Technological Institute of the Philippines	Quezon City - Computer Engineering
Course Code:	CPE 313
Code Title:	Advanced Machine Learning and Deep Learning
2nd Semester	AY 2024-2025

Discussion 3.1	Exploring OpenCV
Name	Galapia, Xander Sam E.
Section	CPE32S3
Date Performed:	02/22/2025
Date Submitted:	02/22/2025
Instructor:	Engr. Roman M. Richard

1. Objectives

↳ 1 cell hidden

2. Intended Learning Outcomes (ILOs)

After this activity, the students should be able to:

- Utilize data preparation techniques for images.
- Perform Face Recognition using multiple algorithms.
- Evaluate the performance of different algorithms.

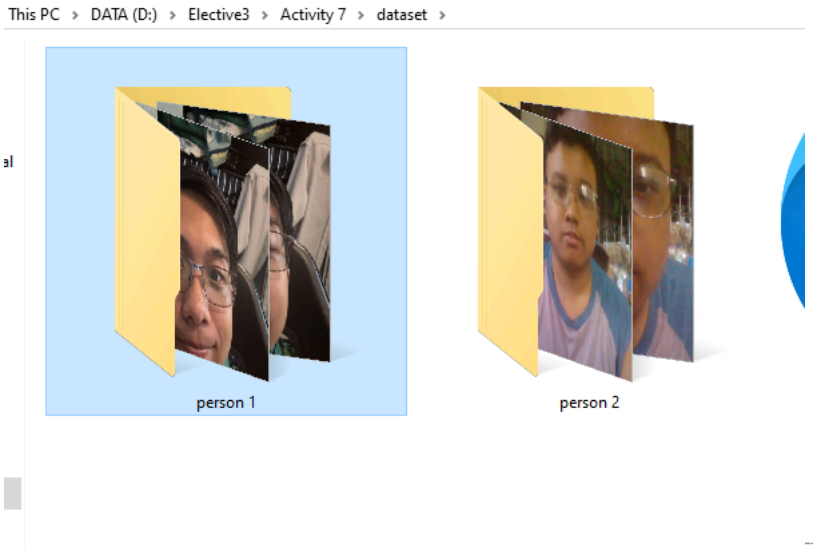
3. Procedures and Outputs

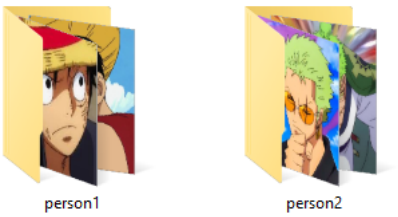
Preparing the training data

Now that we have our data, we need to load these sample pictures into our face recognition algorithms. All face recognition algorithms take two parameters in their `train()` method: an array of images and an array of labels. What do these labels represent? They are the IDs of a certain individual/face so that when face recognition is performed, we not only know the person was recognized but also who—among the many people available in our database—the person is.

To do that, we need to create a comma-separated value (CSV) file, which will contain the path to a sample picture followed by the ID of that person.

Include a Screenshot of Your Dataset Here





Loading the data and recognizing faces

Next up, we need to load these two resources (the array of images and CSV file) into the face recognition algorithm, so it can be trained to recognize our face. To do this, we build a function that reads the CSV file and—for each line of the file—loads the image at the corresponding path into the images array and the ID into the labels array.

```
import numpy as np
import os
import errno
import sys
import cv2

def read_images(dataset, sz=None):
  c = 0
  X, y = [], []

  for dirname, dirnames, filenames in os.walk(dataset):
    for subdirname in dirnames:
      subject_path = os.path.join(dirname, subdirname)
      for filename in os.listdir(subject_path):
        try:
          if(filename == ".directory"):
            continue
          filepath = os.path.join(subject_path, filename)
          im = cv2.imread(os.path.join(subject_path, filename), cv2.IMREAD_GRAYSCALE)

          # Resize the images to the prescribed size
          if (sz is not None):
            im = cv2.resize(im, (200,200))

          X.append(np.asarray(im, dtype=np.uint8))
          y.append(c)

        except IOError as e:
          print(f"I/O Error({e.errno}): {e.strerror}")
        except:
          print("Unexpected error:", sys.exc_info()[0])
          raise
      c = c+1
  return [X, y]


test_data = read_images("dataset")
print(test_data)
```

```
[152, 151, 150, ..., 144, 143, 142]], dtype=uint8), array([[152, 130, 105, ..., 223, 220, 218],
[156, 134, 108, ..., 222, 220, 219],
[157, 135, 108, ..., 219, 219, 219],
...,
[133, 133, 133, ..., 107, 134, 146],
[133, 133, 133, ..., 112, 136, 138],
[133, 133, 133, ..., 119, 143, 139]], dtype=uint8), array([[141, 143, 145, ..., 214, 214, 213],
[140, 142, 143, ..., 214, 213, 213],
[138, 140, 142, ..., 214, 213, 212],
...,
[124, 124, 124, ..., 148, 149, 149],
[123, 123, 123, ..., 150, 151, 151],
[123, 123, 123, ..., 153, 153, 154]], dtype=uint8), array([[108, 108, 108, ..., 213, 214, 213],
[108, 108, 108, ..., 213, 214, 213],
[108, 108, 108, ..., 213, 214, 213],
...,
[ 69,  69,  67, ..., 157, 157, 156],
[ 67,  65,  62, ..., 156, 155, 155],
[ 63,  60,  55, ..., 155, 154, 154]], dtype=uint8), array([[183, 183, 183, ..., 194, 193, 193],
[183, 183, 183, ..., 196, 195, 196],
[183, 183, 183, ..., 196, 197, 199],
...,
[117, 117, 116, ..., 220, 218, 218],
[119, 118, 116, ..., 219, 214, 212],
[120, 119, 117, ..., 220, 212, 210]], dtype=uint8), array([[172, 176, 178, ..., 217, 216, 216],
[173, 177, 177, ..., 217, 216, 215],
[176, 178, 176, ..., 216, 215, 214],
...,
[ 91,  91,  91, ..., 246, 247, 246],
[ 90,  90,  90, ..., 245, 246, 245],
[ 89,  89,  89, ..., 245, 245, 245]], dtype=uint8), array([[189, 188, 187, ..., 217, 214, 210],
[189, 188, 187, ..., 217, 215, 212],
[189, 188, 187, ..., 218, 217, 214],
...,
[166, 168, 173, ..., 178, 175, 173],
[171, 170, 172, ..., 178, 173, 170],
[173, 170, 170, ..., 178, 172, 168]], dtype=uint8)], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]]
```

Question: Run the function above on your generated dataset. Provide an analysis and note all the challenges you have encountered running this code.

You must put a person/sample picture into a solo folder and have it properly named and be put into the file(dataset) for it to work

---

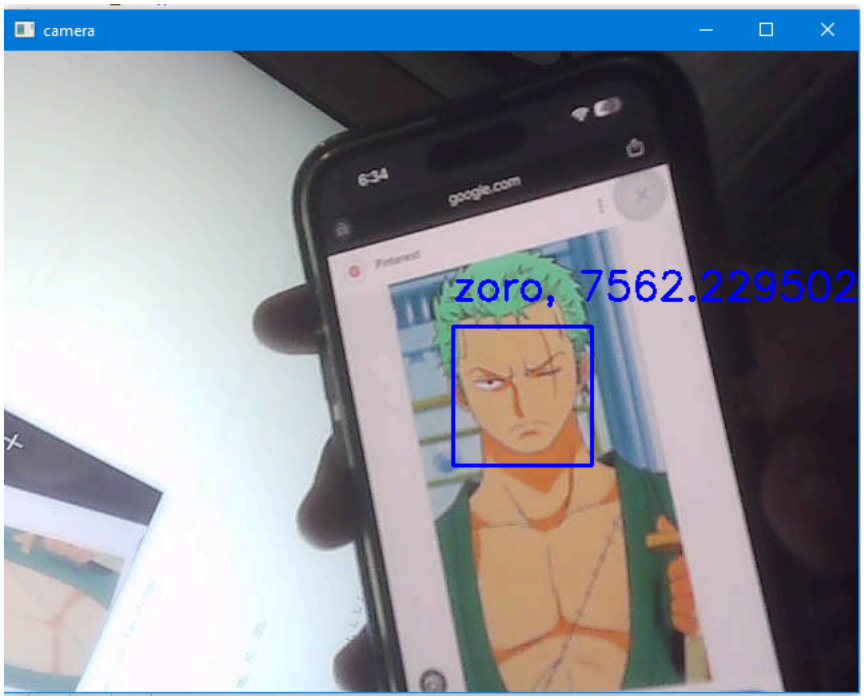
Performing Face Recognition Algorithms

Here is a sample script for testing the Face Recognition Algorithm. In this section, we're going to follow the same process but with different algorithms for face recognitions, namely:

- Eigenface Recognition
- Fisherface Recognition
- Local Binary Pattern Histograms (LBPH) Recognition

```
def face_rec():
    names = ['xander', 'jm']  # Put your names here for faces to recognize

    [X, y] = read_images("dataset", sz=(200, 200))
    y = np.asarray(y, dtype=np.int32)

    model = cv2.face.LBPHFaceRecognizer_create()
    model.train(X, y)

    camera = cv2.VideoCapture(0)
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

    while True:
        ret, img = camera.read()
        if not ret:
            break

        faces = face_cascade.detectMultiScale(img, 1.3, 5)

        for (x, y, w, h) in faces:
            gray = cv2.cvtColor(img[y:y + h, x:x + w], cv2.COLOR_BGR2GRAY)
            roi = cv2.resize(gray, (200, 200), interpolation=cv2.INTER_LINEAR)

            try:
                params = model.predict(roi)
                label = names[params[0]]
```

```
                color = (255, 0, 0)  # Default color (blue)

                if label == 'Friend2':
                    color = (0, 255, 0)  # Green color for Friend2

                cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
                cv2.putText(img, label + ", " + str(params[1]), (x, y - 20), cv2.FONT_HERSHEY_SIMPLEX, 1, color, 2)
            except:
                continue

        cv2.imshow("camera", img)
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break

    camera.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    face_rec()
```

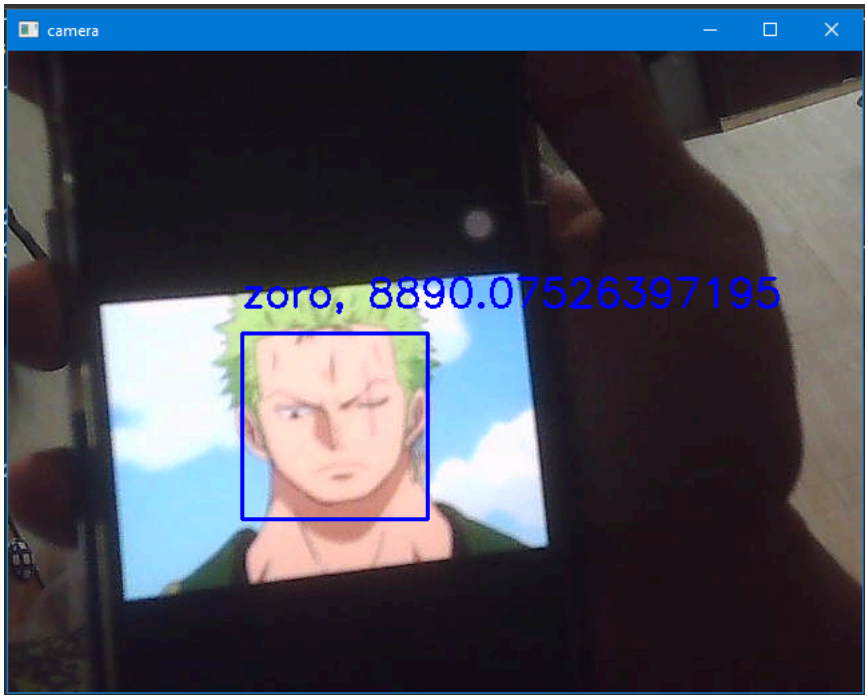


Question: Provide an analysis of the sample script for the process using the Eigenface Model. What is the sample code doing? Are you able to troubleshoot any problems encountered?

Perform the remaining face recognition techniques by using the same (or modified) process from the sample code:

- `model = cv2.face.createFisherFaceRecognizer()`
- `model = cv2.face.createLBPHFaceRecognizer()`

LBPH



4. Supplementary Activity

Your accomplisment of the tasks below contribute to the achievement of ILO1, ILO2, and ILO3 for this module.

Tasks:

1. Create a new dataset for testing, this dataset must include the following:
    - The same person/s that the model has to recognize.
    - Different person/s that the model should not recognize.
2. For each model, perform 20 tests. Document the testing performed and provide observations.
3. Conclude on the performed tests by providing your evaluation of the performance of the models.

```
##Read.py
import numpy as np
import os
import errno
import sys
import cv2

def read_images(dataset, sz=None):
  c = 0
  X, y = [], []

  for dirname, dirnames, filenames in os.walk(dataset):
    for subdirname in dirnames:
      subject_path = os.path.join(dirname, subdirname)
      for filename in os.listdir(subject_path):
        try:
          if(filename == ".directory"):
            continue
          filepath = os.path.join(subject_path, filename)
          im = cv2.imread(os.path.join(subject_path, filename), cv2.IMREAD_GRAYSCALE)

          # Resize the images to the prescribed size
          if (sz is not None):
            im = cv2.resize(im, (200,200))

          X.append(np.asarray(im, dtype=np.uint8))
          y.append(c)

        except IOError as e:
          print(f"I/O Error({e.errno}): {e.strerror}")
        except:
          print("Unexpected error:", sys.exc_info()[0])
          raise
      c = c+1
  return [X, y]
```

```
test_data = read_images("dataset")
print(test_data)


##LBPH.py
from Read import read_images
import numpy as np
import cv2

def face_rec():
    names = ['xander', 'jm']  # Put your names here for faces to recognize

    [X, y] = read_images("dataset", sz=(200, 200))
    y = np.asarray(y, dtype=np.int32)

    model = cv2.face.LBPHFaceRecognizer_create()
    model.train(X, y)

    camera = cv2.VideoCapture(0)
    face_cascade = cv2.CascadeClassifier('D:/Elective3/Activity 7/dataset/haarcascade_frontalface_default.xml')

    while True:
        ret, img = camera.read()
        if not ret:
            break

        faces = face_cascade.detectMultiScale(img, 1.3, 5)

        for (x, y, w, h) in faces:
            gray = cv2.cvtColor(img[y:y + h, x:x + w], cv2.COLOR_BGR2GRAY)
            roi = cv2.resize(gray, (200, 200), interpolation=cv2.INTER_LINEAR)

            try:
                params = model.predict(roi)
                label = names[params[0]]
                confidence = params[1]

                if confidence < 100:
                    color = (0, 255, 0) # Green color for recognized faces
                    cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
                    cv2.putText(img, label + ", " + str(confidence), (x, y - 20), cv2.FONT_HERSHEY_SIMPLEX, 1, color, 2)
                else:
                    color = (0, 0, 255) #Red color for unrecognised faces
                    cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
                    cv2.putText(img, "Unrecognized", (x, y - 20), cv2.FONT_HERSHEY_SIMPLEX, 1, color, 2)

            except Exception as e:
                print(f"Error during prediction: {e}")
                continue

        cv2.imshow("camera", img)
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break

    camera.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    face_rec()


##Fisher.py
from Read import read_images
import numpy as np
import cv2

def face_rec():
    names = ['jm', 'xander']  # Put your names here for faces to recognize

    [X, y] = read_images("dataset", sz=(200, 200))
    y = np.asarray(y, dtype=np.int32)

    model = cv2.face.FisherFaceRecognizer_create()
    model.train(X, y)

    camera = cv2.VideoCapture(0)
    face_cascade = cv2.CascadeClassifier('D:/Elective3/Activity 7/dataset/haarcascade_frontalface_default.xml')

    while True:
        ret, img = camera.read()
        if not ret:
            break
```

```
        faces = face_cascade.detectMultiScale(img, 1.3, 5)

        for (x, y, w, h) in faces:
            gray = cv2.cvtColor(img[y:y + h, x:x + w], cv2.COLOR_BGR2GRAY)
            roi = cv2.resize(gray, (200, 200), interpolation=cv2.INTER_LINEAR)

            try:
                params = model.predict(roi)
                label = names[params[0]]
                confidence = params[1]

                if confidence < 200:
                    color = (0, 255, 0) # Green color for recognized faces
                    cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
                    cv2.putText(img, label + ", " + str(confidence), (x, y - 20), cv2.FONT_HERSHEY_SIMPLEX, 1, color, 2)
                else:
                    color = (0, 0, 255) #Red color for unrecognised faces
                    cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
                    cv2.putText(img, "Unrecognized", (x, y - 20), cv2.FONT_HERSHEY_SIMPLEX, 1, color, 2)

            except Exception as e:
                print(f"Error during prediction: {e}")
                continue

        cv2.imshow("camera", img)
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break

    camera.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    face_rec()


#Eigen.py
from Read import read_images
import numpy as np
import cv2

def face_rec():
    names = ['jm', 'xander']  # Put your names here for faces to recognize

    [X, y] = read_images("dataset", sz=(200, 200))
    y = np.asarray(y, dtype=np.int32)

    model = cv2.face.EigenFaceRecognizer_create()
    model.train(X, y)

    camera = cv2.VideoCapture(0)
    face_cascade = cv2.CascadeClassifier('D:/Elective3/Activity 7/dataset/haarcascade_frontalface_default.xml')

    while True:
        ret, img = camera.read()
        if not ret:
            break

        faces = face_cascade.detectMultiScale(img, 1.3, 5)

        for (x, y, w, h) in faces:
            gray = cv2.cvtColor(img[y:y + h, x:x + w], cv2.COLOR_BGR2GRAY)
            roi = cv2.resize(gray, (200, 200), interpolation=cv2.INTER_LINEAR)

            try:
                params = model.predict(roi)
                label = names[params[0]]
                confidence = params[1]

                if confidence < 10:
                    color = (0, 255, 0) # Green color for recognized faces
                    cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
                    cv2.putText(img, label + ", " + str(confidence), (x, y - 20), cv2.FONT_HERSHEY_SIMPLEX, 1, color, 2)
                else:
                    color = (0, 0, 255) #Red color for unrecognised faces
                    cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
                    cv2.putText(img, "Unrecognized", (x, y - 20), cv2.FONT_HERSHEY_SIMPLEX, 1, color, 2)

            except Exception as e:
                print(f"Error during prediction: {e}")
                continue

        cv2.imshow("camera", img)
        if cv2.waitKey(1) & 0xFF == ord("q"):
```

```
            break

    camera.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    face_rec()
```

Fisher

↳ 20 cells hidden

Eigen

↳ 11 cells hidden

LBPH

↳ 20 cells hidden

5. Summary, Conclusions and Lessons Learned

In this activity we used the same type of topic as Activity 6 where to use the camera, but here we are tasked to create a face recognition where we add a pictures for the database. The Procedure pictures is from an anime where it went wrong recognizing zoro 1 time, while for the supplementary activity was using my photo for database and searching for random pictures and checking if it recognizes the random person.

The fisher detects both my face and jm but it detects jm wrong as there is no jm picture included in the 20 sample but it also detected some errors in those pictures.

The Eigen recognizes all the people as unknown even if the camera is on me so I only put 11 pictures/samples

The LBPH have the highest wrong recognition of me but it didn't detected jm as there is no picture of jm in the samples but

Overall, the Fisher is better compared to LBPH and Eigen and by putting my name first in the list or jm it gives a better result. There is probably need for more samples for better accuracy.

Proprietary Clause