Technological Institute of the Philippines	Quezon City - Computer Engineering
Course Code:	CPE 313
Code Title:	Advance Machine Learning and Deep Learning
2nd Semester	AY 2024-2025
ACTIVITY NO. 7	**Performing Face Recognition**
Name	Alorro, Jay-ann
Section	CPE32S3
Date Performed:	02/20/25
Date Submitted:	02/21/25
Instructor:	Dr. Jonathan V. Taylar / Engr. Verlyn V. Nojor / Engr. Roman M. Richard

1. Objectives

This activity aims to enable students to perform data preparation and face recognition on their own generated dataset.

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

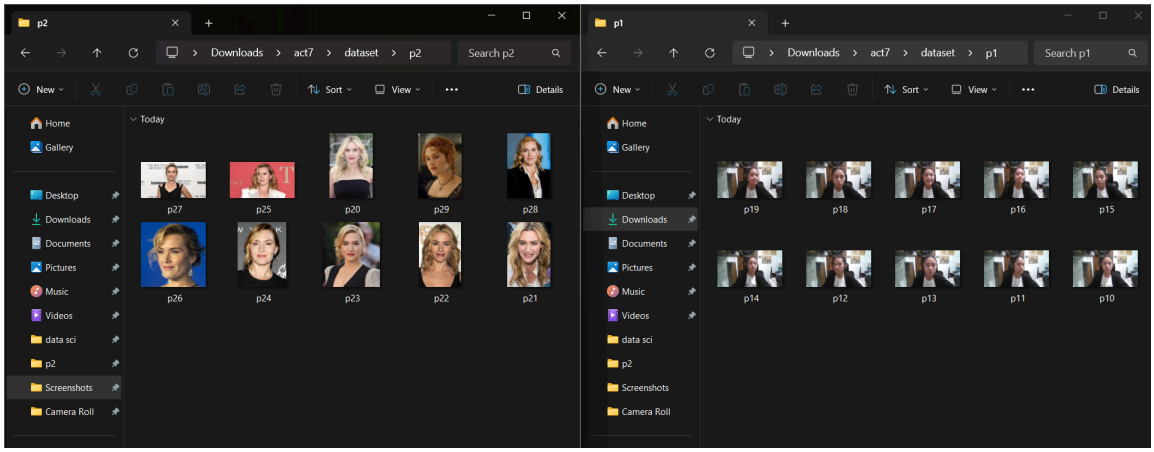
Preparing the training data

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.

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

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.

In [1]:
```
import numpy as np
import os
import errno
import sys
import cv2

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

    for dirname, dirnames, filenames in os.walk(path):
        for subdirname in dirnames:
            subject_path = os.path.join(dirname, subdirname)
            for filename in os.listdir(subject_path):
                try:
                    if filename.startswith('.'):  # skip hidden/system files
                        continue

                    filepath = os.path.join(subject_path, filename)
                    im = cv2.imread(filepath, cv2.IMREAD_GRAYSCALE)

                    # If imread failed, skip
                    is None:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
                        print(f"Skipping file {filepath} - not a valid image
                        continue

                    # 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 Exception as e:
                    print("Unexpected error:", e)
                    raise
            c += 1
    return [X, y]
```

In [6]:
```
dataset_path = r"C:\Users\Jay-ann Alorro\Downloads\dataset"
img_dataset = read_images(dataset_path)
img_dataset
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Out[6]:
```
[[array([[122, 126, 132, ..., 140, 140, 140],
         [126, 131, 134, ..., 139, 140, 140],
         [129, 135, 136, ..., 139, 139, 140],
         ...,
         [ 17,  17,  17, ...,  58,  65,  69],
         [ 17,  17,  17, ...,  60,  69,  75],
         [ 17,  17,  18, ...,  63,  74,  81]], dtype=uint8),
  array([[ 59,  58,  59, ...,  44,  50,  64],
         [ 56,  56,  59, ...,  60,  63,  71],
         [ 55,  56,  60, ...,  76,  76,  77],
         ...,
         [ 23,  23,  23, ..., 129, 130, 133],
         [ 23,  23,  23, ..., 126, 133, 132],
         [ 22,  23,  23, ..., 126, 135, 130]], dtype=uint8),
  array([[ 52,  69,  85, ...,  76,  75,  74],
         [ 61,  69,  79, ...,  75,  75,  75],
         [ 71,  70,  73, ...,  75,  74,  74],
         ...,
         [ 79,  74,  68, ..., 234, 239, 242],
         [ 71,  64,  62, ..., 239, 242, 243],
         [ 61,  56,  61, ..., 238, 241, 243]], dtype=uint8),
  array([[93, 94, 95, ..., 75, 75, 75],
         [94, 94, 94, ..., 79, 79, 79],
         [93, 94, 92, ..., 76, 76, 76],
         ...,
         [ 8,  8,  8, ..., 27, 22, 17],
         [ 8,  8,  8, ..., 21, 18, 15],
         [ 8,  8,  8, ..., 21, 20, 19]], dtype=uint8),
  array([[ 51,  61,  66, ...,  65,  55,  47],
         [ 50,  58,  61, ...,  57,  55,  52],
         [ 54,  58,  60, ...,  57,  59,  60],
         ...,
         [ 19,  18,  17, ..., 122, 159, 192],
         [ 21,  21,  19, ..., 118, 142, 183],
         [ 20,  20,  21, ..., 120, 138, 187]], dtype=uint8),
  array([[ 95,  94,  94, ..., 116, 115, 114],
         [ 95,  94,  94, ..., 116, 115, 114],
         [ 95,  94,  94, ..., 115, 114, 113],
         ...,
         [128, 127, 123, ..., 100,  92,  87],
         [126, 125, 123, ..., 100,  92,  87],
         [125, 123, 123, ..., 100,  92,  87]], dtype=uint8),
  array([[ 75,  73,  70, ...,   5,   4,   3],
         [ 79,  80,  77, ...,   5,   4,   4],
         [ 73,  77,  79, ...,  11,   9,   6],
         ...,
         [213, 214, 216, ...,  57,  72,  85],
         [215, 215, 216, ...,  62,  77,  84],
         [217, 218, 219, ...,  58,  86, 102]], dtype=uint8),
  array([[231, 231, 231, ..., 197, 196, 196],
         [231, 231, 231, ..., 197, 197, 196],
         [231, 231, 231, ..., 198, 197, 197],
         ...,
         [ 13,  13,  13, ...,   8,  10,  12],
         [ 13,  13,  13, ...,   8,   9,  11],
         [ 13,  13,  13, ...,   7,   9,  10]], dtype=uint8),
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
 array([[ 70,  51,  57, ...,  86,  84,  94],
        [ 51,  34,  40, ...,  77,  75,  85],
        [ 57,  40,  45, ...,  79,  78,  89],
        ...,
        [ 40,  26,  24, ..., 222, 231, 232],
        [ 40,  26,  24, ..., 233, 241, 242],
        [ 40,  26,  24, ..., 223, 228, 234]], dtype=uint8),
 array([[ 99, 103, 107, ..., 101, 101, 101],
        [ 98, 102, 106, ..., 100, 101, 101],
        [ 98, 101, 104, ...,  99, 101, 102],
        ...,
        [ 25,  27,  27, ...,  14,  15,  16],
        [ 24,  26,  27, ...,  12,  13,  15],
        [ 23,  24,  25, ...,  11,  12,  13]], dtype=uint8),
 array([[130, 128, 126, ..., 128, 128, 129],
        [126, 125, 123, ..., 129, 130, 131],
        [124, 123, 123, ..., 130, 131, 132],
        ...,
        [123, 123, 123, ..., 133, 135, 135],
        [123, 123, 123, ..., 133, 135, 135],
        [123, 123, 123, ..., 133, 135, 135]], dtype=uint8),
 array([[66, 66, 67, ..., 73, 73, 73],
        [63, 63, 63, ..., 67, 67, 67],
        [59, 59, 59, ..., 61, 61, 61],
        ...,
        [62, 63, 64, ..., 74, 71, 68],
        [67, 68, 69, ..., 76, 73, 69],
        [70, 71, 72, ..., 78, 75, 71]], dtype=uint8),
 array([[245, 245, 245, ..., 237, 238, 238],
        [245, 245, 245, ..., 237, 238, 238],
        [245, 245, 245, ..., 237, 238, 238],
        ...,
        [233, 216, 222, ..., 220, 220, 220],
        [230, 248, 255, ..., 215, 214, 215],
        [255, 206, 101, ..., 234, 234, 235]], dtype=uint8),
 array([[96, 96, 96, ..., 85, 85, 93],
        [96, 96, 96, ..., 88, 88, 91],
        [96, 96, 96, ..., 91, 91, 88],
        ...,
        [ 2,  2,  2, ...,  3,  3,  3],
        [ 2,  2,  2, ...,  3,  3,  3],
        [ 2,  2,  2, ...,  3,  3,  3]], dtype=uint8),
 array([[174, 173, 173, ...,  60,  60,  60],
        [173, 173, 174, ...,  61,  60,  60],
        [175, 175, 173, ...,  61,  61,  61],
        ...,
        [  2,   2,   2, ...,   9,   9,   9],
        [  2,   2,   2, ...,   5,   6,   6],
        [  2,   2,   2, ...,   2,   2,   2]], dtype=uint8)],
[0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]
```

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

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

- Some challenges that I have encountered in this code is that the file format of the images need to be the same and I didn't realized it until I made some modification in the code which tells me if there is an invalid image or not.

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

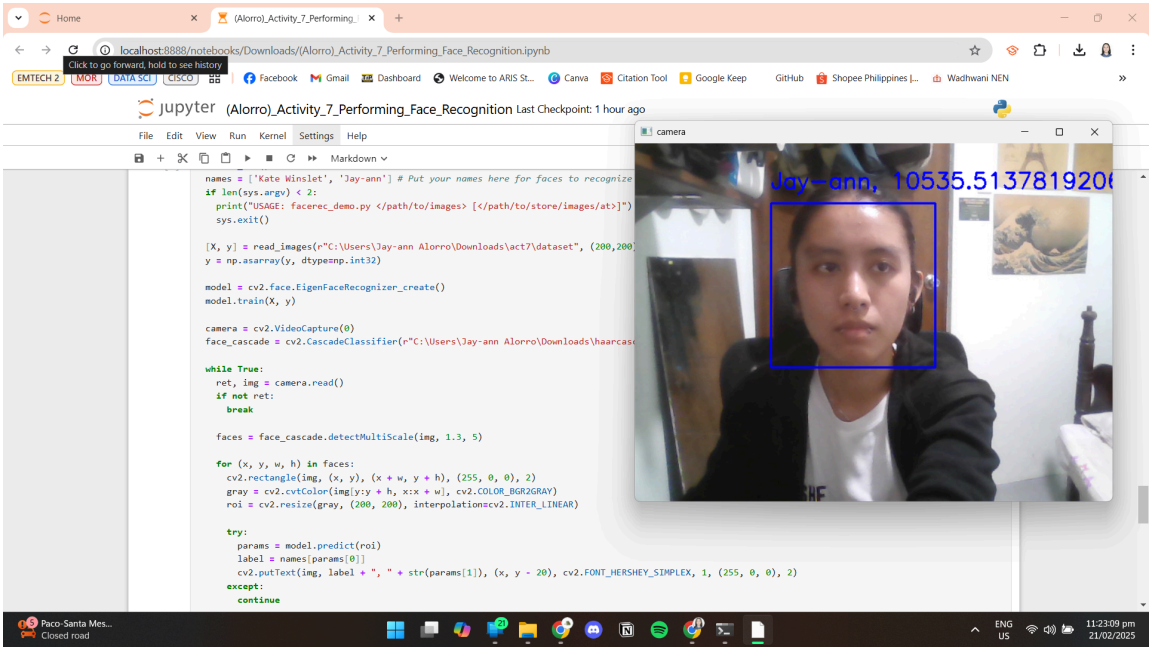
In [ ]:
```
pip install opencv-contrib-python --user
```

```
Requirement already satisfied: opencv-contrib-python in c:\users\jay-ann alo
rro\appdata\roaming\python\python312\site-packages (4.11.0.86)
Requirement already satisfied: numpy>=1.21.2 in c:\users\jay-ann alorro\anac
onda3\lib\site-packages (from opencv-contrib-python) (1.26.4)
Note: you may need to restart the kernel to use updated packages.
```

In [17]:
```
def face_rec(filepath):
  names = ['Jay-ann', 'Mama'] # Put your names here for faces to recognize
  if len(sys.argv) < 2:
    print("USAGE: facerec_demo.py </path/to/images> [</path/to/store/images/
    sys.exit()

  [X, y] = read_images(filepath, (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(r"C:\Users\Jay-ann Alorro\Downloads\h

  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:
      cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
      gray = cv2.cvtColor(img[y:y + h, x:x + w], cv2.COLOR_BGR2GRAY)
      roi = cv2.resize(gray, (200, 200), interpolation=cv2.INTER_LINEAR)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
      try:
        params = model.predict(roi)
        label = names[params[0]]
        cv2.putText(img, label + ", " + str(params[1]), (x, y - 20), cv2.FON
      except:
        continue

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

  camera.release()
  cv2.destroyAllWindows()

if __name__ == "__main__":
    face_rec(r"C:\Users\Jay-ann Alorro\Downloads\dataset")
```



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?

- In the sample script, it demonstrate a face recognition using Eigenface model. It loads a dataset of grayscale images and resizes them to a uniform size. It then trains an EigenFaceRecognition model on the images. The script then opens a webcam that detects faces in real-time and classify its identity.
- The troubleshooting that I did was to ensure that the images are read correctly, having the right version of OpenCV and adapting the command-line input code when running it in a Jupyter Notebook.

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

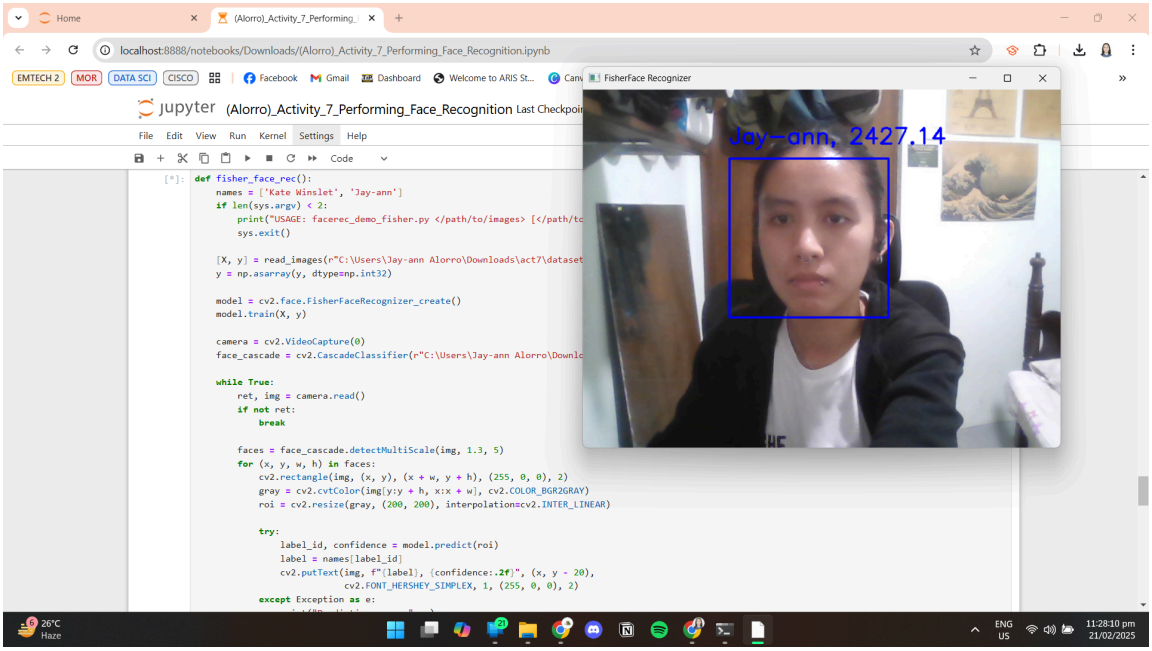
Perform the remaining face recognition techniques by using the same (or modified) process from the sample code:

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

In [16]:

```
def fisher_face_rec(filepath):
    names = ['Jay-ann', 'Mama']
    if len(sys.argv) < 2:
        print("USAGE: facerec_demo_fisher.py </path/to/images> [</path/to/st
        sys.exit()

    [X, y] = read_images(filepath, (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(r"C:\Users\Jay-ann Alorro\Downloads

    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:
            cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
            gray = cv2.cvtColor(img[y:y + h, x:x + w], cv2.COLOR_BGR2GRAY)
            roi = cv2.resize(gray, (200, 200), interpolation=cv2.INTER_LINEA

            try:
                label_id, confidence = model.predict(roi)
                label = names[label_id]
                cv2.putText(img, f"{label}, {confidence:.2f}", (x, y - 20),
                            cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)
            except Exception as e:
                print("Prediction error:", e)
                continue

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

    camera.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    fisher_face_rec(r"C:\Users\Jay-ann Alorro\Downloads\dataset")
```

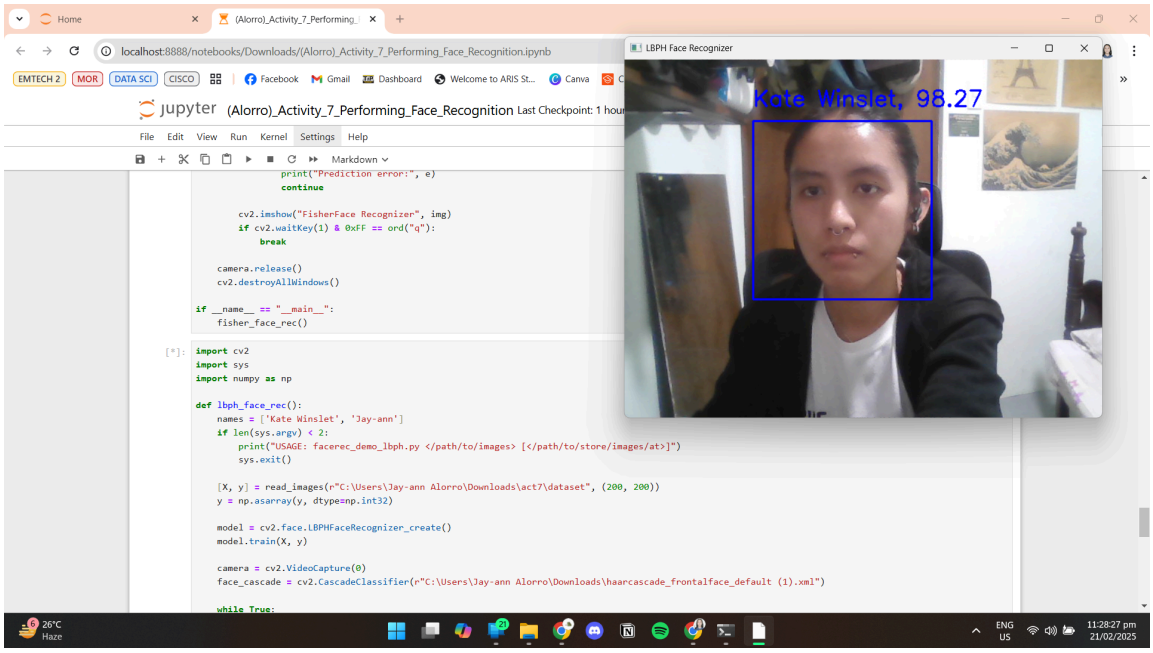
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [15]:
```
import cv2
import sys
import numpy as np

def lbph_face_rec(filepath):
    names = ['Jay-ann', 'Mama']
    if len(sys.argv) < 2:
        print("USAGE: facerec_demo_lbph.py </path/to/images> [</path/to/stor
        sys.exit()

    [X, y] = read_images(filepath, (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(r"C:\Users\Jay-ann Alorro\Downloads

    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:
            cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
            gray = cv2.cvtColor(img[y:y + h, x:x + w], cv2.COLOR_BGR2GRAY)
            roi = cv2.resize(gray, (200, 200), interpolation=cv2.INTER_LINEA

            try:
                label_id, confidence = model.predict(roi)
                label = names[label_id]
                cv2.putText(img, f"{label}, {confidence:.2f}", (x, y - 20),
                            cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
            except Exception as e:
                print("Prediction error:", e)
                continue

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

    camera.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    lbph_face_rec(r"C:\Users\Jay-ann Alorro\Downloads\dataset")
```



Question: The `predict()` method returns a two-element array. Provide your analysis of the two returned values and their important ince this application.

- The first element is the predicted label (an integer index mapping to a person's name), and the second is the confidence score. A lower confidence value means a closer match, indicating higher reliability of the recognition.

4. Supplementary Activity

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

Tasks:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

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.

In [9]:
```
# new dataset
dataset2 = r"C:\Users\Jay-ann Alorro\Downloads\dataset2"
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [14]:
```
def face_rec_supple(filepath):
  names = ['Recognized', 'Unrecognized'] # Put your names here for faces to
  if len(sys.argv) < 2:
    print("USAGE: facerec_demo.py </path/to/images> [</path/to/store/images/
    sys.exit()

  [X, y] = read_images(filepath, (200,200))
  y = np.asarray(y, dtype=np.int32)

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

  camera = cv2.VideoCapture(0)
  face_cascade = cv2.CascadeClassifier(r"C:\Users\Jay-ann Alorro\Downloads\h

  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:
      cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
      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]]
        cv2.putText(img, label + ", " + str(params[1]), (x, y - 20), cv2.FON
      except:
        continue

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

  camera.release()
  cv2.destroyAllWindows()
```

In [15]:
```
face_rec_supple(dataset2)
```

Upon testing, Local Binary Pattern Histograms (LBPH) Recognition performed the best out of all the models trained. The three model's performance variation is not that significant, Eigenface and Fisherface actually had the same result. However, the model's performance could vary because the dataset might be limited so adding more images might improve all of their performance.

Result Summary:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

- Eigenface Recognition: 11/20
- Fisherface Recognition: 11/20
- Local Binary Pattern Histograms (LBPH) Recognition: 13/20

Testing Results:
https://docs.google.com/document/d/1VCPOwGyIUbks7R9gPb3Ir7qxbXdcNPo6ks2rcc usp=sharing

5. Summary, Conclusions and Lessons Learned

This activity focuses on face recognition which guided me through data preparation and implementing multiple face recognition algorithms. It involves training a model with a data of facial images and evaluating other different recognition techniques. I have realize that facial recognition is a crucial aspect of computer vision and it requires proper dataset preparation and algorithm selection.

In conclusion, I have learned that data preparation is essential for accurate face recognition. There are different algorithms and all of them can vary in performance. Facial recognition builds upon face detection but requires deeper feature analysis.

Proprietary Clause

Property of the Technological Institute of the Philippines (T.I.P.). No part of the materials made and uploaded in this learning management system by T.I.P. may be copied, photographed, printed, reproduced, shared, transmitted, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior consent of T.I.P.

This notebook was converted with convert.ploomber.io

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js