

**Kathmandu University**  
**Dhulikhel, Kavrepalanchok**



**A Project Report**

**on**

**“Principal Component Analysis in Image Processing”**

**[Course Code: MCSC 202]**

**Submitted by**

**Bibash Acharya**

**Registration No: 022409-17**

**Group: Computer Engineering**

**Batch: 2019**

**Submitted to:**

**Dr. Gokul KC**

**Department of Mathematics**

**Submission Date: 1<sup>st</sup> September, 2023**

## **Abstract**

The proposal has been drafted in order to meet the requirements of course MCSC 202 (Numerical Methods) offered by the Department of Mathematics, Kathmandu University. In this project, I intend to work in Principal Component Analysis in Image Processing. Principal Component Analysis (PCA) in image processing is a technique used for dimensionality reduction, feature extraction, and data compression. It transforms high-dimensional image data into a lower-dimensional form while retaining most of the significant information. PCA involves flattening images into vectors, calculating the covariance matrix, and finding its eigenvalues and eigenvectors. In facial recognition, eigenfaces are the principal components (eigenvectors) derived from a set of facial images. They represent the "basis images" that capture the most important features distinguishing one face from another.

**Keywords:** Principal Component Analysis (PCA), Eigen Faces

# Table of Contents

Abstract.....	i
List of figures .....	iv
List of Abbreviations/Acronyms.....	v
Chapter 1: Introduction .....	1
1.1 Background.....	1
1.2 Objectives.....	2
1.2 Motive and Significance.....	2
Chapter 2: Related Works .....	3
2.1 Face Recognition using PCA .....	3
2.2 Principal Component Analysis testing on Image data.....	4
Chapter 3: Implementation.....	5
3.1. Tools Used .....	5
3.1.1 Anaconda .....	5
3.2 Packages and Libraries Used .....	5
3.3 Code Logic .....	6
3.4 Demonstration .....	9
Chapter 4: Discussion on the Achievements.....	12
4.1 Discussion.....	12
4.2 Features.....	12
4.2.1. Successful Dimensionality Reduction: .....	12
4.2.2 Data Compression:.....	12
4.2.3. Enhanced Computational Efficiency: .....	12

<b>Chapter 5. Conclusion and Recommendation.....</b>	<b>13</b>
<b>5.1 Limitations.....</b>	<b>13</b>
<b>5.2 Future Enhancements.....</b>	<b>13</b>
<b>Chapter 6: Source Code .....</b>	<b>14</b>
<b>Chapter 7: Video Demonstration.....</b>	<b>14</b>
<b>References .....</b>	<b>15</b>

## List of figures

Figure 3.3(a): Importing PCA Algorithm in main file	6
Figure 3.3(b): PCA Algorithm Class in main file	6
Figure 3.3(c): Logic for Recorded Files	7
Figure 3.3(d): Logic for Video Files	7
Figure 3.3(e): Logic for Image Files	8
Figure 3.3(f): Ending Code for main file	8
Figure 3.4(a): Running Face_Recognition.py file	9
Figure 3.4(b): Running Command in Code Editor Terminal	9
Figure 3.4(c): Original Image	10
Figure 3.4(d): Eigen Face 0	10
Figure 3.4(e): After PCA Image	11

## **List of Abbreviations/Acronyms**

PCA = Principal Component Analysis

ML = Machine Learning

# Chapter 1: Introduction

## 1.1 Background

Image processing involves analyzing and manipulating visual data, often represented as high-dimensional matrices of pixel values. With the increasing size and complexity of image datasets, efficiently processing and analyzing these large volumes of data becomes a significant challenge. Traditional methods may struggle with the computational demands and the "curse of dimensionality," where the performance of algorithms deteriorates as the number of dimensions increases.

Principal Component Analysis (PCA) emerges as a powerful solution to this problem by reducing the dimensionality of image data while preserving its most critical features.

In image processing, PCA is particularly valuable due to the high dimensionality inherent in images. For instance, a typical grayscale image of  $100 \times 100$  pixels has 10,000 dimensions. When applied to a collection of such images, PCA identifies the directions of maximum variance, represented by eigenvectors (also known as principal components), and reduces the data's dimensionality by projecting it onto these principal components. The corresponding eigenvalues indicate the amount of variance captured by each principal component, guiding the selection of the most informative components.

One notable application of PCA in image processing is in facial recognition, where it is used to derive "eigenfaces." These eigenfaces are the principal components that capture the most distinctive features of human faces, enabling efficient and accurate recognition. By reducing the dimensionality of facial images while retaining the most significant variations, PCA allows for the creation of a compact and informative representation of faces, facilitating tasks such as identity verification and face detection.

Overall, PCA has become a fundamental tool in image processing, offering a robust method for dimensionality reduction, noise reduction, and feature extraction. It enhances the efficiency and effectiveness of image analysis techniques, making it an essential component in modern image processing workflows.

For this project, I have used Anaconda ( A Python Data Science Distributions ) along with other Python Data Science Libraries like numpy, scipy, and opencv.

## **1.2 Objectives**

When the project is completed, I aim to achieve the following objectives:

1. To apply PCA to reduce the dimensionality of image data while retaining the most significant features, enhancing computational efficiency in image analysis tasks.
2. To extract key features from images, such as eigenfaces in facial recognition, to improve the accuracy and effectiveness of classification and recognition algorithms.

## **1.2 Motive and Significance**

The primary motive behind employing Principal Component Analysis (PCA) in image processing is to address the challenges associated with high-dimensional data. Images, inherently high-dimensional, pose significant computational and analytical difficulties, particularly when dealing with large datasets. By reducing the dimensionality of image data, PCA simplifies the complexity, making it more manageable for various tasks such as classification, recognition, and compression.

The significance of PCA in image processing lies in its ability to extract the most important features from images while discarding redundant or less informative data. This leads to several key benefits: Enhanced Computational Efficiency, Improved Accuracy, and Data Compression.

Therefore, PCA serves as a foundational technique in image processing, enabling more efficient and effective handling of complex visual data, and contributing to advancements in various fields such as computer vision, medical imaging, and multimedia applications.



## Chapter 2: Related Works

### 2.1 Face Recognition using PCA

This is a similar project done by an Indian Programmer which applies Principal Component Analysis to solve problem of Face Recognition (Code Heroku, 2019).



*fig 2.1: Face Recognition using PCA (Code Heroku Project)*

Mint is a free personal finance management tool developed by Intuit that allows users to manage their finances in one place. Mint allows you to create a budget and track your spending in real-time. You can set spending limits for different categories such as groceries, dining out, and entertainment, and Mint will track your spending in those categories. Mint allows you to track your investments and monitor your portfolio performance. You can see your asset allocation and compare your returns to market benchmarks (Mint, n.d.).

You can set financial goals such as saving for a down payment on a house or paying off debt. Mint will help you track your progress towards those goals.

## 2.2 Principal Component Analysis testing on Image data

This project involves application of PCA technique on image data and assessing its performance in terms of information retention and compressibility (Kumar, 2020)

Comparison of RGB image and principal components composite image



*Fig 2.2: Original vs PCA Image (Principal Component Analysis testing on Image data project)*

## **Chapter 3: Implementation**

### **3.1. Tools Used**

#### **3.1.1 Anaconda**

The Anaconda Distribution is a popular open-source distribution of the Python and R programming languages. It's widely used in data science, machine learning, and scientific computing due to its ease of use and comprehensive package management (Anaconda.org, 2024).

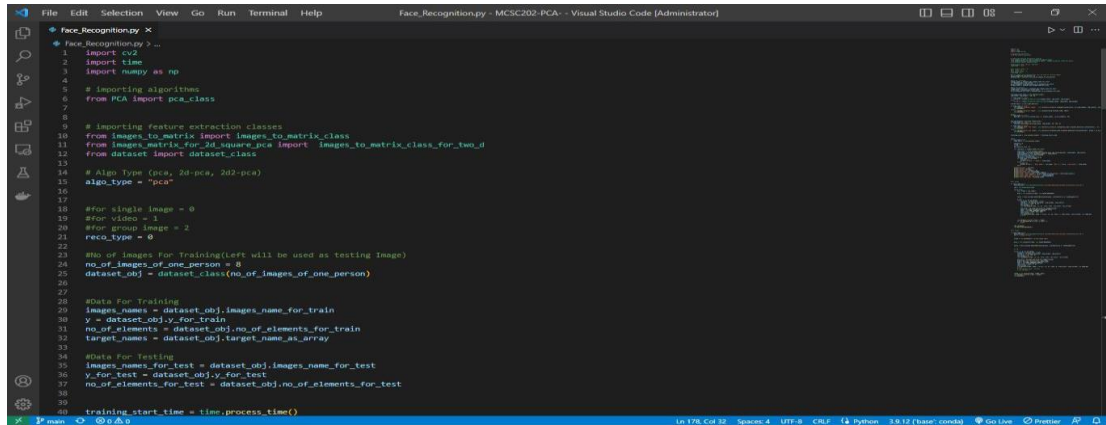
### **3.2 Packages and Libraries Used**

- numpy
- opencv
- scipy

### 3.3 Code Logic

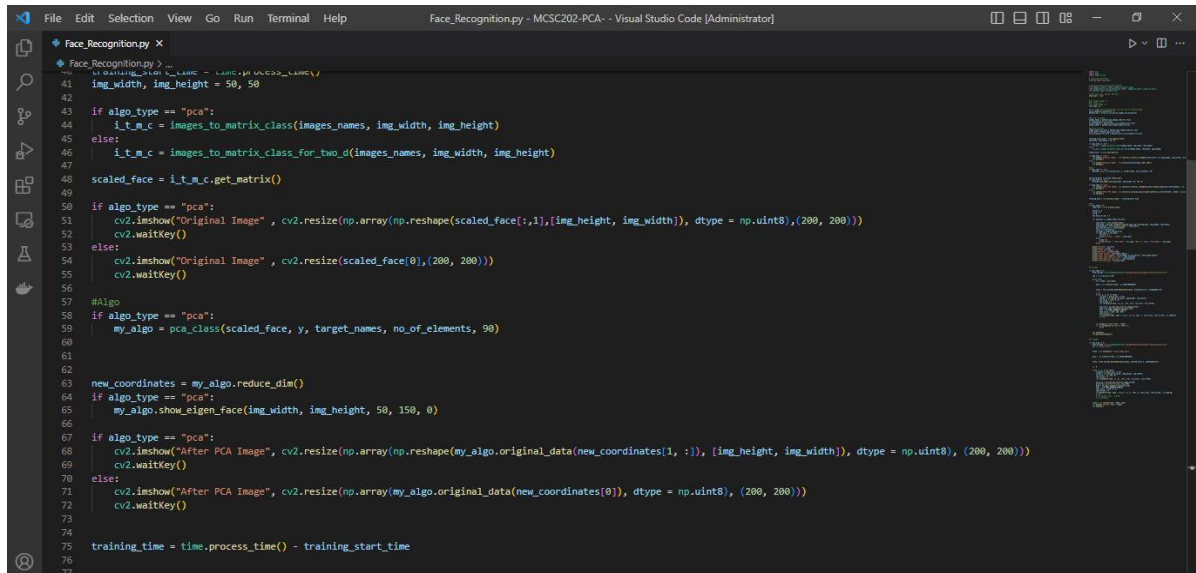
The project uses ORL dataset. The main code to run is in Face\_Recognition.py file which imports PCA Algorithms code modules from PCA.py and other necessary file.

Here are some screenshots from our project available:



```
File Edit Selection View Go Run Terminal Help Face_Recognition.py - MCSC202-PCA - Visual Studio Code [Administrator]
Face_Recognition.py X
Face_Recognition.py > ...
1 import cv2
2 import time
3 import numpy as np
4 # Importing algorithms
5 from PCA import pca_class
6
7
8 # Importing feature extraction classes
9 from images_to_matrix import images_to_matrix_class
10 from images_matrix_for_2d_square_pca import images_to_matrix_class_for_two_d
11 from dataset import dataset_class
12
13 # Algo type (pca, 2d-pca, 2d2-pca)
14 algo_type = "pca"
15
16
17 # For single image = 0
18 # For video = 1
19 # For group image = 2
20 reco_type = 0
21
22 # No. of images for Training (left will be used as testing image)
23 no_of_images_of_one_person = 8
24 dataset_obj = dataset_class(no_of_images_of_one_person)
25
26
27 # Data For Training
28 images_names = dataset_obj.images_name_for_train
29 y = dataset_obj.y_for_train
30 no_of_elements = dataset_obj.no_of_elements_for_train
31 target_names = dataset_obj.target_name_as_array
32
33 # Data For Testing
34 images_names_for_test = dataset_obj.images_name_for_test
35 y_for_test = dataset_obj.y_for_test
36 no_of_elements_for_test = dataset_obj.no_of_elements_for_test
37
38
39 training_start_time = time.process_time()
```

fig 3.3 (a): Importing PCA Algorithm in main file



```
File Edit Selection View Go Run Terminal Help Face_Recognition.py - MCSC202-PCA - Visual Studio Code [Administrator]
Face_Recognition.py X
Face_Recognition.py > ...
41 training_start_time = time.process_time()
42 img_width, img_height = 50, 50
43
44 if algo_type == "pca":
45     i_t_m_c = images_to_matrix_class(images_names, img_width, img_height)
46 else:
47     i_t_m_c = images_to_matrix_class_for_two_d(images_names, img_width, img_height)
48
49 scaled_face = i_t_m_c.get_matrix()
50
51 if algo_type == "pca":
52     cv2.imshow("Original Image", cv2.resize(np.array(np.reshape(scaled_face[:,1],[img_height, img_width])), dtype = np.uint8), (200, 200)))
53     cv2.waitKey()
54 else:
55     cv2.imshow("Original Image", cv2.resize(scaled_face[0], (200, 200)))
56     cv2.waitKey()
57
58 # Algo
59 if algo_type == "pca":
60     my_algo = pca_class(scaled_face, y, target_names, no_of_elements, 90)
61
62
63 new_coordinates = my_algo.reduce_dim()
64 if algo_type == "pca":
65     my_algo.show_eigen_face(img_width, img_height, 50, 150, 0)
66
67 if algo_type == "pca":
68     cv2.imshow("After PCA Image", cv2.resize(np.array(np.reshape(my_algo.original_data[new_coordinates[1, :]], [img_height, img_width])), dtype = np.uint8), (200, 200)))
69     cv2.waitKey()
70 else:
71     cv2.imshow("After PCA Image", cv2.resize(np.array(my_algo.original_data[new_coordinates[0]], dtype = np.uint8), (200, 200)))
72     cv2.waitKey()
73
74
75 training_time = time.process_time() - training_start_time
76
77
```

fig 3.3 (b): PCA Algorithm Class in main file

```

77
78 #Reco
79 if reco_type == 0:
80     time_start = time.process_time()
81
82     correct = 0
83     wrong = 0
84     i = 0
85     net_time_of_reco = 0
86
87     for img_path in images_names_for_test:
88
89         time_start = time.process_time()
90         find_name = my_algo.recognize_face(my_algo.new_cord(img_path, img_height, img_width))
91         time_elapsed = (time.process_time() - time_start)
92         net_time_of_reco += time_elapsed
93         rec_y = y_for_test[i]
94         rec_name = target_names[rec_y]
95         if find_name is rec_name:
96             correct += 1
97             print("Correct", " Name:", find_name)
98         else:
99             wrong += 1
100             print("Wrong:", " Real Name:", rec_name, "Rec Y:", rec_y, "Find Name:", find_name)
101             i += 1
102
103     print("Correct", correct)
104     print("Wrong", wrong)
105     print("Total Test Images", i)
106     print("Percent", correct/i*100)
107     print("Total Person", len(target_names))
108     print("Total Train Images", no_of_images_of_one_person * len(target_names))
109     print("Total Time Taken for reco", time_elapsed)
110     print("Time Taken for one reco", time_elapsed/i)
111     print("Training Time", training_time)
112
113
114
115 #For Video

```

*fig 3.3 (c): Logic for Recorded Files*

```

113
114
115 #For Video
116
117 if reco_type == 1:
118     face_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_frontalface_alt2.xml')
119
120     cap = cv2.VideoCapture(0)
121
122     while True:
123         ret, frame = cap.read()
124
125         gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
126
127         faces = face_cascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=7)
128
129         i = 0
130         for (x, y, w, h) in faces:
131             roi_gray = gray[y:y+h, x:x+w]
132             scaled = cv2.resize(roi_gray, (img_height, img_width))
133             rec_color = (255, 0, 0)
134             rec_stroke = 2
135             cv2.rectangle(frame, (x, y), (x+w, y+h), rec_color, rec_stroke)
136
137             new_cord = my_algo.new_cord_for_image(scaled)
138             name = my_algo.recognize_face(new_cord)
139             font = cv2.FONT_HERSHEY_SIMPLEX
140             font_color = (255, 255, 255)
141             font_stroke = 2
142             cv2.putText(frame, name + str(i), (x, y), font, 1, font_color, font_stroke, cv2.LINE_AA)
143             i += 1
144
145         cv2.imshow('Colored Frame', frame)
146         if cv2.waitKey(20) & 0xFF == ord('q'):
147             break
148
149
150
151
152

```

*fig 3.3 (d): Logic for Video Files*

```

153 cap.release()
154 cv2.destroyAllWindows()
155
156 #for Image
157
158 if reco_type == 2:
159     face_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_frontalface_alt1.xml')
160     dir = r'images/Group/'
161
162     frame = cv2.imread(dir+ "group_image.jpg")
163
164     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
165
166     faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=3)
167
168     i = 0
169
170     for (x, y, w, h) in faces:
171         roi_gray = gray[y:y+h, x:x+w]
172         scaled = cv2.resize(roi_gray, (img_height, img_width))
173         rec_color = (0, 255, 0)
174         rec_stroke = 5
175         cv2.rectangle(frame, (x, y), (x+w, y+h), rec_color, rec_stroke)
176
177         new_cord = my_algo.new_cord_for_image(scaled)
178         print("New Cord PCA="+str(i), new_cord)
179         name = my_algo.recognize_face(new_cord)
180         font = cv2.FONT_HERSHEY_SIMPLEX
181         font_color = (255, 0, 0)
182         font_stroke = 5
183         cv2.putText(frame, name + str(i), (x, y), font, 0, font_color, font_stroke, cv2.LINE_AA)
184         i += 1
185         # cv2.imshow('Face', scaled)
186         # cv2.waitKey()
187
188
189
190
191

```

fig 3.3 (e): Logic for Image Files

```

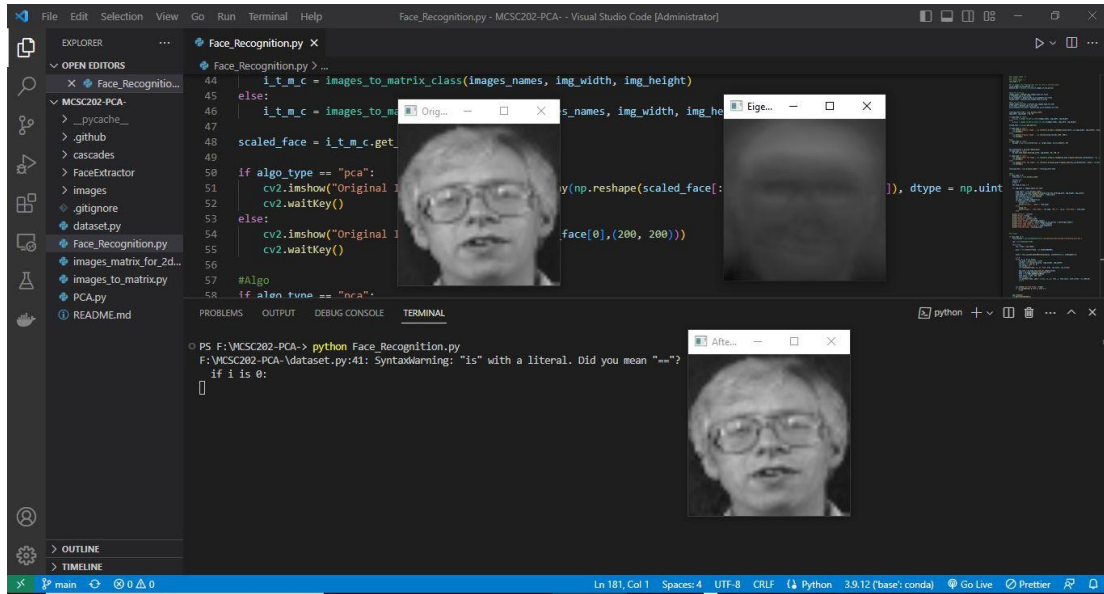
169
170 faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=3)
171
172 i = 0
173
174 for (x, y, w, h) in faces:
175     roi_gray = gray[y:y+h, x:x+w]
176     scaled = cv2.resize(roi_gray, (img_height, img_width))
177     rec_color = (0, 255, 0)
178     rec_stroke = 5
179     cv2.rectangle(frame, (x, y), (x+w, y+h), rec_color, rec_stroke)
180
181     new_cord = my_algo.new_cord_for_image(scaled)
182     print("New Cord PCA="+str(i), new_cord)
183     name = my_algo.recognize_face(new_cord)
184     font = cv2.FONT_HERSHEY_SIMPLEX
185     font_color = (255, 0, 0)
186     font_stroke = 5
187     cv2.putText(frame, name + str(i), (x, y), font, 0, font_color, font_stroke, cv2.LINE_AA)
188     i += 1
189     # cv2.imshow('Face', scaled)
190     # cv2.waitKey()
191
192
193
194 frame = cv2.resize(frame, (1080, 568))
195 cv2.imshow('Colored Frame', frame)
196 cv2.waitKey()
197
198
199
200
201
202
203

```

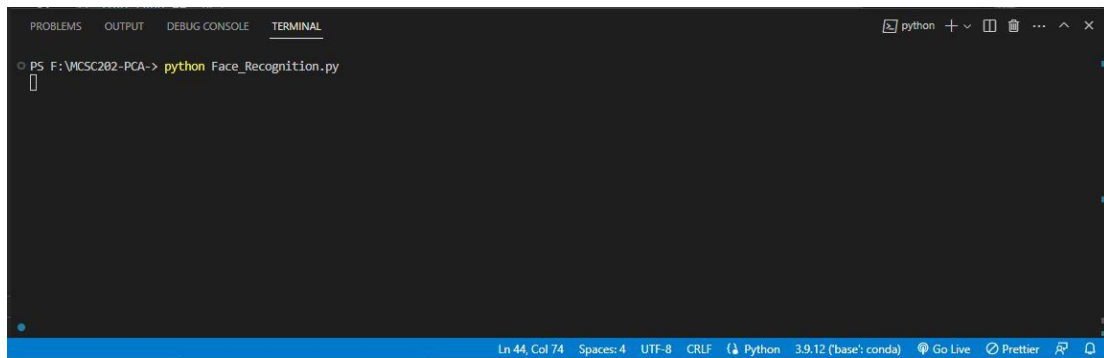
fig 3.3 (f): Ending Code for main file



### 3.4 Demonstration



*fig 3.4(a): Running FaceRecognition.py file*



*fig 3.4(b): Running Command in Code Editor Terminal*



*fig 3.4(c): Original Image*



*fig 3.4(d): Eigen Face 0*





*fig 3.4(e): After PCA Image*

## **Chapter 4: Discussion on the Achievements**

Although many difficulties were faced in building this project, the project ran successfully, therefore, the objective was achieved,. So, basically in this section achievements of this project has been discussed.

### **4.1 Discussion**

The application of Principal Component Analysis (PCA) in this image processing project has led to several notable achievements, demonstrating its effectiveness in addressing the challenges associated with high-dimensional image data.

### **4.2 Features**

The features available in our project are listed below.

#### **4.2.1. Successful Dimensionality Reduction**

PCA effectively reduced the dimensionality of the image data, transforming large datasets into more manageable forms without significant loss of critical information.

#### **4.2.2 Data Compression**

PCA enabled effective data compression, significantly reducing the storage requirements for large image datasets.

#### **4.2.3. Enhanced Computational Efficiency**

The reduction in dimensionality and the elimination of noise led to enhanced computational efficiency across various image processing tasks.

## **Chapter 5. Conclusion and Recommendation**

After all the hard work of several weeks, the project turned out as we expected. Objectives were successfully met for implementing learning and coding algorithms for Principal Components Analysis in Image Processing. There were shortcomings in different sections but altogether the project demonstration is satisfactory.

### **5.1 Limitations**

Some of the shortcomings of this Principal Component Analysis Projects are:

- It is limited to only Images File Data only.
- It is accurate for small dataset.

### **5.2 Future Enhancements**

If I plan to continue the development of our project in the future, some of the enhancements could be:

- To expand this project to work on video files .
- To work on larger latasets

## **Chapter 6: Source Code**

<https://github.com/acharyabibash/MCSC202-PCA->

## **Chapter 7: Video Demonstration**

[https://www.dropbox.com/scl/fi/hn7s8mgskge20vtpadnum/PCA\\_Demo.mp4?rlkey=hbkkrrzfbbtwjte2d86j6opq5&st=3f9wxlw9&dl=0](https://www.dropbox.com/scl/fi/hn7s8mgskge20vtpadnum/PCA_Demo.mp4?rlkey=hbkkrrzfbbtwjte2d86j6opq5&st=3f9wxlw9&dl=0)

## References

Anaconda.org. (2024). *About-us*. Retrieved from Anaconda.com:  
<https://www.anaconda.com/about-us>

Code Heroku. (2019). *Intro to Machine Learning*. Retrieved from github.com:  
<https://github.com/codeheroku/Introduction-to-Machine-Learning/tree/master/Face%20Recognition%20Using%20PCA>

Kumar, S. (2020). *Principal-Component-Analysis-testing-on-Image-data*. Retrieved from github.com: <https://github.com/Skumarr53/Principal-Component-Analysis-testing-on-Image-data>