AMATH 482: HOMEWORK 2

ROHAN PANDEY

University of Washington, Seattle, WA
rpande@uw.edu

Abstract. This report focuses on projecting and recognizing movements of the humanoid robot OptimuS-VD, which records 38 joint movements at 60Hz. We reduce the dimensionality of the data, visualize the movements, and design an algorithm to classify actions in real-time. The dataset consists of a 114×100 matrix, representing joint positions over 1.4 seconds. Three distinct movements are recorded, and our algorithm is tested on sample data from each.

1. Introduction and Overview

In the fast-evolving world of robotics, it is essential to be able to recognize the movements of a robot in real-time. This is important for a variety of reasons, such as safety, efficiency, and even for the purpose of entertainment! In this report using Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) we will project the recordings of the robot OptimuS-VD to a lower dimension and then design an algorithm that will recognize which movement OptimuS-VD is performing in real-time.

The data provided to us is a matrix of 114×100, where the first dimension records the locations of the joints and the second dimension the time steps. The data has been recorded for 3 different movements for 1.4 seconds and using the test sample of each movement we will test our algorithm. We also visualize the movements of the robot using the data provided to us, which will help us understand the movements of the robot better. This project is important in the field of robotics as it will help us understand the movements of the robot better and also help us recognize the movements of the robot in real-time.

In the following sections, we present the theoretical background for SVD, PCA and the Frobenius norm that was used in finding out the energy added by the singular values from SVD. We then discuss the implementation of the algorithm and then results obtained from the algorithm. We then conclude the report with a summary of the results obtained and the conclusions drawn from the results.

2. Theoretical Background

2.1. **Singular Value Decomposition (SVD).** Singular Value Decomposition (SVD) is a factorization of a matrix A into three matrices U, Σ, and V^T such that $A = U\Sigma V^T$. Here, U and V are orthogonal matrices and Σ is a diagonal matrix with the singular values of A on the diagonal. The singular values are the square roots of the eigenvalues of A^TA or AA^T. The singular values are always non-negative and are arranged in descending order. The SVD is used in a variety of applications such as data compression, image compression, and in the field of robotics to analyze the movements of robots.

Date: February 14, 2025.

2.2. **Principal Component Analysis (PCA).** PCA is a technique for reducing the dimensionality of a dataset while preserving as much variance as possible. It finds a new orthogonal basis such that:

- The first principal component captures the maximum variance
- Each subsequent component captures the maximum remaining variance under the constraint that it is orthogonal to the previous components

Given a dataset $\mathbb{X}$ of size $m \times n$, where m is the number of samples and n is the number of features, PCA follows the following steps:

(1) Find the mean and subtract it from each feature:
$$X_{\text{centered}} = X - \mu$$
(2) Compute the covariance matrix of the dataset:
$$\Sigma = C = \frac{1}{m} X_{\text{centered}}^T X_{\text{centered}}$$
(3) Compute the SVD of the covariance matrix:
$$C = U \Sigma V^T$$
(4) The principal components are the columns of U:
$$Z = X_{\text{centered}} U$$
(5) Now we finally work on dimension reduction by selecting the first k columns of Z to get the reduced dataset Z_k:
$$Z_k = X_{\text{centered}} U_k$$
where U_k is the matrix of the first k columns of U

PCA is a powerful technique for reducing dimensionality while preserving structure. It finds a new basis that maximizes variance and transforms data into a lower-dimensional space.

2.3. **Frobenius Norm.** The singular values from the SVD of a matrix A can be used to calculate the energy of the matrix. The energy of a matrix is defined as the sum of the squares of the singular values of the covariance matrix. The Frobenius norm of a matrix A is defined as [4]:

$$\|A\|_F = \sqrt{\sum_{i=1}^{r} \sigma_i^2}$$

Where σ_i are the singular values of the matrix A. The Frobenius norm is used to calculate the energy of the matrix and is used in the context of PCA to find the energy added by the singular values. In terms of the modes that we are using we can determine how much variance there is from the real dataset and our prediction with this formula:

$$E_m = \frac{\sum_{i=1}^{m} \sigma_i^2}{\sum_{i=1}^{N} \sigma_i^2}$$

Once we get our energy we can determine how many modes we need to keep getting a good approximation of the dataset. Through SVD we can reduce the dimensionality of the dataset and still get a good approximation of the dataset by removing uncertain modes.

3. Algorithm Implementation and Development

The code that was needed for this report involved several key steps, using Python libraries such as `Matplotlib` [1], `NumPy` [2], and `Plotly` [3]. The following algorithm outlines a general overview of the steps taken to analyze the data and project it to a lower dimension and solves all the tasks:

Algorithm 1 Motion Recognition via PCA

Input: Training samples X_{train}, Test samples X_{test}
Output: PCA projections, classification accuracies
Preprocessing:
  Load and concatenate all training samples.
  Center the data by subtracting the mean.
PCA Computation:
  Perform SVD on the centered data: $[U, \Sigma, V^T] \leftarrow \text{SVD}(X_{\text{train}})$.
  Compute and plot cumulative energy to select the number of modes.
  Energy per mode found using Frobenius norm as:

$$E_{\text{cumulative, k}} = \sum_{i=1}^{k} \frac{s_i^2}{\sum_j s_j^2}$$

  where k is the number of modes and s_l is the l-th singular value.
Visualization:
  Project data onto the first 2 and 3 PCA modes.
  Visualize trajectories in 2D and 3D spaces.
Classification:
  For each k (number of PCA modes) from PCA Computation:
    Compute centroids for each movement class.
    Classify samples by assigning the label of the nearest centroid.
    Evaluate accuracy using ground truth labels.
Testing:
  Project test samples onto the PCA space.
  Classify and evaluate performance on test data.

4. Computational Results

First let's determine using our training data, at what PCA mode do we achieve a certain amount of cumulative energy. For this task we will be testing $70\%, 80\%, 90\%, 95\%, 99\%$ cumulative energy.

Here's the results in a table:

Cumulative Energy	Number of PCA Modes
70%	2
80%	3
90%	5
95%	7

The corresponding plot is below:

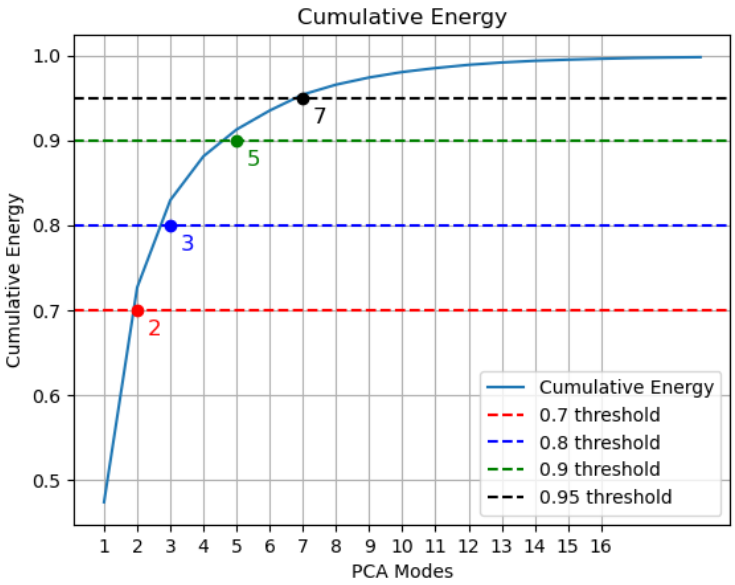

FIGURE 1. The PCA Mode needed to achieve a certain amount of cumulative energy

Following up from the results above, we can now visualize the trajectories of the robot in 2D and 3D space (provided in the code). The following figure however only shows the 3D PCA representation of the robot's movements and includes the calculated centroids.

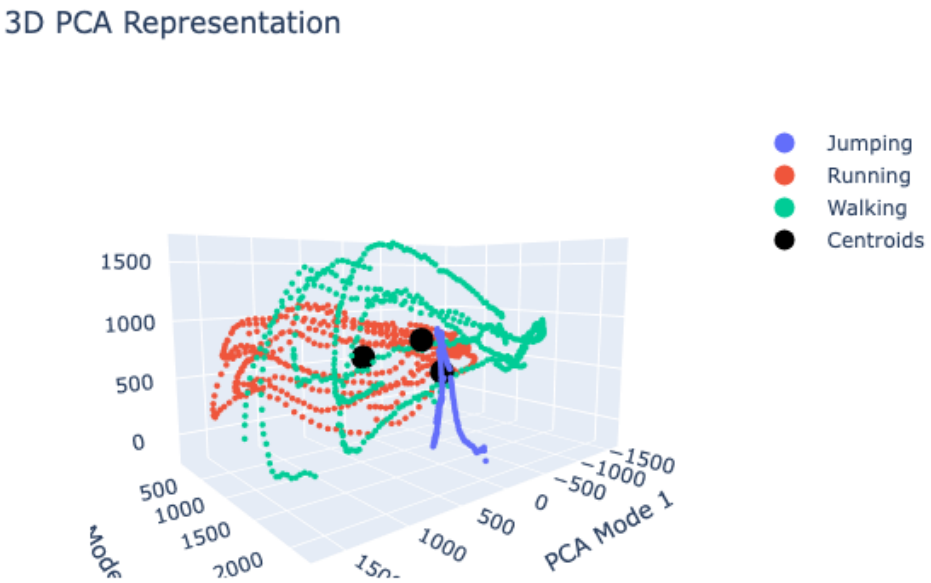

FIGURE 2. 3D Representation

Continuing on from the results above, we can work on testing the accuracy of our algorithm on the test data. The following graph shows the accuracy of the algorithm on the test data for different PCA modes, something important to notice is that the accuracy actually seems to hit an upper limit of 91.06% around $k = 13$ modes.

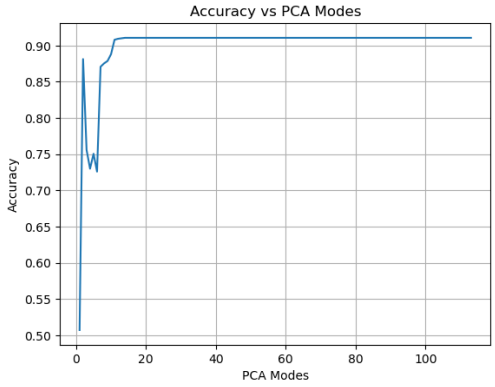

FIGURE 3. Accuracy of the algorithm on the test data

As per our tasks, we now continue on to test the algorithm on the test data and see how well it performs. And to get more accurate data we also find the accuracy of the modes that allowed us to achieve $70\%, 80\%, 90\%, 95\%, 99\%$ cumulative energy. Additionally, note that here, our constant limit seems to be around 0.9534 for the accuracy of the algorithm on the test data.

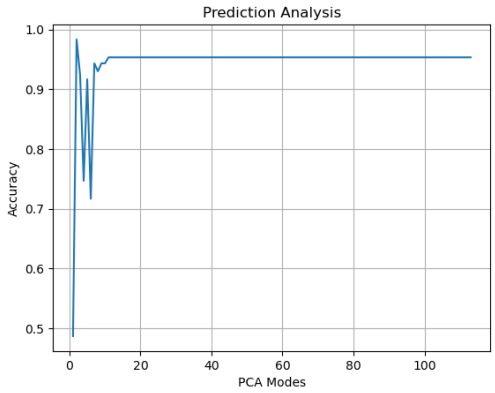

FIGURE 4. Accuracy of the algorithm on the test data

Cumulative Energy	**Accuracy**
70%	0.9834
80%	0.9234
90%	0.9167
95%	0.9434

5. SUMMARY AND CONCLUSIONS

In this report, we projected the movements of the humanoid robot OptimuS-VD to a lower dimension using PCA and SVD. We visualized the movements of the robot in 2D and 3D space and designed an algorithm to classify the movements in real-time. We found that we needed 2 PCA

modes to achieve 70% cumulative energy, 3 modes for 80%, 5 modes for 90%, and 7 modes for 95% cumulative energy. Furthermore, we also found that the accuracy of the algorithm on the training data was constant at 91.06% for more than 13 PCA modes. We also found that the accuracy of the algorithm on the test data was 98.34% for 70% cumulative energy, 92.34% for 80%, 91.67% for 90%, and 94.34% for 95% cumulative energy. And similarly as $k \to \infty$ the accuracy went to 0.9534. It is interesting to compare between the train and test accuracies and see that the test accuracy is higher than the train accuracy. This might be because the test data is more structured and the algorithm is able to classify the movements better. Regardless, the algorithm shows that the training data was able to classify the movements with a relatively high accuracy, and when tested on the testing data, a similarly high performance was found. This algorithm can be used in real-time applications to classify the movements of the robot and can be used in a variety of applications such as safety, efficiency, and entertainment.

Acknowledgements

The author is thankful to Professor Frank for providing the materials and resources needed to complete this assignment. The author is also thankful to the peers Eric Ye and Chenab for discussion and implementation of visualization techniques in Python.

References

[1] Matplotlib Developers. Matplotlib: Visualization with Python. `https://matplotlib.org/`, 2025. [Accessed: Jan. 26, 2025].

[2] NumPy Developers. NumPy: Fundamental package for scientific computing in Python. `https://numpy.org/`, 2025. [Accessed: Jan. 26, 2025].

[3] Plotly Technologies Inc. Plotly: The interactive graphing library for Python. `https://plotly.com/`, 2025. [Accessed: Jan. 26, 2025].

[4] Wikipedia contributors. Matrix norm — Wikipedia, The Free Encyclopedia, 2025.