement characteristics of the original system.
Lastly, the rotation of the last experiment was not present in the data, because the tracking code could not discern the rotation with the binary method.