

Artificial Intelligence

Pada Pengenalan Wajah

CLASSROOM DEV TEAM

August 30, 2022

Contents

1 Pengantar	1
1.1 Artificial Intelligence Pada Pengenalan Wajah	1
1.2 Face Detection	2
1.3 Face Recognition	2
2 Model Algoritma Face Recognition Pada OpenCV	4
2.1 Eigenface	4
2.2 Local Binary Pattern Histogram(LBPH)	5
2.3 Fisherfaces	6
3 Implementasi Sistem	7
3.1 Proses Instalasi	7
3.1.1 Instalasi Python	7
3.1.2 Instalasi OpenCV	10
3.2 Pembuatan Sistem Face Detection	11
3.3 Pembuatan Sistem Face Recognition	13
3.3.1 Face Recognition dengan video	13
3.3.2 Face Recognition dengan kamera	20

List of Figures

2.1	Proses algoritma fisherface	6
3.1	Website resmi python	7
3.2	Pilihan Versi Python	8
3.3	Instalator python	8
3.4	Pilihan fitur	9
3.5	Pilihan lanjutan dan penyesuaian lokasi	9
3.6	Instalasi openCV	10
3.7	Cek openCV pada python IDLE	10
3.8	Cek openCV pada CMD	10
3.9	Memasukan library openCV	11
3.10	Memasukan library openCV	11
3.11	Memasukan video	11
3.12	Membuka video dan ubah warna cira	11
3.13	Deteksi wajah	12
3.14	Kode bingkai wajah	12
3.15	Kode deteksi wajah	12
3.16	Hasil deteksi wajah	12
3.17	Memasukan library openCV	13
3.18	Proses face detection dengan masukan video	13
3.19	Tambah variabel	13
3.20	Membaca video dan merubah warna citra	14
3.21	Penyimpanan dataset	14
3.22	Mengatur jumlah gambar yang diambil	14
3.23	Kode pengambilan dataset	15
3.24	Pengambilan dataset	15
3.25	Pengambilan dataset	16
3.26	Pengambilan dataset	16
3.27	Pengambilan dataset	16
3.28	Pengambilan dataset	17
3.29	Pengambilan dataset	17
3.30	Pengambilan dataset	17
3.31	Import Library	17

3.32 Pengenalan wajah	18
3.33 Deteksi wajah	18
3.34 Font dan ID	18
3.35 Array nama	18
3.36 Input video/kamera serta mengatur ukuran frame	18
3.37 Membaca dan perubahan citra	19
3.38 Mengenali kecocokan data	19
3.39 Menampilkan kamera atau video serta akhir program	19
3.40 Hasil recognition video	20
3.41 Memasukan library	20
3.42 Proses face detection dengan masukan kamera	20
3.43 Masukan id serta nama untuk label dataset	21
3.44 Proses face detection dan pengambilan dataset	21
3.45 Kumpulan label direktori dataset	21
3.46 Isi direktori salah satu label dataset	21
3.47 Import library	22
3.48 Pengambilan dataset	22
3.49 Variabel	22
3.50 Menelusuri folder dataset	23
3.51 Menelusuri folder dataset dan konversi gambar	23
3.52 Melampirkan nama	23
3.53 Simpan hasil training	24
3.54 Program training	24
3.55 Import library	25
3.56 Model trained	25
3.57 Pengaturan kamera	25
3.58 Import library	26
3.59 Hasil face recognition jarak sekitar 30 cm dengan kamera	26
3.60 Hasil face recognition jarak sekitar 1 meter dengan kamera	27

1

Pengantar

1.1 Artificial Intelligence Pada Pengenalan Wajah

Dilansir dari Stanford Computer science, Artificial Intelligence(AI) atau kecerdasan buatan adalah ilmu dan rekayasa pembuatan mesin cerdas, melibatkan mekanisme untuk menjalankan suatu tugas menggunakan komputer. Sehingga artificial intelligence merupakan sebuah teknologi yang memungkinkan sistem komputer, perangkat lunak, program dan robot untuk “berpikir” secara cerdas layaknya manusia. Kecerdasan buatan suatu mesin dibuat oleh manusia melalui algoritma pemrograman yang kompleks.¹

Secara garis besar, AI dapat melakukan salah satu dari keempat faktor berikut:

- a. *Acting Humanly* , sistem bertindak layaknya manusia.
- b. *Thinking Humanly* , sistem dapat berpikir seperti manusia.
- c. *Think Rationally* , sistem dapat berpikir secara rasional.
- d. *Act Rationally* , sistem mampu bertindak secara rasional.

Pengenalan dan identifikasi wajah merupakan contoh sistem penerapan konsep Artificial Intelligence menggunakan biometrik wajah yang terus berkembang pada bidang *computer vision*. Kecerdasan buatan ini digunakan secara *real-time* untuk menangkap dan mengenali wajah seseorang pada kamera.

Computer Vision adalah bagaimana komputer/mesin dapat melihat, teknik computer vision mampu memvisualisasikan data menganalisa berupa gambar image atau dalam bentuk video. Tujuan utama dari Computer Vision adalah

¹Mustofa, Zaenal. *Artificial Intelligence (AI): Pengertian, Perkembangan, Cara Kerja, Dan Dampaknya*.Universitas STEKOM

agar komputer atau mesin dapat meniru kemampuan perceptual mata manusia dan otak, atau bahkan dapat mengunggulinya untuk tujuan tertentu.²

1.2 Face Detection

Face Detection atau pengenalan wajah merupakan sebuah teknologi untuk menangkap wajah seseorang pada kamera yang menjadi tahap awal dalam sistem pengenalan wajah (*Face Recognition*) yang digunakan dalam identifikasi biometrik. Deteksi wajah juga dapat digunakan untuk pencarian atau pengindeksan data wajah dari citra atau video yang berisi wajah dengan berbagai ukuran, posisi, dan latar belakang.³

Pembuatan pendekripsi wajah ini dapat dibuat menggunakan openCV yang merupakan aplikasi perangkat lunak untuk pengolahan citra dinamis secara *real-time*, selain itu openCV juga banyak mendukung bahasa pemrograman diantaranya C++, C, python, dan java. Pada pembahasan kali ini, penjelasan mengenai proses pembuatan deteksi wajah akan menggunakan openCV dengan bahasa pemrograman python. Proses deteksi objek maupun wajah dapat menggunakan metode algoritma Haar Cascade Classifier.

Algoritma Haar Cascade Classifier merupakan salah satu algoritma yang digunakan untuk mendekripsi sebuah wajah dengan cepat dan *real-time* sebuah benda termasuk wajah manusia. Metode ini menggunakan haar-like features dimana perlu dilakukan training terlebih dahulu untuk mendapatkan suatu pohon keputusan dengan nama cascade classifier sebagai penentu apakah ada obyek atau tidak dalam frame yang diproses, dengan mengelompokkan fitur-fitur pada gambaran wajah berdasarkan sisi yang terang dan sisi yang gelap. Adanya fitur Haar ditentukan dengan cara mengurangi rata-rata piksel pada daerah gelap dari rata-rata piksel pada daerah terang⁴

1.3 Face Recognition

Face recognition adalah sebuah teknologi yang mampu untuk mengidentifikasi dan mengkonfirmasi identitas seseorang menggunakan wajah mereka. Face recognition menjadi salah satu sistem identifikasi biometrik yang paling baik dalam mengidentifikasi seseorang dengan fitur-fitur khusus pada tubuh maupun DNA yang menjadi pembeda antara satu orang dengan orang lainnya.

²Wibowo, Ari. *Implementasi Teknik Computer Vision*. Universitas Widyaatama

³NUGROHO, Setyo, Drs. Agus Hardjoko, MSc.,PhD. *Sistem pendekripsi wajah manusia pada citra digital*, Universitas Gajah mada, diakses dari <http://etd.repository.ugm.ac.id/pelitian/detail/23416>

⁴Suhepy Abidin. *Dekripsi Wajah Menggunakan Metode Haar Cascade Classifier Berbasis Webcam Pada Matlab*. Jurusan Teknik Elektro, Politeknik Negeri Ujung Pandang

Menurut US Government Accountability Office, ada 4 komponen yang dibutuhkan untuk melakukan face recognition, yaitu: kamera, faceprint, Database dan terakhir Algoritme untuk membandingkan faceprint dari wajah target dengan faceprint dalam database.⁵ Setelah terpenuhinya komponen tersebut, dilakukan beberapa tahap untuk melakukan face recognition.

Menurur Haisong Gu, Qiang Ji, dan Zhiwei Zu (2002), pengenalan wajah umumnya melalui 3 tahapan untuk mendapatkan hasil.

1. *Face Detection*, dilakukan untuk mendeteksi adanya atau tidak pada frame yang dibaca sistem. Pada proses ini menggunakan metode *Haar-cascade Classifier*
2. *Facial Expression Information Extraction*, dilakukan pada wajah yang sudah terdeteksi untuk mengekstraksi informasi penting yang akan didapatkan dari wajah untuk membedakan wajah. Fitur atau bagian wajah yang telah diekstraksi nantinya akan digunakan untuk pencocokan wajah.
3. *Expression Classification*, merupakan proses akhir dalam pengenalan wajah, dimana sistem melakukan pencocokan dari masukan wajah dengan data yang ada pada database

Ada beberapa algoritma untuk melakukan pencocokan pada proses pengenalan wajah yang disediakan oleh openCV. Local Binary Pattern Histogram (LBPH) adalah salah satu dari tiga algoritma pengenalan wajah bawaan pada library OpenCV antara lain Eigenface, Fisherfaces, dan LBPH. Dibandingkan dengan kedua algoritma tersebut, LBPH tidak hanya dapat mengenali muka depan, tetapi juga mengenali muka samping yang lebih fleksibel. Metode ini bekerja dengan membandingkan dan mencocokan histogram yang sudah diekstraksi dengan citra wajah yang sudah ada pada database/dataset.

⁵Putri, Monica. *Cara Kerja Face Recognition*. Universitas Binus

2

Model Algoritma Face Recognition Pada OpenCV

Ada beberapa model algoritma yang sudah disediakan openCV untuk melakukan pengenalan wajah, diantaranya adalah Eigenface, Fisherfaces, dan *Local binary Pattern Histogram*(LBPH). Berikut penjelasan untuk perbedaan dari ketiga model algoritma yang disediakan oleh openCV.

2.1 Eigenface

Eigenface adalah salah satu algoritma pengenalan wajah yang berdasarkan pada Principle Component Analysis (PCA) yang dikembangkan di MIT. Algoritma EigenFace secara keseluruhan cukup sederhana, Training Image direpresentasikan dalam sebuah vektor flat (gabungan vektor) dan digabung bersama-sama menjadi sebuah matriks tunggal. Eigen Vector kemudian diekstraksi dan disimpan dalam file temporary atau database. Training image kemudian diproyeksikan dalam feature space yang dinamai face space yang ditentukan oleh eigen vektor(Mukti, 2008).¹

Principal Component Analysis (PCA) atau disebut juga transformasi KarhunenLoeve adalah teknik yang digunakan untuk menyederhanakan suatu data, dengan cara mentransformasi linear sehingga terbentuk sistem koordinat baru dengan variansi maksimum. PCA dapat digunakan untuk mereduksi dimensi suatu data tanpa mengurangi karakteristik data tersebut secara signifikan (Cahyadi, 2007: 93) ²

¹Alam, R.G guntur, dkk. *IMPLEMENTASI ALGORITMA EIGENFACE UNTUK FACE RECOGNITION PADA OBJEK FOTO ID CARD*.Telematik : Vol 7, No 2, April 2015.

²Firliana, Lina, dkk.*Implementasi Principal Component Analysis (PCA) Untuk Pengenalan Wajah Manusia*. Universitas Nusantara

Berikut proses langkah-langkah pengenalan wajah menggunakan model algoritma Eigenface.

1. Penyusunan flat vektor matriks
2. Mengambil nilai tengah dari kumpulan matriks
3. Menghitung selisih antara nilai matriks *training image* dengan nilai tengah
4. Menghitung nilai matriks kovarian
5. Menghitung nilai *eigenvalue* dan *eigenvector*
6. Menghitung nilai *eigenface*
7. Proses indentifikasi wajah

2.2 Local Binary Pattern Histogram(LBPH)

Local Binary Patter(LBP) adalah salah satu dari metode yang terkenal dalam mengenali sebuah objek. Dalam hal ini, cara yang digunakan adalah membedakan objek dengan background. Local Binary Patter Histogram(LBPH) adalah sebuah kombinasi algoritma antara LBP dengan Histogram of Oriented Gradients(HOG)

2.3 Fisherfaces

Algoritma Fisherface ini merupakan gabungan dari metode Principal Component Analys (PCA) dengan Fisher's Linear Discriminant (FLD). FLD merupakan salah satu contoh metode class specific, karena prinsip dasar metode ini berusaha untuk membentuk jarak (scatter) antar kelas dan intra kelas sehingga dapat menghasilkan klasifikasi yang lebih baik, dan mereduksi dimensi yang didapat dari perhitungan PCA. Semakin besar rasio antar kelas, vector ciri yang dihasilkan semakin tidak sensitive terhadap perubahan ekspresi maupun perubahan pencahayaan, sehingga dapat menghasilkan klasifikasi yang lebih baik.³

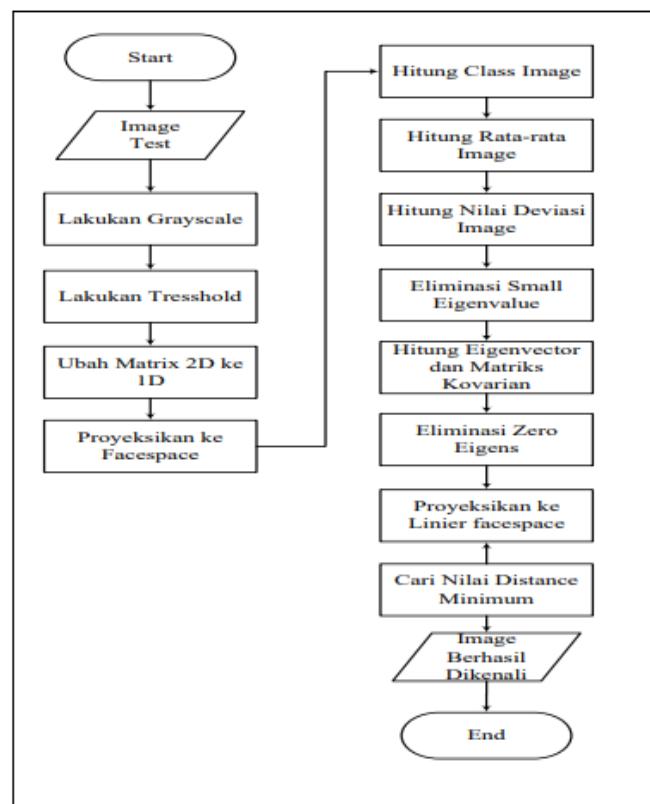


Figure 2.1: Proses algoritma fisherface

³ Amalia, Nurul. Perbandingan Algoritma Fisherface dan Algoritma Local Binary Pattern Untuk Pengenalan Wajah. Universitas Budi Darma, Medan

3

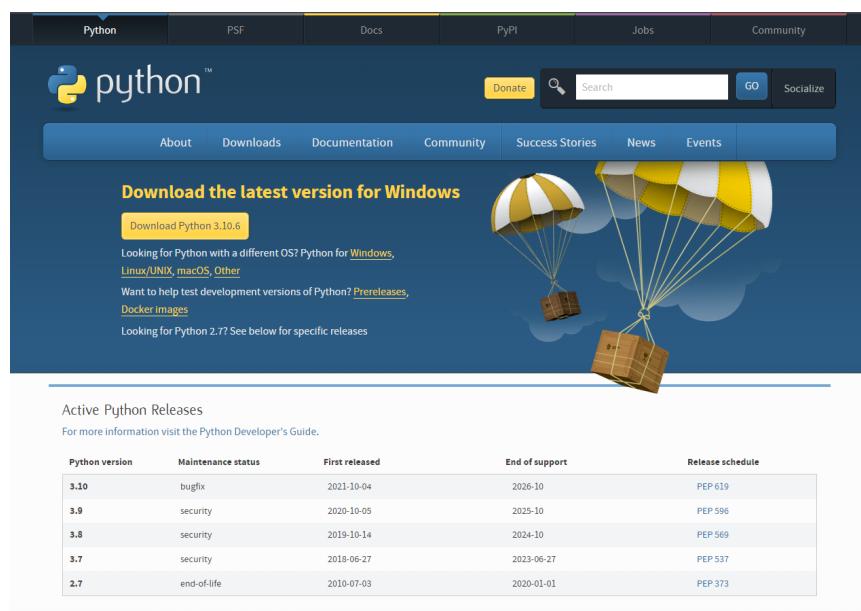
Implementasi Sistem

3.1 Proses Instalasi

Pembuatan sistem face detection dan face recognition akan menggunakan bahasa pemrograman Python dan *library* openCV. Berikut adalah cara instalasi Python serta library openCV yang akan digunakan.

3.1.1 Instalasi Python

Instalator Python dapat didownload pada website resmi python <https://www.python.org/downloads>



Python version	Maintenance status	First released	End of support	Release schedule
3.10	bugfix	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537
2.7	end-of-life	2010-07-03	2020-01-01	PEP 373

Figure 3.1: Website resmi python

Download instalator versi terbaru dari python atau sesuaikan dengan kebutuhan penggunaan

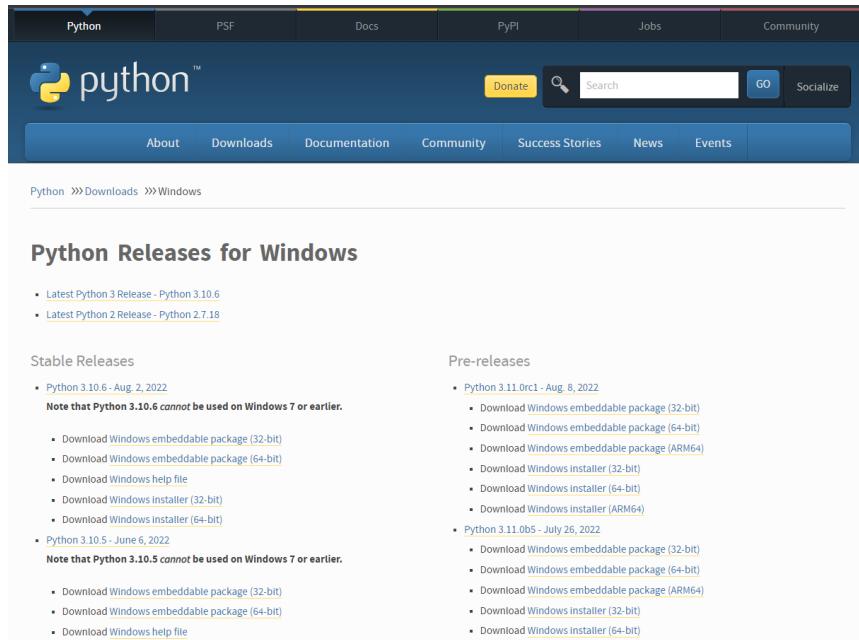


Figure 3.2: Pilihan Versi Python

Kemudian, buka file instalator python yang telah didownload, centang "Add Python 3.10 to PATH" dan klik *Customize Installation*

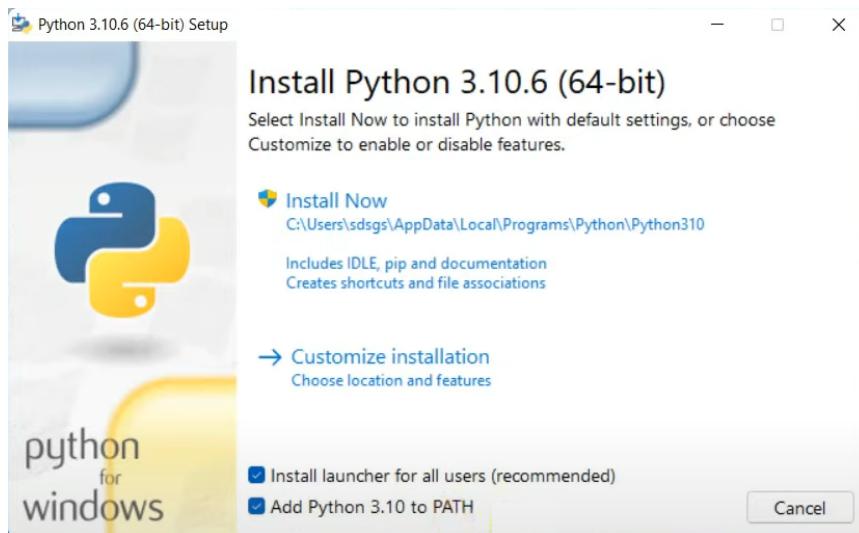


Figure 3.3: Instalator python

Pilih fitur yang akan digunakan, untuk saran pilih semua fitur agar instalasi python lengkap, lalu klik 'next'

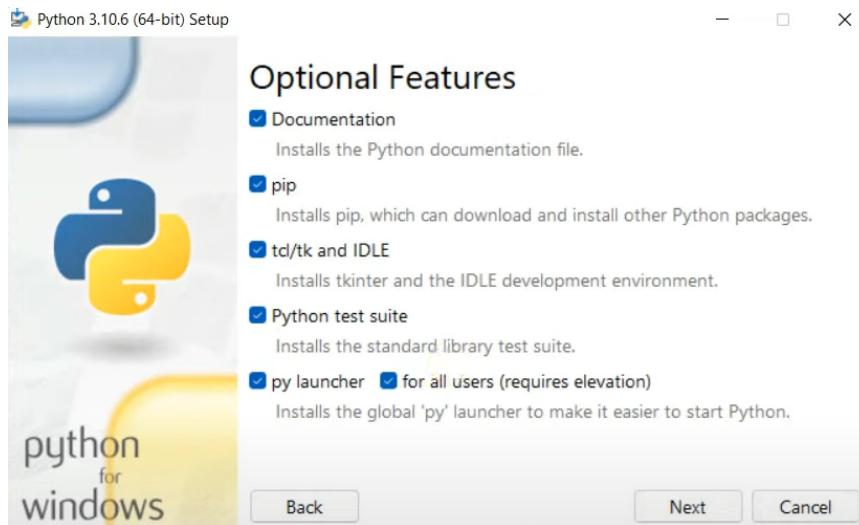


Figure 3.4: Pilihan fitur

Centang pilihan sesuai pada gambar, kemudian pilih direktori untuk lokasi penyimpanan instalasi python sesuai kebutuhan. Lalu klik "Install" dan tunggu hingga proses instalasi selesai.

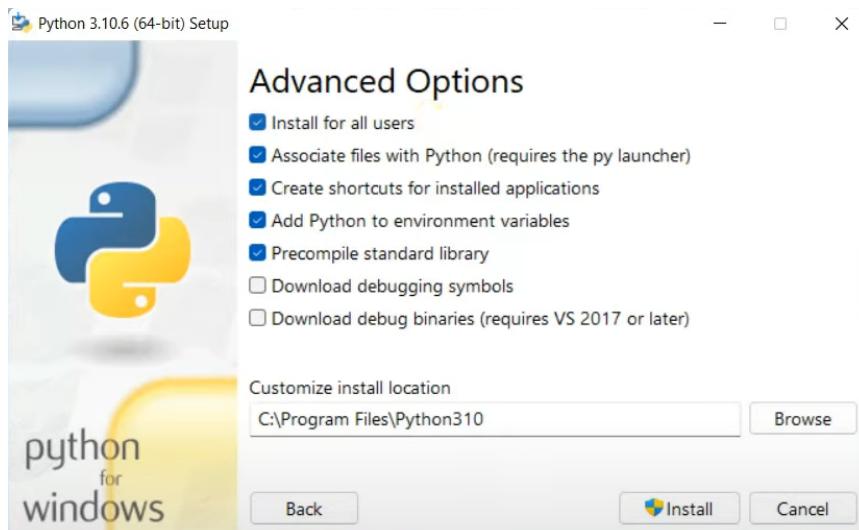


Figure 3.5: Pilihan lanjutan dan penyesuaian lokasi

3.1.2 Instalasi OpenCV

Untuk Instalasi openCV dapat dilakukan melalui CMD(*Command Prompt*). Buka direktori penyimpanan instalasi python, lalu menuju direktori 'Scripts' tempat pip.exe berada. Lalu tuliskan perintah *pip install opencv-contrib-python* untuk memulai instalasi openCV, tunggu instalasi hingga selesai.

```
Scripts> pip install opencv-contrib-python
```

Figure 3.6: Instalasi openCV

Setelah instalasi selesai, buka python IDLE atau pada CMD didalam direktori instalasi python, buka python.

Setelah python terbuka, ketik **import cv2** lalu enter, jika saat pengecekan openCV pada python tidak terjadi error, maka openCV berhasil diinstall.

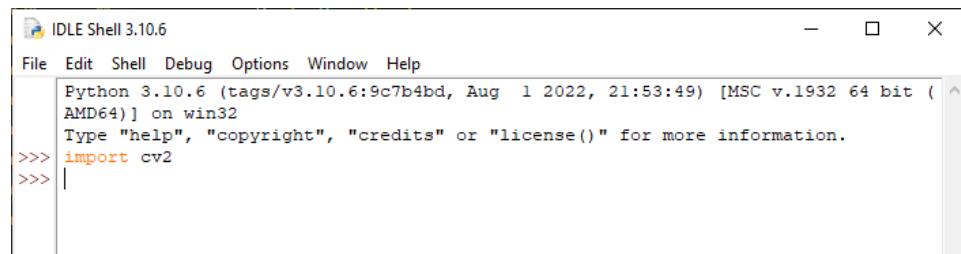


Figure 3.7: Cek openCV pada python IDLE

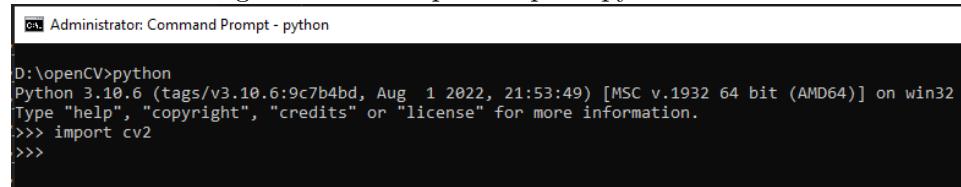


Figure 3.8: Cek openCV pada CMD

3.2 Pembuatan Sistem Face Detection

Pembuatan sistem Face Detection kali ini menggunakan algoritma Haar Cascade Classifier. Proses pertama yang dilakukan adalah dengan mengubah citra warna menjadi citra *grayscale*, selanjutkan melakukan pemindaihan pada citra *grayscale* untuk mendapatkan nilai fitur citra dengan *Haar-Like Feature* yang menyatakan objek wajah.

Berikut ini merupakan bagian-bagian untuk membuat sistem face detection atau pendekripsi wajah.

- Memasukan library, **cv2** = merupakan *library* openCV

```
import cv2
```

Figure 3.9: Memasukan library openCV

- Proses face detection, untuk melakukan proses deteksi wajah akan menggunakan algoritma *haarcascade*. Dengan fungsi **cv2.CascadeClassifier**

```
faceDetect = cv2.CascadeClassifier(
    'haarcascade_frontalface_default.xml')
```

Figure 3.10: Memasukan library openCV

- Memasukan video, **cv2.VideoCapture** merupakan fungsi untuk membaca video yang akan dijadikan *sample* dan bisa juga untuk menampilkan frame kamera yang terkoneksi dengan komputer.

```
cap = cv2.VideoCapture('data/video_test.mp4')
```

Figure 3.11: Memasukan video

- Membuka video/kamera dan merubah warna. Kode pada baris pertama digunakan untuk membuka video atau kamera yang terhubung, baris kedua merupakan kode untuk mengubah warna citra menjadi hitam putih menggunakan **cv2.cvtColor**

```
_, frame = cap.read()
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Figure 3.12: Membuka video dan ubah warna cira

- Fungsi untuk mendeteksi wajah, menggunakan fungsi **detectMultiScale**

```
face = faceDetect.detectMultiScale(gray, 1.3, 5);
```

Figure 3.13: Deteksi wajah

- Fungsi untuk membuat bingkai tanda jika wajah terdeteksi oleh sistem menggunakan **cv2.rectangle** jika berbentuk persegi. Dan fungsi **cv2.imshow** untuk menjalankan sistem yang telah dibuat.

```
for(x,y,w,h) in face:
    cv2.rectangle(frame, (x,y), (x+w, y+h), (255,0,0),2)

cv2.imshow('face', frame)
```

Figure 3.14: Kode bingkai wajah

Berikut adalah seluruh kode dan hasil untuk sistem deteksi wajah:

```
import cv2

faceDetect = cv2.CascadeClassifier(
    'haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture('data/video_test.mp4')
while cap.isOpened():
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    face = faceDetect.detectMultiScale(gray, 1.3, 5);
    for(x,y,w,h) in face:
        cv2.rectangle(frame, (x,y), (x+w, y+h), (255,0,0),2)

    cv2.imshow('face', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Figure 3.15: Kode deteksi wajah

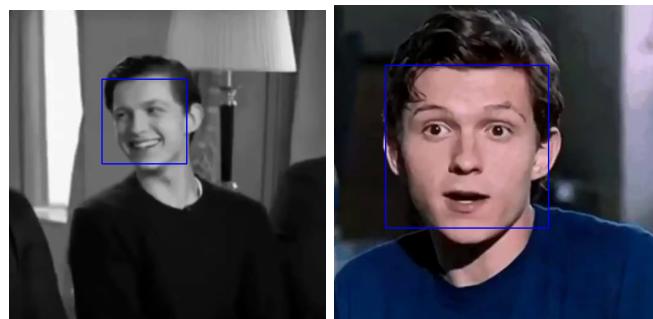


Figure 3.16: Hasil deteksi wajah

3.3 Pembuatan Sistem Face Recognition

Proses pertama untuk melakukan pengenalan wajah yaitu dengan mengumpulkan dataset yang akan di *training* dengan menangkap citra wajah pada saat awal deteksi wajah yang akan disimpan dan dikumpulkan berdasarkan id yang telah dimasukkan user. Setelah dataset terkumpul, selanjutnya sistem akan melakukan *training data* untuk mengenali wajah berdasarkan id. Kemudian proses penenalan wajah pun dilakukan dengan mendeteksi wajah menggunakan algoritma Haar-cascade classifier, lalu sistem akan melakukan pencocokan dengan menggunakan fitur LBPH untuk mencocokan wajah yang terdeteksi dengan dataset yang sudah ditraining sebelumnya.

3.3.1 Face Recognition dengan video

1. Proses mengumpulkan dataset dengan
 - Memasukan library openCV, yaitu **cv2**

```
import cv2
```

Figure 3.17: Memasukan library openCV

- Proses face detection, untuk melakukan proses deteksi wajah akan menggunakan algoritma *haarcascade*. Dengan fungsi **cv2.CascadeClassifier** pada baris pertama, pada baris kedua merupakan fungsi openCV untuk memasukan video atau kamera yang terhubung dengan **cv2.VideoCapture()**

```
face = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cam = cv2.VideoCapture('data/emma.gif')
```

Figure 3.18: Proses face detection dengan masukan video

- Menambahkan variabel 'jumlah' yang dimulai dari 0 untuk menyimpan data perulangan pengambilan gambar dataset dan juga pada baris selanjutnya ada variabel 'id' wadah masukan user untuk id dataset

```
jumlah = 0
id = input("Masukkan ID: ")
```

Figure 3.19: Tambah variabel

- Membaca video atau kamera yang sudah dimasukkan sebelumnya dengan fungsi **read()**, dan untuk mengubah warna citra menjadi hitam-putih/grayscale dengan fungsi **cv2.cvtColor**

```
while (True):
    ret, frame = cam.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Figure 3.20: Membaca video dan merubah warna citra

- Pada baris pertama fungsi untuk deteksi wajah dengan fungsi **detectMultiScale()**. Pada baris selanjutnya ada fungsi membuat bingkai **cv2.rectangle** lalu ada pengulangan untuk jumlah gambar yang ditangkap , lalu ada **cv2.imwrite** untuk menyimpan gambar data direktori dataset yang telah ditentukan, da **cv2.imshow** untuk menampilkan video atau kamera yang aka dideteksi dan diambil gambar untuk dataset.

```
wajah = face.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in wajah:
    cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
    jumlah += 1
    cv2.imwrite('dataset/user.'+str(id)+'. '+ str(jumlah) + ".jpg",
                gray[y:y + h, x:x + w])
    cv2.imshow('Train Face', frame)
```

Figure 3.21: Penyimpanan dataset

- Menentukan jumlah gambar yang akan tersimpan pada dataset

```
k = cv2.waitKey(100) & 0xff
if k == 27:
    break
elif jumlah>=40:
    break
```

Figure 3.22: Mengatur jumlah gambar yang diambil

3. IMPLEMENTASI SISTEM

15

- Keseluruhan kode program untuk pengambilan dataset

```

import cv2

face = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cam = cv2.VideoCapture('data/emma.gif')

jumlah = 0
id = input("Masukkan ID: ")

while (True):
    ret, frame = cam.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    wajah = face.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in wajah:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
        jumlah += 1
        cv2.imwrite('dataset/User.'+str(id)+'.\'' + str(jumlah) + ".jpg",
                    gray[y:y + h, x:x + w])
        cv2.imshow('Train Face', frame)

    k = cv2.waitKey(100) & 0xff
    if k == 27:
        break
    elif jumlah>=40:
        break

cam.release()
cv2.destroyAllWindows()

```

Figure 3.23: Kode pengambilan dataset

- Percobaan pengambilan dataset untuk tiga ID menggunakan manusukan video



Figure 3.24: Pengambilan dataset

2. Proses training dataset

- Import library cv2, numpy, Image from PIL (Pillow) library, dan os

```
import cv2
import numpy as np
from PIL import Image
import os
```

Figure 3.25: Pengambilan dataset

- pada baris ke-1, memasukan nama direktori dataset pada variabel path

Pada baris ke-2, untuk pengenalan wajah, menggunakan model algoritma LBPH yang sudah tersedia pada openCV

Pada bari ke-3, Proses face detection, untuk melakukan proses deteksi wajah akan menggunakan algoritma haarcascade

```
path = 'dataset'
recognizer = cv2.face.LBPHFaceRecognizer_create()
faceDetect = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

Figure 3.26: Pengambilan dataset

- Membuat method atau fungsi untuk mendapatkan image dataset sebagai data training agar dikenali oleh bahasa komputer

```
def getImagesAndLabels(path):
    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    faceSamples=[]
    ids = []
    for imagePath in imagePaths:
        PIL_img = Image.open(imagePath).convert('L')
        img_numpy = np.array(PIL_img, 'uint8')
        id = int(os.path.split(imagePath)[-1].split('.')[1])
        faces = faceDetect.detectMultiScale(img_numpy)
        for(x,y,w,h) in faces:
            faceSamples.append(img_numpy[y:y+h, x:x+w])
            ids.append(id)

    return faceSamples,ids
```

Figure 3.27: Pengambilan dataset

Memasukan direktori dataset pada variabel path menggunakan `os.path.join()` kedalam variabel imagePath. Lalu kumpulan image training akan diubah menggunakan library PIL menjadi sebuah array. Setelah itu, array data training akan dibuat oleh numpy. Dimana array yang diambil merupakan id dan juga faceSample.

- Training data

```
print("[INFO] Training faces. It will take a few second. Please wait . . . ")

face, ids = getImagesAndLabels(path)
recognizer.train(face, np.array(ids))

recognizer.write('recognizers/face-trainer.yml')
print("\n [INFO] {} faces trained. Exiting Program".format(len(np.unique(ids))))
```

Figure 3.28: Pengambilan dataset

Fungsi "getImagesAndLabels (path)", akan mengambil semua foto di direktori: "dataset/", mengembalikan 2 array: "Ids" dan "faces". Dengan array tersebut sebagai input, data akan di training menggunakan fungsi **recognizer.train()**. Untuk hasil training data akan disimpan dalam bentuk .yml pada direktori recognizers dengan fungsi **recognizer.write()**

- Proses training data

```
D:\openCV>python face_trainingData.py
[INFO] Training faces. It will take a few second. Please wait . . .
[INFO] 3 faces trained. Exiting Program
```

Figure 3.29: Pengambilan dataset

- Hasil training data berbentuk file .yml

Name	Date modified	Type	Size
face-trainer.yml	8/24/2022 10:33 AM	Yaml Source File	19,349 KB

Figure 3.30: Pengambilan dataset

3. Proses pengenalan wajah

- Import library, dengan menggunakan library numpy, cv2, dan juga os

```
import cv2
import numpy as np
import os
```

Figure 3.31: Import Library

- Melakukan pengenalan wajah menggunakan algoritman LBPH pada variabel **recognizer**, selanjutnya variabel recognizer membaca hasil dari training dataset dalam bentuk file **.yml** untuk mengekstrak data yang ada di dalam file

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("recognizers/face-trainer.yml")
```

Figure 3.32: Pengenalan wajah

- Proses deteksi wajah untuk dilakukan pengenalan/pencocokan wajah menggunakan algoritma *Haar-Cascade Classifier*

```
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);
```

Figure 3.33: Deteksi wajah

- Baris untuk tipe font yang akan diunculkan, serta variabel id untuk id dataset

```
font = cv2.FONT_HERSHEY_SIMPLEX
id = 0
```

Figure 3.34: Font dan ID

- Pembuatan array baru untuk menampilkan nama sesuai id

```
names = ['None', 'emma', 'zendaya', 'tom', 'Z', 'W']
```

Figure 3.35: Array nama

- Proses memasukan video atau device kamera, dan untuk mengatur ukuran frame yang akan ditampilkan

```
cam = cv2.VideoCapture('data/emma.gif')

minW = 0.1*cam.get(3)
minH = 0.1*cam.get(4)
```

Figure 3.36: Input video/kamera serta mengatur ukuran frame

- Untuk membaca video atau kamera yang dimasukkan sebelumnya, lalu mengubah warna menjadi grayscale. Kemudian deteksi gambar menggunakan **detectMultiScale()**

```

while True:
    ret, img = cam.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor = 1.2,
        minNeighbors = 5,
        minSize = (int(minW), int(minH)),
    )

```

Figure 3.37: Membaca dan perubahan citra

- **recognizer.predict()**, akan mengambil sebagai parameter bagian wajah yang ditangkap untuk dianalisis kecocokannya dengan data training dan akan mengembalikan kemungkinan kecocokan data

```

for(x,y,w,h) in faces:
    cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
    id, confidence = recognizer.predict(gray[y:y+h,x:x+w])

    if (confidence < 100):
        id = names[id]
        confidence = " {0}%".format(round(100 - confidence))
    else:
        id = "unknown"
        confidence = " {0}%".format(round(100 - confidence))

    cv2.putText(img,str(id),(x+5,y-5),font,1,(255,255,255),2)
    cv2.putText(img,str(confidence),(x+5,y+h-5),font,1,(255,255,0),1)

```

Figure 3.38: Mengenali kecocokan data

untuk kondisi **if else** untuk mengatur tampilnya data jika ada kecocokan dalam bentuk persentase. Jika ada kecocokan akan ditampilkan id yang sudah diganti dengan nama pada array dan juga besaran persentase yang dikenali oleh recognizer.

- Proses menampilkan video/kamera dan juga fungsi **waitKey()** untuk menutup program jika mekan tombol 'ESC'

```

cv2.imshow('camera',img)
k = cv2.waitKey(10) & 0xff
if k == 27:
    break

```

Figure 3.39: Menampilkan kamera atau video serta akhir program

- Hasil recognition dengan masukan video dan pembuatan label nama

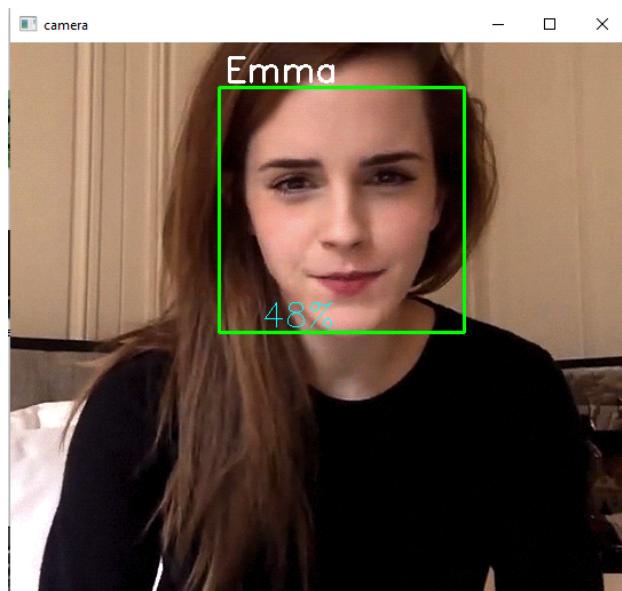


Figure 3.40: Hasil recognition video

3.3.2 Face Recognition dengan kamera

1. Proses pengumpulan dataset

- Memasukan library openCV **cv2** serta **os**. Librari **os** digunakan untuk mengakses fungsi pada sistem operasi

```
import cv2, os
```

Figure 3.41: Memasukan library

- Proses face detection dengan memasukan algoritma *haar-cascade classifier* serta memasukan video dengan **cv2.VideoCapture()**

```
face = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cam = cv2.VideoCapture(0)
```

Figure 3.42: Proses face detection dengan masukan kamera

- Membuat inputan untuk id serta label folder pada kumpulan dataset, pembuatan folder label dataset menggunakan library **os** yang menggunakan fungsi sitem operasi **mkdir**

```

jumlah = 0
id = input("Masukkan ID: ")
folder_name = input("Masukan nama anda: ")
os.mkdir('images/'+folder_name)

```

Figure 3.43: Masukan id serta nama untuk label dataset

- Proses deteksi wajah dan pengambilan dataset sebanyak 40 sample yang dimasukan pada direktori label

```

while (True):
    ret, frame = cam.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    wajah = face.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in wajah:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
        jumlah += 1
        cv2.imwrite('images/'+folder_name+'/'+folder_name+'.'+str(id)+'.'
                   + str(jumlah) + ".jpg", gray[y:y + h, x:x + w])
        cv2.imshow('Train Face', frame)

    k = cv2.waitKey(100) & 0xff
    if k == 27:
        break
    elif jumlah>=40:
        break

```

Figure 3.44: Proses face detection dan pengambilan dataset

- Hasil pengambilan dataset

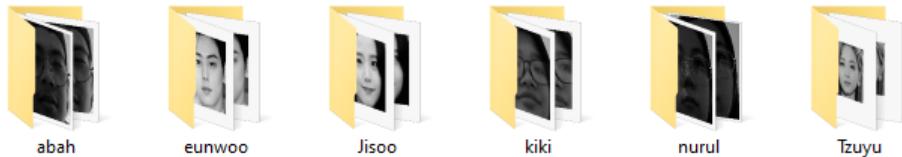


Figure 3.45: Kumpulan label direktori dataset

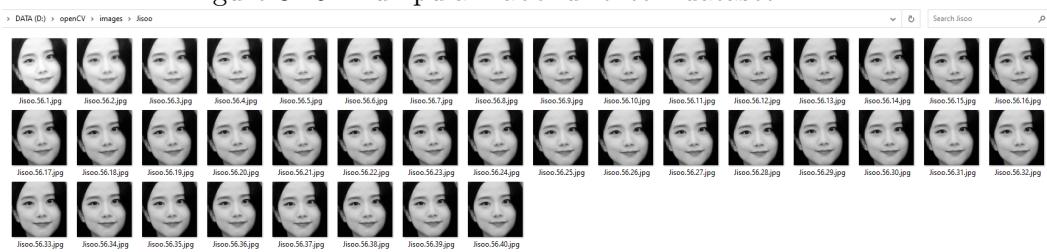


Figure 3.46: Isi direktori salah satu label dataset

2. Proses training dataset

- Import library, training dataset kali ini memerlukan library baru yaitu :

- * **os** untuk mengakses fungsi pada sistem operasi,
- * **pickle** untuk menyimpan dan membaca data ke dalam/dari sebuah file, dan
- * **Image dari PIL** untuk memuat gambar dari file, dan untuk membuat gambar baru.

```
import os, cv2, pickle
from PIL import Image
import numpy as np
```

Figure 3.47: Import library

- Baris ke-1 dan 2 untuk mengambil dataset dengan join direktori tempat kumpulan dataset

Baris ke -3 untuk memasukan algoritma face detection

Baris ke-4 untuk proses deteksi wajah dengan algoritma LBPH

```
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
image_dir = os.path.join(BASE_DIR, "images")

faceDetect = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
recognizer = cv2.face.LBPHFaceRecognizer_create()
```

Figure 3.48: Pengambilan dataset

- Pembuatan variabel array untuk data id dan label gambar

```
current_id = 0
label_ids = {}
ids=[]
labels = []
```

Figure 3.49: Variabel

- Menelusuri seluruh folder dan mencari gambar, file yang diakhiri dengan jpg. Menggunakan root, untuk menemukan nama folder file dan menyimpannya dalam variabel label. Nama orang yang akan ditampilkan di layar akan menjadi labelnya.

```
for root, dirs, files in os.walk(image_dir):
    for file in files:
        if file.endswith("jpg"):
            path = os.path.join(root, file)
            label = os.path.basename(root)
```

Figure 3.50: Menelusuri folder dataset

- Proses mengubah gambar menjadi array numpy dan melakukan training data. Untuk mengambil gambar menggunakan library **Image** Pillow pada python kemudian gambar yang diambilakan dikonversi kedalan grayscale, lalu diubah ke dalam bentur array numpy. Hal ini dilakukan untuk memberikan label terhadap id tertentu. selanjutnya akan dilakukan deteksi wajah menggunakan algoritma **cascade**

```
if not label in label_ids:
    label_ids[label] = current_id
    current_id += 1
id_ = label_ids[label]

pil_image = Image.open(path).convert("L")
image_array = np.array(pil_image, "uint8")

face = faceDetect.detectMultiScale(image_array, 1.3, 4)
```

Figure 3.51: Menelusuri folder dataset dan konversi gambar

- Mengisi bingkai deteksi dengan nama label sesuai kecocokannya

```
for(x,y,w,h) in face:
    roi = image_array[y:y+h, x:x+w]
    ids.append(roi)
    labels.append(id_)
```

Figure 3.52: Melampirkan nama

- Menyimpan hasil training data pada file face-trainer.yml

```

with open("pickles/face-labels.pickle", 'wb') as f:
    pickle.dump(label_ids, f)

recognizer.train(ids, np.array(labels))
recognizer.save("recognizers/face-trainer.yml")
print("\n [INFO] faces trained. Exiting Program")

```

Figure 3.53: Simpan hasil training

- Keseluruhan program training data

```

import os, cv2, pickle
from PIL import Image
import numpy as np

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
image_dir = os.path.join(BASE_DIR, "images")

faceDetect = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
recognizer = cv2.face.LBPHFaceRecognizer_create()

current_id = 0
label_ids = {}
ids = []
labels = []

for root, dirs, files in os.walk(image_dir):
    for file in files:
        if file.endswith("jpg"):
            path = os.path.join(root, file)
            label = os.path.basename(root)

            if not label in label_ids:
                label_ids[label] = current_id
                current_id += 1
            id_ = label_ids[label]

            pil_image = Image.open(path).convert("L")
            image_array = np.array(pil_image, "uint8")

            face = faceDetect.detectMultiScale(image_array, 1.3, 4)

            for(x,y,w,h) in face:
                roi = image_array[y:y+h, x:x+w]
                ids.append(roi)
                labels.append(id_)

print("[INFO] Training faces. It will take a few second. Please wait . . . ")

with open("pickles/face-labels.pickle", 'wb') as f:
    pickle.dump(label_ids, f)

recognizer.train(ids, np.array(labels))
recognizer.save("recognizers/face-trainer.yml")
print("\n [INFO] faces trained. Exiting Program")

```

Figure 3.54: Program training

3. Proses pengenalan wajah

- Import Library

```
import cv2
import numpy as np
import pickle
```

Figure 3.55: Import library

- Menggunakan algoritma LBPH untuk pengenalan wajah, dengan membaca model file hasil data training sebelumnya. Dan penggunaan library **pickle** untuk mengambil label nama sesuai id

```
faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("recognizers/face-trainer.yml")

labels = {"person_name": 1}
with open("pickles/face-labels.pickle", 'rb') as f:
    og_labels = pickle.load(f)
    labels = {v:k for k,v in og_labels.items() }
```

Figure 3.56: Model trained

- Pengaturan kamera, pengubahan warna citra menjadi grayscale untuk pencocokan dengan train dataset

```
cam = cv2.VideoCapture(0)
minW = 0.1*cam.get(3)
minH = 0.1*cam.get(4)
names = "none"

font = cv2.FONT_HERSHEY_SIMPLEX
while True:
    ret, img = cam.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor = 1.2,
        minNeighbors = 5,
        minSize = (int(minW), int(minH)),
    )
```

Figure 3.57: Pengaturan kamera

- Proses prediksi untuk pencocokan inputan wajah pada kamera dengan train dataset. Dengan menampilkan label nama sesuai id dan persentase kecocokan.

```
for(x, y, w, h) in faces:  
    cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)  
    id_, conf = recognizer.predict(gray[y:y+h, x:x+w])  
  
    if(conf < 100):  
        names = labels[id_]  
        conf = " {0}%".format(round(100 - conf))  
    else:  
        names = "unknown"  
        conf = " {0}%".format(round(100 - conf))  
  
    cv2.putText(img,names,(x+5,y-5),font,1,(255,255,255),2)  
    cv2.putText(img,str(conf),(x+5,y+h-5),font,1,(255,255,0),1)  
  
cv2.imshow('camera',img)  
k = cv2.waitKey(10) & 0xff  
if k == 27:  
    break
```

Figure 3.58: Import library

- Hasil

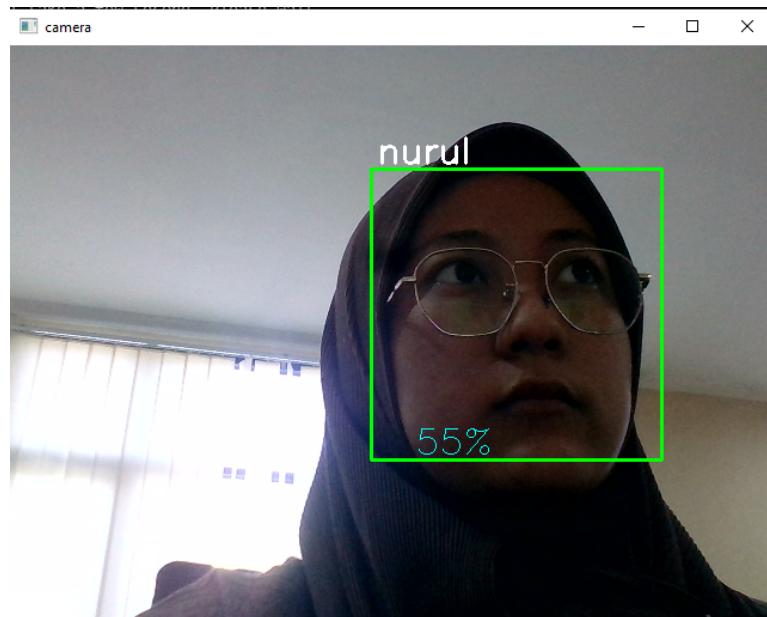


Figure 3.59: Hasil face recognition jarak sekitar 30 cm dengan kamera

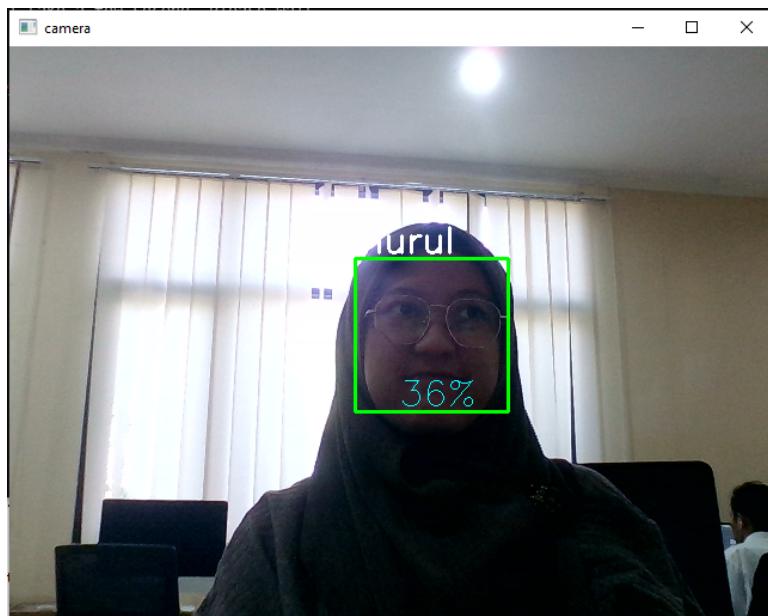


Figure 3.60: Hasil face recognition jarak sekitar 1 meter dengan kamera