Nama : Nova Andre Saputra

NRP : 2103181060

Kelas : 2 D3 IT B

1. Deskripsi Proyek Akhir

Proyek pengenalan wajah ini menggunakan implementasi algoritma *EigenFaces*. Konsepnya menggunakan Principal Component Analysis (PCA), PCA ini dapat digunakan dalam OpenCV. Eigenface merupakan salah satu contoh dari metode PCA.

Proyek akhir ini akan ada tiga sub-tugas yakni : penambahan dataset melalui webcam selama 15 detik, pengenalan wajah melalui video streaming (webcam) dan pengenalan wajah melalui video file. Untuk saat ini saya sudah menyelesaikan pengerjaan bagian pengenalan wajah melalui video streaming (webcam) dan video file. Namun untuk saat ini, saya masih terus mencari referensi mengenai perekaman wajah dan menyimpannya ke dalam dataset dengan baik, sementara saya tambahkan dataset melalui perekaman selama 15 detik dengan pengambilan 5 foto wajah dataset, dan saya ingin menambahkan informasi tambahan bagi user agar lebih mudah mengerti terhadap alur aplikasi (GUI).

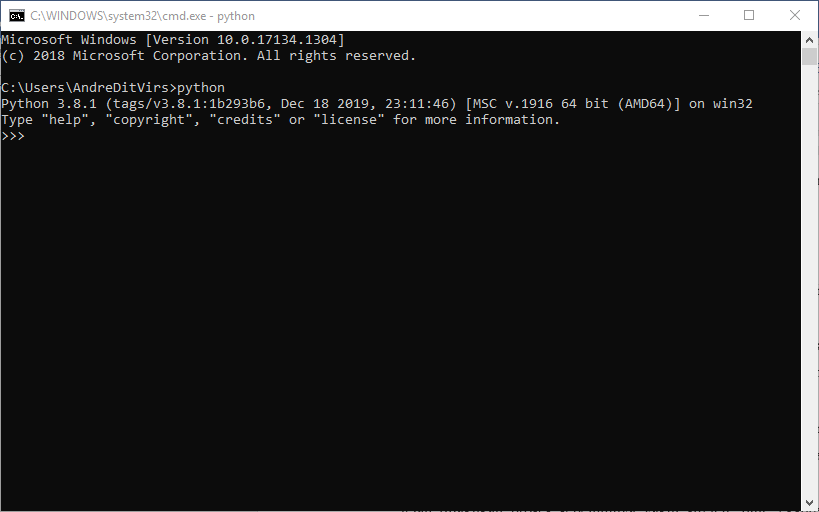
Berikut penjelasan langkah cara mengenali wajah dengan metode eigenface :

Di proyek ini saya masih menggunakan lima file yang memiliki fungsi terpisah, yakni file untuk training gambar dari dataset dan file untuk melakukan pengenalan dengan data yang didapatkan dari proses training gambar melalui file .json

* File add\_eigen.py : Melakukan perekaman dan penyimpanan 5 foto wajah dataset dari webcam dengan haarcascade.
* File t\_eigen.py : Melakukan training dengan menggunakan file *t\_eigen.py*, gambar akan diambil dari folder dataset yang telah dinamai dengan format nrp-nama dan melakukan pengambilan wajah (dengan file haarcascade) dan menyimpannya dalam warna *grayscale* di folder dataset\_haar. Gambar yang ada dalam folder dataset\_haar akan diproses kembali dan diubah dalam bentuk .json yang menyimpan semua ektraksi gambar berupa angka.
* File f\_eigen.py : Melakukan pengenalan wajah dengan metode eigenface yang didapat melalui opencv yang menghitung jarak dari gambar tersebut dibandingkan dengan gambar-gambar yang ada di dalam file ekstraksi .json. Jika jarak yang telah diukur tersebut hasilnya diatas nilai dari *threshold* maka gambar tersebut dikenali oleh sistem, tetapi bila nilai yang dihasilkan lebih kecil maka gambar tersebut termasuk dalam gambar yang tidak dikenali oleh sistem karena sistem hanya mengenali gambar yang ada di dalam dataset.
* File fv\_eigen.py : Melakukan pengenalan wajah terhadap file video dengan memasukkan nama file dan formatnya (contoh : .mp4, dan sebagainya). File video akan tampil dengan garis biru deteksi wajah dan jika selesai akan disimpan dengan nama output.avi.
* File gui.py : Menampilkan menu dengan keempat pilihan menu dengan chooice box, sehingga lebih mempermudah user dalam memilih dengan single chooice, urutan sudah diatur dari awal hingga akhir. Terdapat message box yang ditunjukkan kepada user apa yang sedang terjadi/ada tidaknya error.

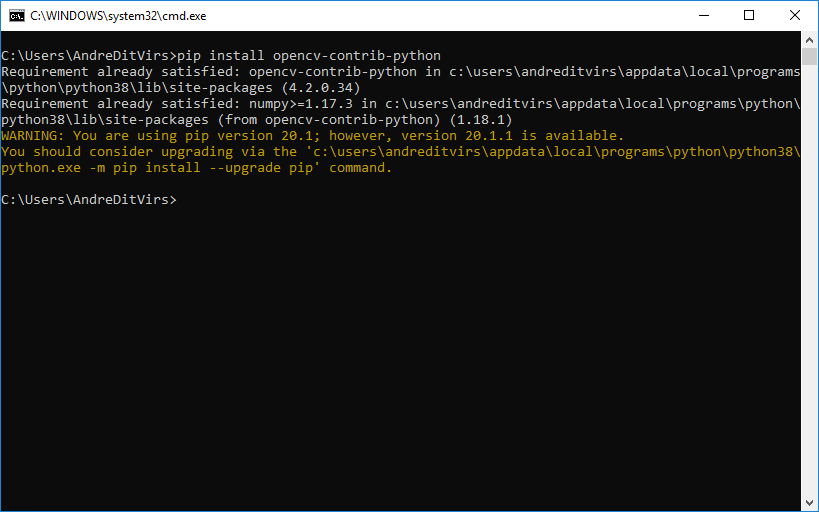
Namun ada beberapa keterbatasan dari algoritma eigenface yakni pencahayaan yang merupakan salah satu faktor pembeda sehingga jika ada perbedaan cahaya wajah akan sukar dikenali dan dianggap berbeda wajah. Adapun faktor lain seperti gambar yang telah dilakukan proses sebelumnya yakni stretch, blur, ekspresi wajah, dan sudut pengambilan gambar.

1. Urutan Proses (Sampai saat ini)
2. Install python terlebih dahulu (di proyek ini saya menggunakan python 3.8.1 dan sistem operasi windows).
3. Coba cek installasi dengan menggunakan perintah python pada cmd, jika berhasil membuka python intepreter maka installasi sudah berhasil.



Gambar 1. Cek instalasi python

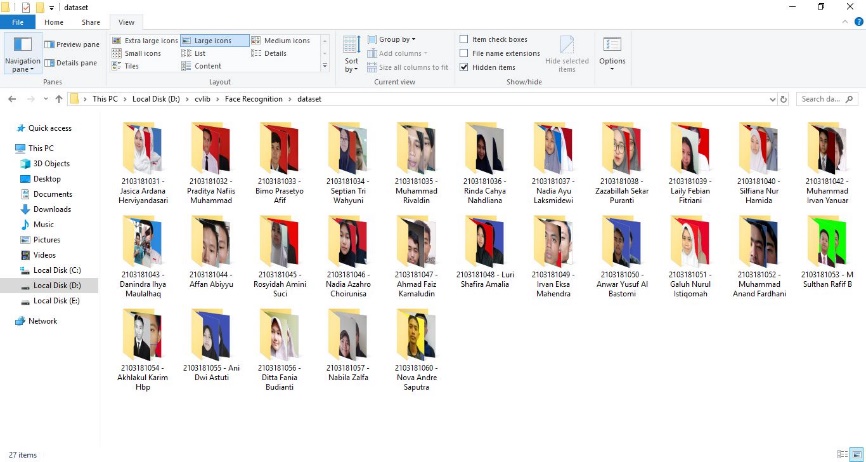
1. Seperti yang dijelaskan sebelumnya, Untuk melakukan pengenalan wajah dengan metode Eigenface, dibutuhkan library OpenCV. Install library OpenCV dengan perintah *pip install opencv-contrib-python* (perlu diperhatikan harus menggunakan opencv-**contrib**-python, jika tidak akan muncul error berupa library “cv2.cv2”).



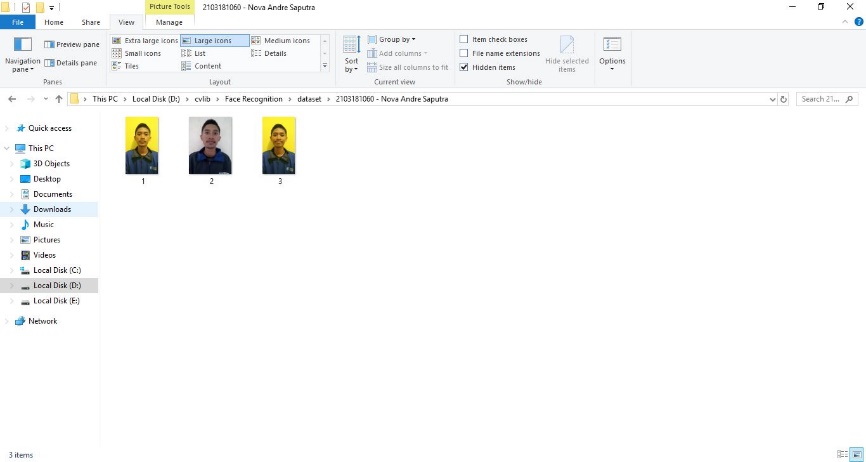
Gambar 2. Instalasi libraryOpenCV

Jika menginstall di dalam virtual environtment (bagi yang sudah menginstall virtualenv dengan *pip install virtualenv* namun upgrade dulu pip dengan *python -m pip install –upgrade pip* agar menjadi versi baru), maka buat dulu virtualenv, aktifkan virtual environtmentnya dan jalankan perintah *pip install opencv-contrib-python*.

1. Siapkan folder dataset yang berisi folder nrp-nama dan di di dalamnya terdapat gambar – gambar yang siap ditrainning (semakin banyak, semakin baik/mudah dibedakan).

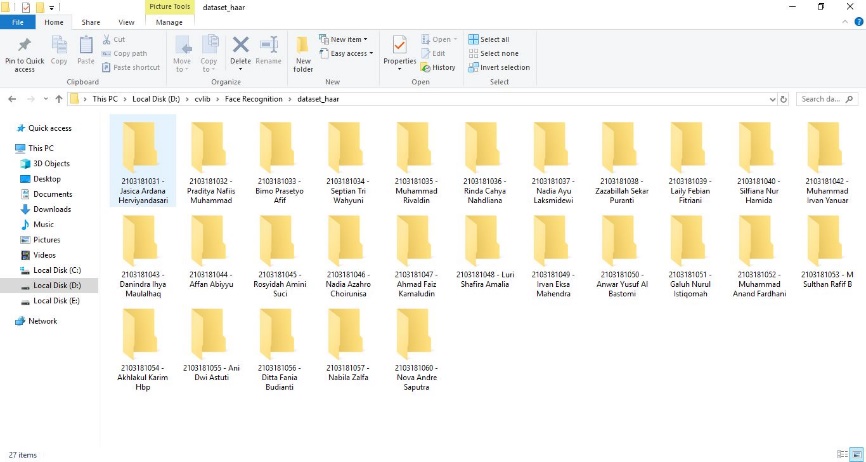


Gambar 3. Folder dataset



Gambar 4. Contoh foto pada salah satu folder nrp-nama

1. Siapkan juga folder dataset\_haar, folder ini dibiarkan kosong saja, tapi tetap buat folder nrp-nama

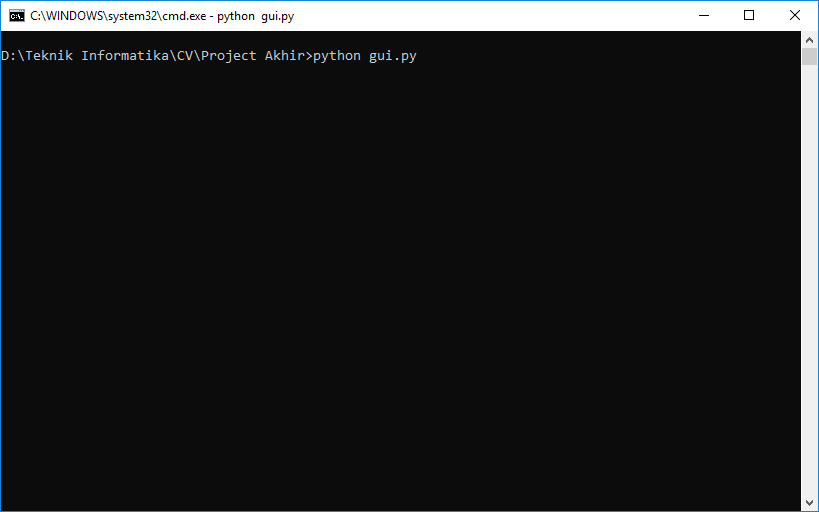


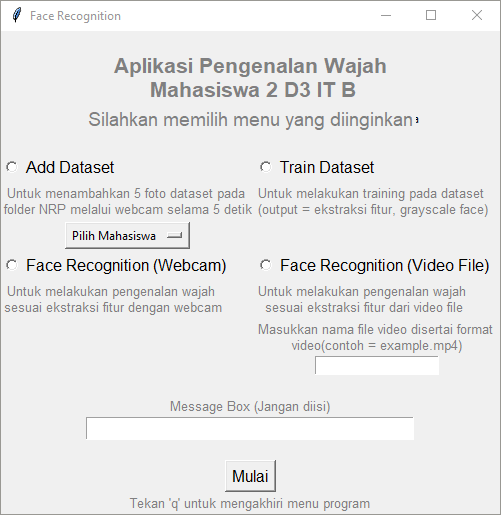
Gambar 5. Folder kosong dataset\_haar

1. Buat kode untuk face recognition (ada di lampiran)

Seperti yang dijelaskan pada deskripsi file terbagi menjadi lima file maka

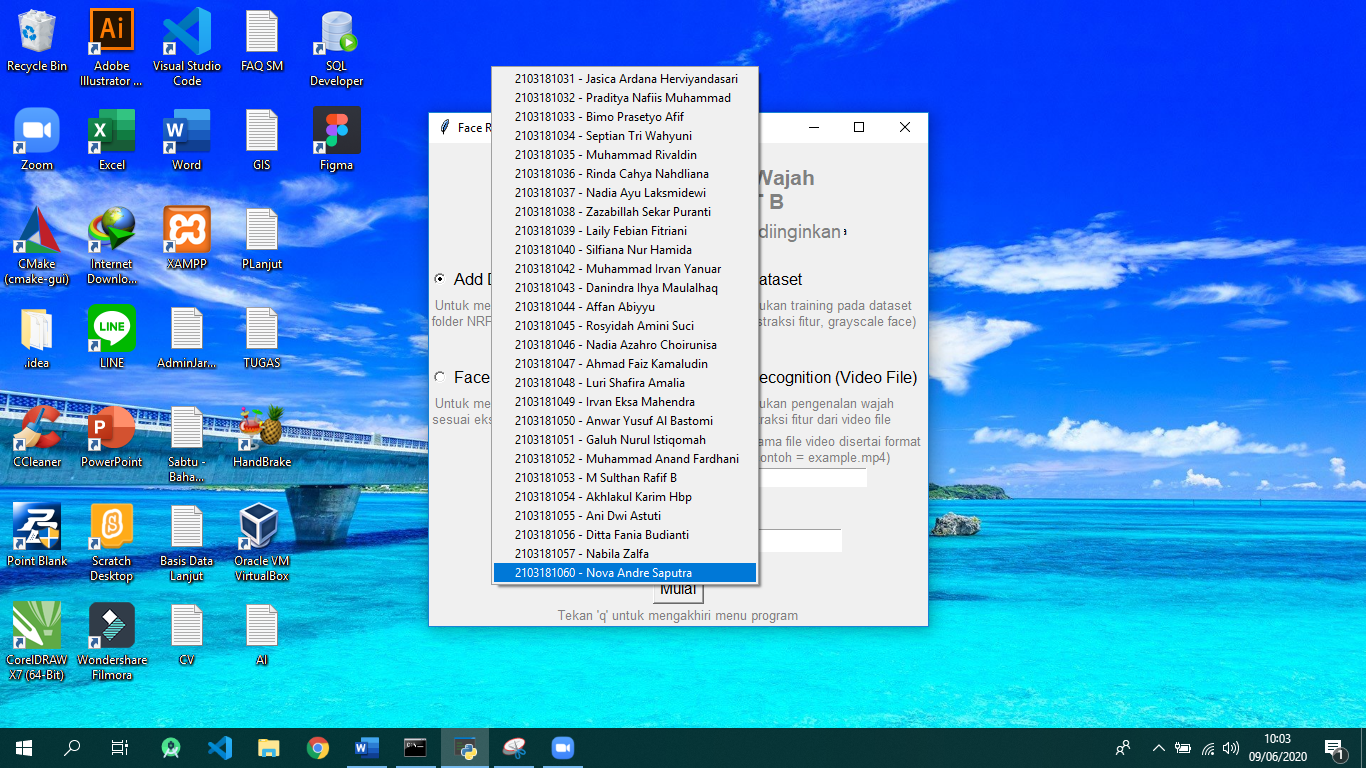
Jalankan file gui.py yang berisi shortcut menu dari keempat file python proses pengenalan wajah yakni add\_eigen.py, t\_eigen.py, f\_eigen.py, fv\_eigen.py



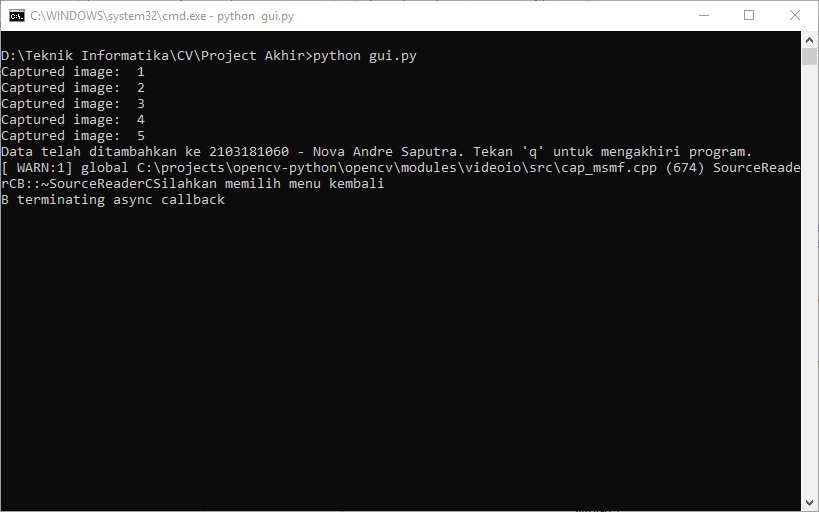
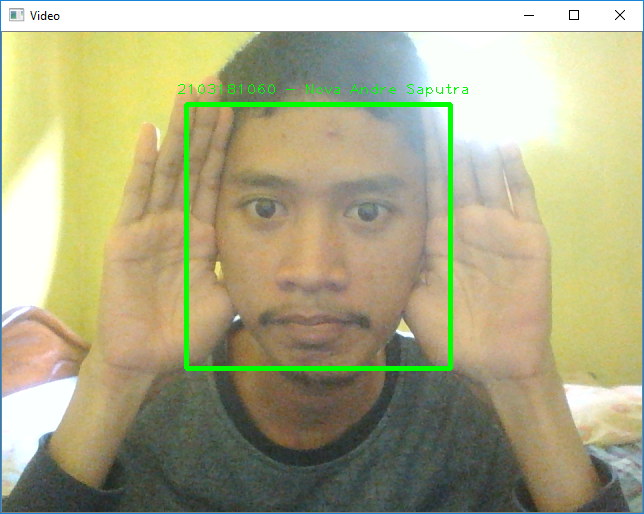


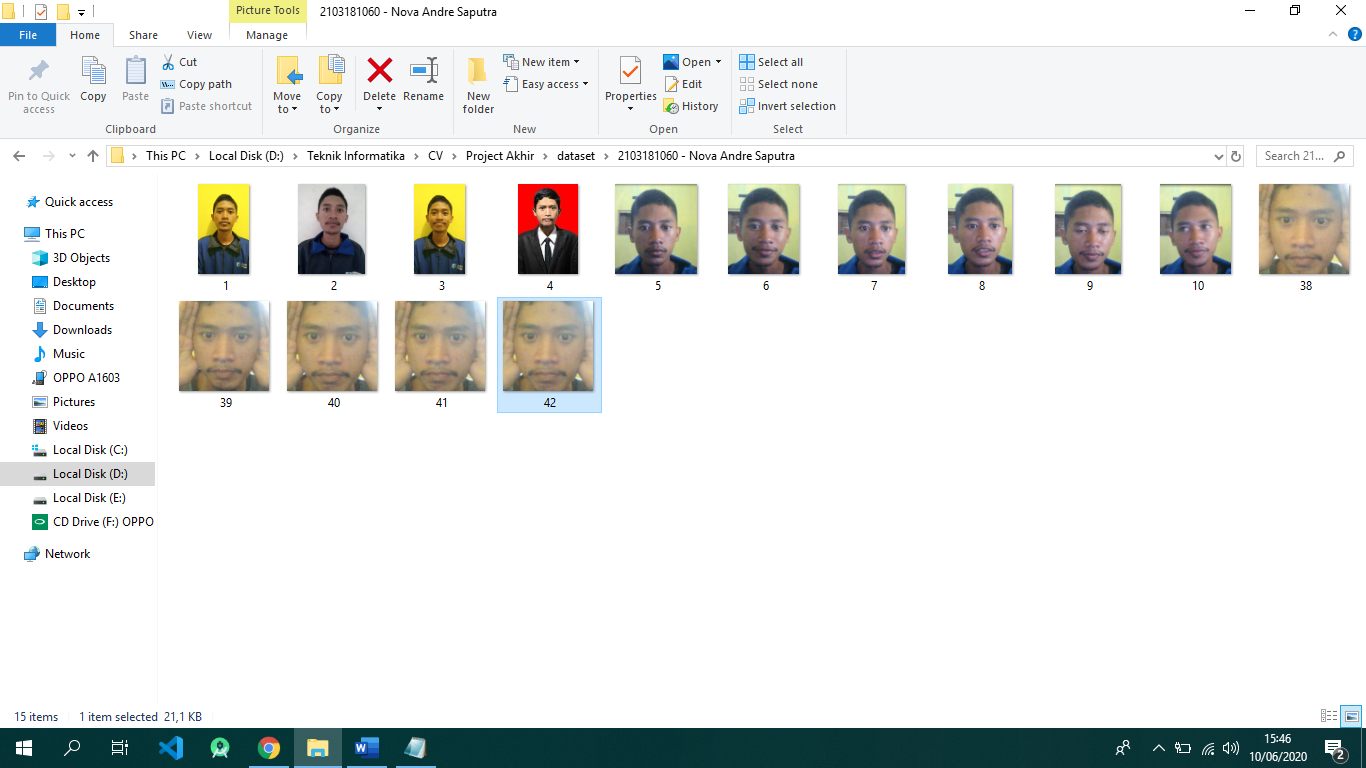
Gambar 6. Jalankan file gui.py

Pilih salah satu menu untuk awalannya Anda dapat memilih add dataset untuk menambahkan foto ke suatu folder pada NRP tertentu untuk memperkuat data pengenalan. Pilih Add Dataset -> pilih nrp-nama mahasiswa di dropdown dan klik tombol mulai.



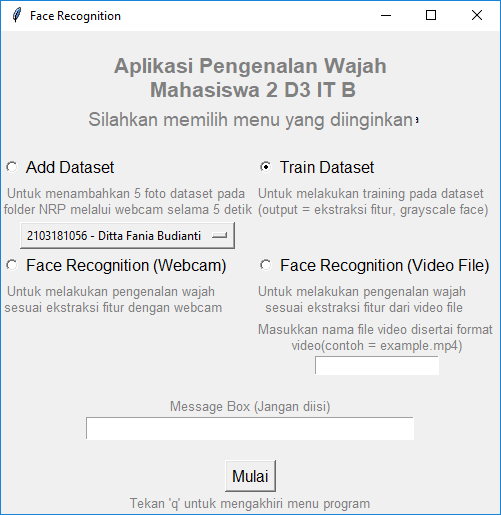
Gambar 7. Pilih nrp-nama di dropdown

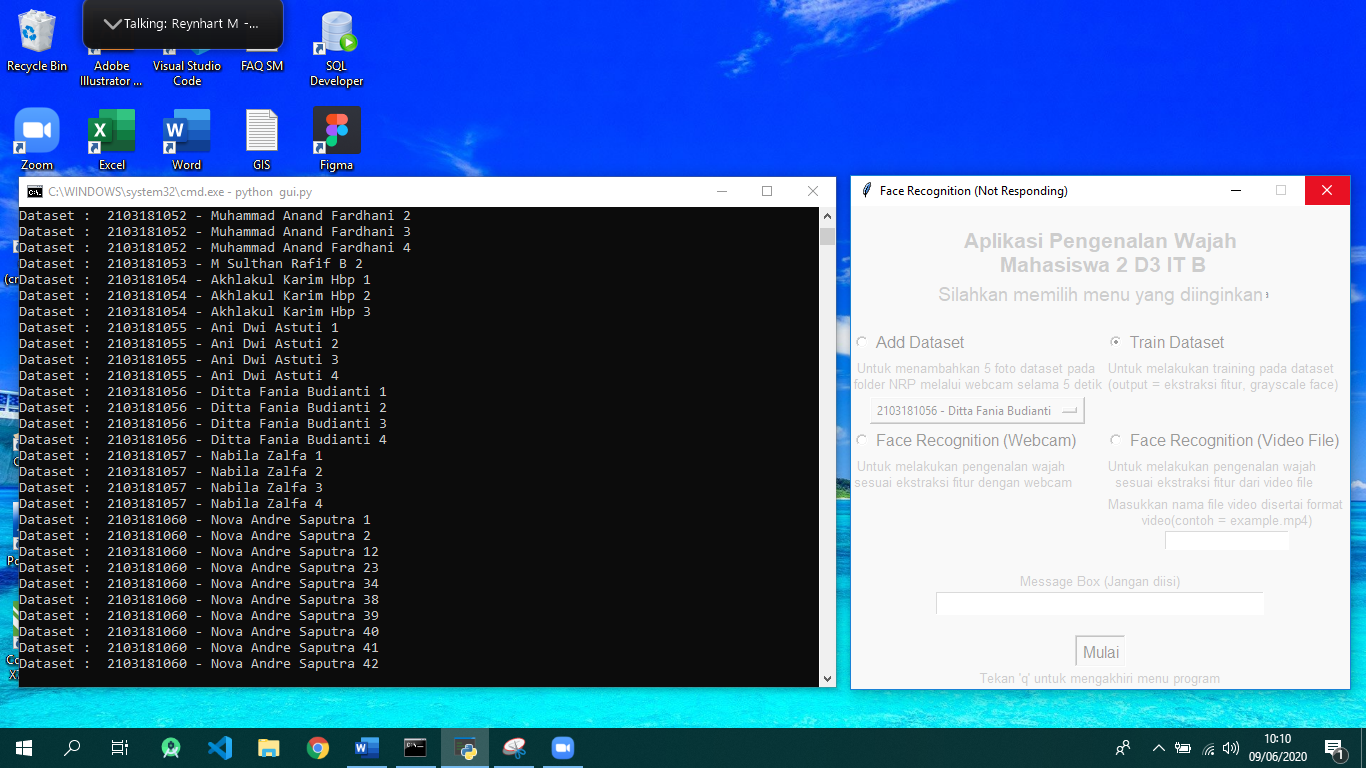




Gambar 7. Proses Add Dataset saat perekaman dengan webcam (garis hijau untuk perekaman dataset)

Pilih menu Train Dataset untuk menjalankan t\_eigen.py sehingga, akan dilakukan pembacaan dari folder dataset yang sebelumnya telah ditambahkan foto wajah dari webcam (dari add dataset sebelumnya) dan/atau dimasukkan manual.

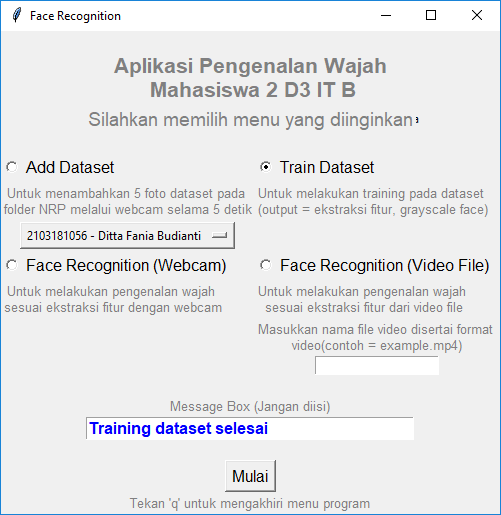
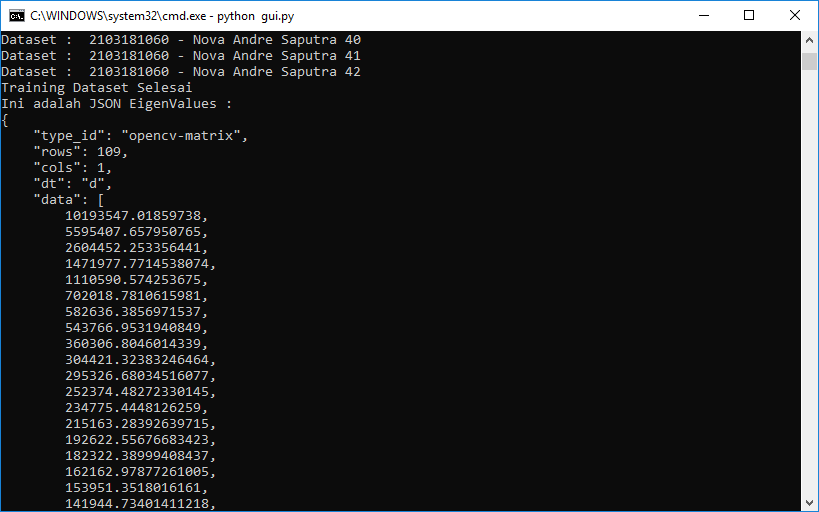




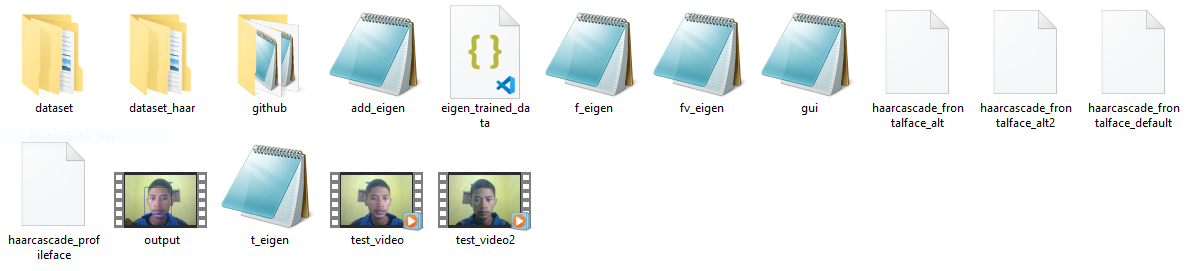
Gambar 8. Proses training dataset

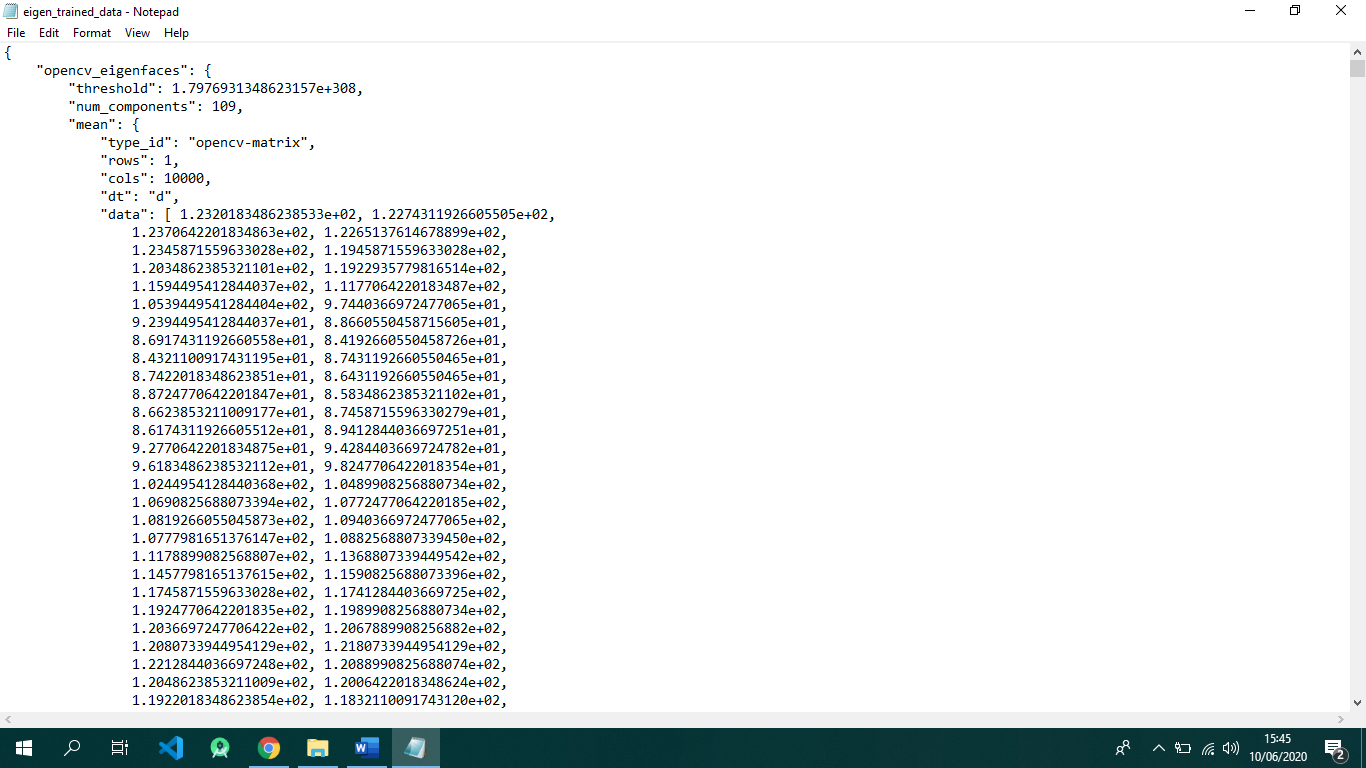
Program akan membaca setiap gambar dalam folder dataset (dengan haarcascade) sehingga gambar akan terproses menjadi terpotong, grayscale, sama ukurannya lalu disimpan dalam folder dataset\_haar. Program juga akan meng-generate file .xml sebagai penyimpanan hasil fitur untuk digunakan dalam file f\_eigen.py nanti.

Dan setelah training dataset selesai, maka akan tercetak JSON nilai eigen (*eigenvalues*) untuk validasi



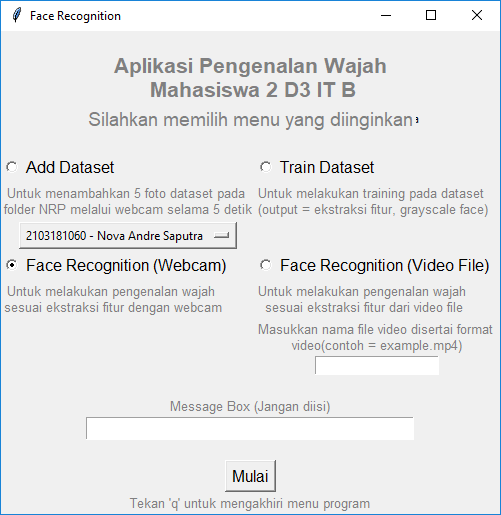
Gambar 9. Cetak JSON *eigenvalue object* saat training selesai



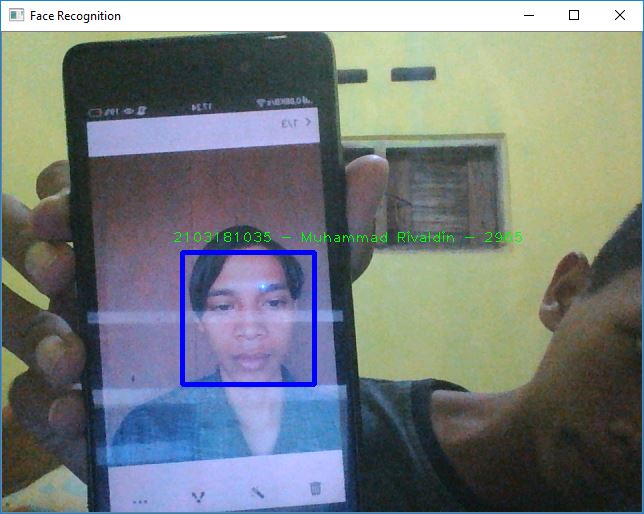
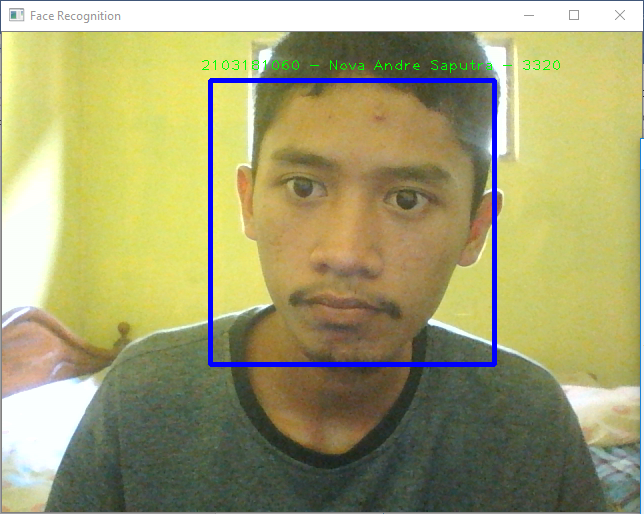


Gambar 10. File eigen\_trained\_data.json sebagai file untuk menyimpan hasil ekstraksi

Pilih menu Face Recognition (Webcam) untuk memulai pengenalan dengan menggunakan webcam.



Gambar 10. Pilih menu Face Recognition (Webcam)



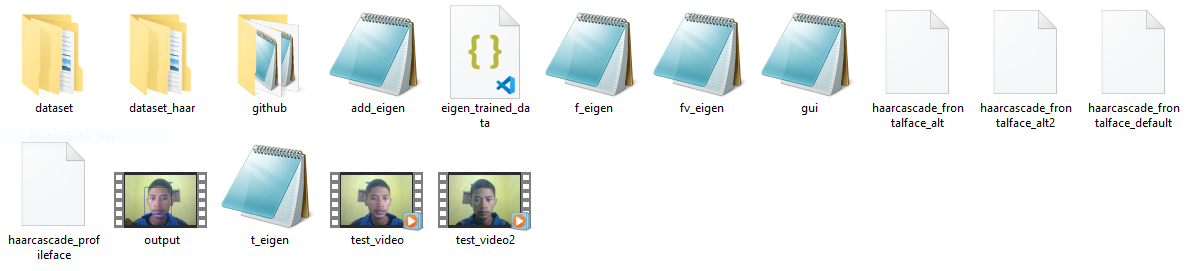
Gambar 11. Pengenalan wajah saat webcam (garis biru untuk pengenalan)

Pilih menu Face Recognition (Video File) untuk menjalankan file fv\_eigen dan juga masukkan nama dan format file video pada folder yang sama dengan dimana aplikasi dijalankan.



Gambar 12. Pengenalan wajah saat video file, program membuka video dan langsung mengenali wajah

Setelah melakukan pengenalan wajah terhadap video, hasil video yang disertai dengan kotak pengenalan akan disimpan dengan nama output.avi.



Progress setelah ini adalah menambahkan deskripsi project ke gui.py sehingga *user* dapat lebih mudah untuk menggunakan program.

1. Lampiran

Source code dapat dilihat di https://github.com/AndreDitVir01/eigenfaces\_face\_recognition-cv-desktop

* + File gui.py

from tkinter import \*

import os

folder\_path = 'D:\\Teknik Informatika\\CV\\Project Akhir\\'

in\_path = folder\_path+'dataset\\'

def list\_dir():

listds = os.listdir(in\_path)

number\_files = len(listds)

os.chdir(in\_path)

arr = os.listdir()

return arr

def show\_info():

os.chdir(folder\_path)

message\_txt.delete(0.0, 'end')

menu = menu\_value\_choice.get()

name = str('"'+tkvar.get()+'"')

if menu == 1:

os.system('python add\_eigen.py %s'%name)

message\_txt.insert(0.0, "Dataset berhasil ditambahkan")

elif menu == 2:

os.system('python t\_eigen.py')

message\_txt.insert(0.0, "Training dataset selesai")

elif menu == 3:

os.system('python f\_eigen.py')

message\_txt.insert(0.0, "Mengakhiri Webcam")

elif menu == 4 :

video\_name = video\_file.get()

if video\_name != '':

os.system('python fv\_eigen.py %s'%video\_name)

message\_txt.insert(0.0, "Video tersimpan")

else:

message\_txt.insert(0.0, "Anda belum memasukkan nama video")

else:

message\_txt.insert(0.0, "Anda belum memilih salah satu menu")

rootWindow = Tk()

video\_file = StringVar()

tkvar = StringVar(rootWindow)

mahasiswa = sorted(list\_dir())

tkvar.set('Pilih Mahasiswa')

popupMenu = OptionMenu(rootWindow, tkvar, \*mahasiswa)

Label(rootWindow, text="Pilih Mahasiswa").grid(row=2, column=2)

rootWindow.resizable(height = None, width = None)

rootWindow.title("Face Recognition")

welcome\_label = Label(rootWindow, text="Aplikasi Pengenalan Wajah\n Mahasiswa 2 D3 IT B", fg="gray",

font=("arial", 16,"bold"))

procedure\_label = Label(rootWindow, text="Silahkan memilih menu yang diinginkan", fg="gray",

font=("arial", 14))

add\_label = Label(rootWindow, text="Untuk menambahkan 5 foto dataset pada \nfolder NRP melalui webcam selama 5 detik", fg="gray",

font=("arial", 10))

train\_label = Label(rootWindow, text="Untuk melakukan training pada dataset \n(output = ekstraksi fitur, grayscale face)", fg="gray",

font=("arial", 10))

webcam\_label = Label(rootWindow, text="Untuk melakukan pengenalan wajah \nsesuai ekstraksi fitur dengan webcam", fg="gray",

font=("arial", 10))

video\_label = Label(rootWindow, text="Untuk melakukan pengenalan wajah \nsesuai ekstraksi fitur dari video file", fg="gray",

font=("arial", 10))

message\_label = Label(rootWindow, text="Message Box (Jangan diisi)", fg="gray",

font=("arial", 10))

att\_label = Label(rootWindow, text="Tekan 'q' untuk mengakhiri menu program", fg="gray",

font=("arial", 10))

video\_entry\_label = Label(rootWindow, text="Masukkan nama file video disertai format \nvideo(contoh = example.mp4)", fg="gray",

font=("arial", 10))

video\_entry = Entry(rootWindow, textvariable=video\_file) # Untuk masukkan nama video

menu\_value\_choice = IntVar()

radio\_button\_a = Radiobutton(rootWindow, text="Add Dataset", variable=menu\_value\_choice,

font=("arial", 12), value=1)

radio\_button\_t = Radiobutton(rootWindow, text="Train Dataset", variable=menu\_value\_choice,

font=("arial", 12), value=2)

radio\_button\_f = Radiobutton(rootWindow, text="Face Recognition (Webcam)", variable=menu\_value\_choice,

font=("arial", 12), value=3)

radio\_button\_fv = Radiobutton(rootWindow, text="Face Recognition (Video File)", variable=menu\_value\_choice,

font=("arial", 12), value=4)

message\_txt = Text(rootWindow, width=36, height=1, wrap=WORD, fg="blue",

font=("arial", 12, "bold"))

lets\_see\_button = Button(rootWindow, text="Mulai", font=("arial", 12),

command=lambda: show\_info())

# Setting the layout

rootWindow.grid\_rowconfigure(0, minsize=20)

welcome\_label.grid(row=1, columnspan=4)

procedure\_label.grid(row=2, columnspan=4)

rootWindow.grid\_rowconfigure(3, minsize=20)

radio\_button\_a.grid(row=4, column=1, sticky=W)

add\_label.grid(row=5, column=1, sticky=W)

radio\_button\_t.grid(row=4, column=2, sticky=W)

train\_label.grid(row=5, column=2, sticky=W)

popupMenu.grid(row=6, column=1)

radio\_button\_f.grid(row=7, column=1, sticky=W)

webcam\_label.grid(row=8, column=1, sticky=W)

radio\_button\_fv.grid(row=7, column=2, sticky=W)

video\_label.grid(row=8, column=2, sticky=W)

video\_entry\_label.grid(row=9, column=2, sticky=W)

video\_entry.grid(row=10, column=2, sticky=N)

rootWindow.grid\_rowconfigure(11, minsize=20)

message\_label.grid(row=12, columnspan=4)

message\_txt.grid(row=13, columnspan=4, sticky=N, rowspan=1)

rootWindow.grid\_rowconfigure(14, minsize=20)

lets\_see\_button.grid(row=15, columnspan=4)

att\_label.grid(row=16, columnspan=4)

rootWindow.mainloop()

* + File add\_eigen.py

import numpy as np

import cv2

import sys

import os

FREQ\_DIV = 15 #frequency divider for capturing training images

RESIZE\_FACTOR = 4

NUM\_TRAINING = 5

class TrainEigenFaces:

def \_\_init\_\_(self):

cascPath = "haarcascade\_frontalface\_default.xml"

self.face\_cascade = cv2.CascadeClassifier(cascPath)

self.face\_dir = 'dataset'

self.face\_name = sys.argv[1]

self.path = os.path.join(self.face\_dir, self.face\_name)

if not os.path.isdir(self.path):

os.mkdir(self.path)

self.model = cv2.face.EigenFaceRecognizer\_create()

self.count\_captures = 0

self.count\_timer = 0

def capture\_training\_images(self):

video\_capture = cv2.VideoCapture(0)

while True:

self.count\_timer += 1

ret, frame = video\_capture.read()

inImg = np.array(frame)

outImg = self.process\_image(inImg)

cv2.imshow('Video', outImg)

# When everything is done, release the capture on pressing 'q'

if cv2.waitKey(1) & 0xFF == ord('q'):

video\_capture.release()

cv2.destroyAllWindows()

return

def process\_image(self, inImg):

frame = cv2.flip(inImg,1)

resized\_width, resized\_height = (100, 100)

if self.count\_captures < NUM\_TRAINING:

color = cv2.cvtColor(frame, 0)

color\_resized = cv2.resize(color, (int(color.shape[1]/RESIZE\_FACTOR), int(color.shape[0]/RESIZE\_FACTOR)))

faces = self.face\_cascade.detectMultiScale(

color\_resized,

scaleFactor=1.1,

minNeighbors=5,

minSize=(30, 30),

flags=cv2.CASCADE\_SCALE\_IMAGE

)

if len(faces) > 0:

areas = []

for (x, y, w, h) in faces:

areas.append(w\*h)

max\_area, idx = max([(val,idx) for idx,val in enumerate(areas)])

face\_sel = faces[idx]

x = face\_sel[0] \* RESIZE\_FACTOR

y = face\_sel[1] \* RESIZE\_FACTOR

w = face\_sel[2] \* RESIZE\_FACTOR

h = face\_sel[3] \* RESIZE\_FACTOR

face = color[y:y+h, x:x+w]

face\_resized = cv2.resize(face, (resized\_width, resized\_height))

img\_no = sorted([int(fn[:fn.find('.')]) for fn in os.listdir(self.path) if fn[0]!='.' ]+[0])[-1] + 1

if self.count\_timer%FREQ\_DIV == 0:

cv2.imwrite('%s/%s.png' % (self.path, img\_no), face\_resized)

self.count\_captures += 1

print("Captured image: ", self.count\_captures)

cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 3)

cv2.putText(frame, self.face\_name, (x - 10, y - 10), cv2.FONT\_HERSHEY\_PLAIN, 1,(0, 255, 0))

elif self.count\_captures == NUM\_TRAINING:

print("Data telah ditambahkan ke "+self.face\_name+". Tekan 'q' untuk mengakhiri program.")

self.count\_captures += 1

return frame

if \_\_name\_\_ == '\_\_main\_\_':

trainer = TrainEigenFaces()

trainer.capture\_training\_images()

print("Silahkan memilih menu kembali")

* + File t\_eigen.py

import numpy as np

import cv2

import sys

import os

import json

RESIZE\_FACTOR = 4

folder\_path = 'D:\\Teknik Informatika\\CV\\Project Akhir\\'

in\_path = folder\_path+'dataset\\'

out\_path = folder\_path+'dataset\_haar\\'

listds = os.listdir(in\_path)

number\_files = len(listds)

os.chdir(in\_path)

arr = os.listdir()

class TrainEigenFaces:

def \_\_init\_\_(self):

print("Dataset = " + in\_path)

print("Jumlah file dataset = " + str(number\_files))

# Looping untuk DATASET

self.face\_cascade = cv2.CascadeClassifier(folder\_path+'haarcascade\_frontalface\_default.xml')

self.model = cv2.face.EigenFaceRecognizer\_create()

def capture\_training\_images(self):

while True:

for (subdirs, dirs, files) in os.walk(in\_path):

for subdir in dirs:

img\_path = os.path.join(in\_path, subdir)

img\_no=1

resized\_width, resized\_height = (112, 92)

for img\_name in os.listdir(img\_path):

img\_file = img\_path+'/'+img\_name

image = cv2.imread(img\_file)

self.process\_image(np.array(image), subdir, img\_no)

img\_no+=1

return

def process\_image(self, inImg, path, img\_no):

frame = cv2.flip(inImg,1)

resized\_width, resized\_height = (100, 100)

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

gray\_resized = cv2.resize(gray, (int(gray.shape[1]/RESIZE\_FACTOR), int(gray.shape[0]/RESIZE\_FACTOR)))

faces = self.face\_cascade.detectMultiScale(

gray\_resized,

scaleFactor=1.1,

minNeighbors=5,

minSize=(30, 30),

flags=cv2.CASCADE\_SCALE\_IMAGE

)

if len(faces) > 0:

areas = []

for (x, y, w, h) in faces:

areas.append(w\*h)

max\_area, idx = max([(val,idx) for idx,val in enumerate(areas)])

face\_sel = faces[idx]

x = face\_sel[0] \* RESIZE\_FACTOR

y = face\_sel[1] \* RESIZE\_FACTOR

w = face\_sel[2] \* RESIZE\_FACTOR

h = face\_sel[3] \* RESIZE\_FACTOR

face = gray[y:y+h, x:x+w]

face\_resized = cv2.resize(face, (resized\_width, resized\_height))

cv2.imwrite('%s/%s.png' % (out\_path+path, img\_no), face\_resized)

print("Dataset : ", path, img\_no)

# print("Training data ini berhasil. Tekan 'q' untuk berhenti disini")

return frame

def eigen\_train\_data(self):

imgs = []

tags = []

index = 0

for (subdirs, dirs, files) in os.walk(out\_path):

for subdir in dirs:

img\_path = os.path.join(out\_path, subdir)

for fn in os.listdir(img\_path):

path = img\_path + '/' + fn

tag = index

imgs.append(cv2.imread(path, 0))

tags.append(int(tag))

index += 1

(imgs, tags) = [np.array(item) for item in [imgs, tags]]

self.model.train(imgs, tags)

self.model.save(folder\_path+'eigen\_trained\_data.json')

print("Training Dataset Selesai")

return

def show\_json(self):

os.chdir(folder\_path)

with open('eigen\_trained\_data.json', 'r') as f:

mahasiswa = json.load(f)

print("Ini adalah JSON EigenValues : \n" + json.dumps(mahasiswa['opencv\_eigenfaces']['eigenvalues'], indent=4))

return

if \_\_name\_\_ == '\_\_main\_\_':

trainer = TrainEigenFaces()

trainer.capture\_training\_images()

trainer.eigen\_train\_data()

trainer.show\_json()

* + File f\_eigen.py

import numpy as np

import cv2

import sys

import os

RESIZE\_FACTOR = 4

folder\_path = 'D:\\Teknik Informatika\\CV\\Project Akhir\\'

class RecogEigenFaces:

def \_\_init\_\_(self):

cascPath = "haarcascade\_frontalface\_default.xml"

self.face\_cascade = cv2.CascadeClassifier(cascPath)

self.out\_dir = folder\_path+'dataset\_haar\\'

self.model = cv2.face.EigenFaceRecognizer\_create()

self.face\_names = []

def load\_trained\_data(self):

names = {}

key = 0

for (subdirs, dirs, files) in os.walk(self.out\_dir):

for subdir in dirs:

names[key] = subdir

key += 1

self.names = names

self.model.read(folder\_path+'eigen\_trained\_data.json')

def show\_video(self):

video\_capture = cv2.VideoCapture(0)

while True:

ret, frame = video\_capture.read()

inImg = np.array(frame)

outImg, self.face\_names = self.process\_image(inImg)

cv2.imshow('Face Recognition', outImg)

# Jika selesai tekan 'q'

if cv2.waitKey(1) & 0xFF == ord('q'):

video\_capture.release()

cv2.destroyAllWindows()

return

def process\_image(self, inImg):

frame = cv2.flip(inImg,1)

resized\_width, resized\_height = (100, 100)

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

gray\_resized = cv2.resize(gray, (int(gray.shape[1]/RESIZE\_FACTOR), int(gray.shape[0]/RESIZE\_FACTOR)))

faces = self.face\_cascade.detectMultiScale(

gray\_resized,

scaleFactor=1.1,

minNeighbors=5,

minSize=(30, 30),

flags=cv2.CASCADE\_SCALE\_IMAGE

)

persons = []

for i in range(len(faces)):

face\_i = faces[i]

x = face\_i[0] \* RESIZE\_FACTOR

y = face\_i[1] \* RESIZE\_FACTOR

w = face\_i[2] \* RESIZE\_FACTOR

h = face\_i[3] \* RESIZE\_FACTOR

face = gray[y:y+h, x:x+w]

face\_resized = cv2.resize(face, (resized\_width, resized\_height))

confidence = self.model.predict(face\_resized)

if confidence[1]<3500:

person = self.names[confidence[0]]

cv2.rectangle(frame, (x,y), (x+w, y+h), (255, 0, 0), 3)

cv2.putText(frame, '%s - %.0f' % (person, confidence[1]), (x-10, y-10), cv2.FONT\_HERSHEY\_PLAIN,1,(0, 255, 0))

else:

person = 'Tidak Dikenal'

cv2.rectangle(frame, (x,y), (x+w, y+h), (0, 0, 255), 3)

cv2.putText(frame, person, (x-10, y-10), cv2.FONT\_HERSHEY\_PLAIN,1,(0, 255, 0))

persons.append(person)

return (frame, persons)

if \_\_name\_\_ == '\_\_main\_\_':

recognizer = RecogEigenFaces()

recognizer.load\_trained\_data()

print ("Tekan 'q' untuk mengakhiri program")

recognizer.show\_video()

* + File fv\_eigen.py

import numpy as np

import cv2

import sys

import os

RESIZE\_FACTOR = 4

folder\_path = 'D:\\Teknik Informatika\\CV\\Project Akhir\\'

class RecogEigenFaces:

def \_\_init\_\_(self):

cascPath = "haarcascade\_frontalface\_default.xml"

self.face\_cascade = cv2.CascadeClassifier(cascPath)

self.face\_dir = folder\_path+'dataset\_haar\\'

self.video\_file=sys.argv[1]

self.model = cv2.face.EigenFaceRecognizer\_create()

self.face\_names = []

def load\_trained\_data(self):

names = {}

key = 0

for (subdirs, dirs, files) in os.walk(self.face\_dir):

for subdir in dirs:

names[key] = subdir

key += 1

self.names = names

self.model.read(folder\_path+'eigen\_trained\_data.json')

def show\_video(self):

video\_capture = cv2.VideoCapture(self.video\_file)

frame\_width = int(video\_capture.get(3))

frame\_height = int(video\_capture.get(4))

out = cv2.VideoWriter('output.avi',cv2.VideoWriter\_fourcc('M','J','P','G'), 10, (frame\_width,frame\_height))

while True:

ret, frame = video\_capture.read()

if ret == True:

inImg = np.array(frame)

outImg, self.face\_names = self.process\_image(inImg)

cv2.imshow('Face Recognition', outImg)

out.write(outImg)

# Jika selesai tekan 'q'

if cv2.waitKey(1) & 0xFF == ord('q'):

video\_capture.release()

out.release()

cv2.destroyAllWindows()

return

else:

return

def process\_image(self, inImg):

frame = cv2.flip(inImg, 1)

resized\_width, resized\_height = (100, 100)

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

gray\_resized = cv2.resize(gray, (int(gray.shape[1]/RESIZE\_FACTOR), int(gray.shape[0]/RESIZE\_FACTOR)))

faces = self.face\_cascade.detectMultiScale(

gray\_resized,

scaleFactor=1.1,

minNeighbors=5,

minSize=(30, 30),

flags=cv2.CASCADE\_SCALE\_IMAGE

)

persons = []

for i in range(len(faces)):

face\_i = faces[i]

x = face\_i[0] \* RESIZE\_FACTOR

y = face\_i[1] \* RESIZE\_FACTOR

w = face\_i[2] \* RESIZE\_FACTOR

h = face\_i[3] \* RESIZE\_FACTOR

face = gray[y:y+h, x:x+w]

face\_resized = cv2.resize(face, (resized\_width, resized\_height))

confidence = self.model.predict(face\_resized)

if confidence[1]<3500:

person = self.names[confidence[0]]

cv2.rectangle(frame, (x,y), (x+w, y+h), (255, 0, 0), 3)

cv2.putText(frame, '%s - %.0f' % (person, confidence[1]), (x-10, y-10), cv2.FONT\_HERSHEY\_PLAIN,1,(0, 255, 0))

else:

person = 'Tidak Dikenal'

cv2.rectangle(frame, (x,y), (x+w, y+h), (0, 0, 255), 3)

cv2.putText(frame, person, (x-10, y-10), cv2.FONT\_HERSHEY\_PLAIN,1,(0, 255, 0))

persons.append(person)

return (frame, persons)

if \_\_name\_\_ == '\_\_main\_\_':

recognizer = RecogEigenFaces()

recognizer.load\_trained\_data()

print ("Tekan 'q' untuk mengakhiri program")

recognizer.show\_video()

File – file yang berada di folder yang filenya dijalankan

