TEHNIČKI FAKULTET – RIJEKA

Diplomski studij računarstva

Dokumentacija

Cloud Face Recognition

Projekt I

Kolegij:Programiranje ugradbenih sustava

Mentor: izv. prof. dr. sc. Kristijan Lenac dipl. ing.

Dominik Beževan

006925878039

Rijeka, kolovoz 2016.

Sadržaj:

1 Uvod

Tema ovog projekta u sklopu kolegija Programiranje ugradbenih sustava bila je realizirati face recognition koristeći dva zasebna cloud API-ja na.

Ideja jest da se čitaju frameovi livefeeda kamere, te se na tim frameovima vrši detekcija lica. Ukoliko se lice detektira, pokreće se prepoznavanje tog lica. Prepoznavanje lica podrazumijeva uspoređivanje nepoznatog, detektiranog lica s unaprijed istreniranim modelom lica poznate osobe. Funkcija prepoznavanja vraća pouzdanost da je detektirano lice zaista lice poznate osobe odnosno istreniranog modela. Detekcija lica realizirat će se koristeći OpenCV knjižnicu funkcija, dok će se prepoznavanje odvijati na cloud servisu, pri ćemu će biti ponuđeno korištenje dva zasebna API-ja: Face++ i FaceR. Ukratko će biti prezentiran način rada dvaju API-ja te njihove prednosti i mane.

Programski kod predviđen je za pokretanje na nekoj od linux distribucija, konkretnom slučaju na Rasbian Jessie OS-u, odnosno Raspberry Pi-ju, uz prethodno podignutu OpenCV knjižnicu funkcija.

2 Opis alata

Osnovni alati koji su korišteni za realizaciju projekta jesu OpenCV, Face++ Cloud API te FaceR Cloud API. Svaki od navedenih alata bit će opisani u narednim potpoglavljima.

2.1 OpenCV v2.4.10

OpenCV je open-source knjižnica funkcija koje se baziraju na real-time image processing. Knjižnica je cross-platform i stoga pogodna za korištenje na Raspberry PI-ju. Za potrebe projekta, koristit će se Haar cascade klasifikator za detekciju lica. Pristup se bazira na strojnom učenju, a radi po principu da se klasifikator najprije istrenira iz velike količine pozitivnih (sadrže objekt) i negativnih slika (ne sadrže objekt), a potom se koristi za detekciju tog istog objekta na drugim slikama. U ovom projektu koristit će se unaprijed istrenirani klasifikator (haarcascade_frontalface_default.xml), koji je dostupan lokalno, nakon instalacije OpenCV knjižnice (opencv/data/haarcascades/**haarcascade_frontalface_default.xml**).

Osim klasifikatora, za potrebe projekta koristit će se cv2 funkcije za manipulaciju frameovima odnosno slikama i inputom kamere.

2.2 Face++ Cloud API + OpenCV

Face++ (Megvii Inc.) je kineska computer vision i data mining platforma koja se sastoji od 3 ključne usluge (Detection, Recognition i Analysis). Platforma je besplatna, može se koristiti za testiranje i za manje nekomercijalne aplikacije. Mjesečno ograničenje uporabe je 50.000 API poziva. Demo testiranje dostupno je na linku http://www.faceplusplus.com/demo-detect/.

Korištenje API-ja sastoji se od treniranja modela i prepoznavanja. Treniranje je pripremna faza i nije dio algoritma na kojem se projekt bazira. Projekt se bazira na drugom dijelu, odnosno prepoznavanju, te se tek u njemu može usporediti koliko je koji algoritam efikasan i efektivan.

2.2.1 Basic Workflow

TRENIRANJE
1. Cloud: Create person – stvaranje praznog modela
2. Cloud: Detect – detekcija lica
3. Cloud: Add face – dodavanje detektiranog lica u stvoreni model
4. Cloud: Train – treniranje modela nakon što se sva željena lica dodaju u model.

PREPOZNAVANJE
5. Local: Haar Classifier – OpenCV detekcija lica.
6. Recognize – Usporedba detektiranog nepoznatog lica s istreniranim modelom osobe (putrem Enroll funkcije).

2.3 FaceR Cloud API + OpenCV

FaceR (tvrtka Animetrics) je američka platforma koja nudi detekciju i prepoznavanje lica. Platforma je freemium, što znači da je komercijalna i nudi više tarifa, ali nudi i besplatnu opciju koja dopušta do 1000 API poziva mjesečno.

2.3.1 Basic Workflow

TRENIRANJE

1. Cloud: Detect – Detekcija lica na ulaznoj slici.
2. Cloud: Enroll – Pohranjivanje jednom detektiranih lica u galeriju (nekoj osobi).

PREPOZNAVANJE

3. Local: Haar Classifier – OpenCV detekcija lica.
4. Cloud: Detect - Detekcija lica na ulaznoj slici.
5. Cloud: Recognize – Usporedba detektiranog nepoznatog lica s istreniranim modelom osobe (putrem Enroll funkcije).

3 Opis koda

Kod se sastoji od jedne glavne skripte koja generira GUI, te od 3 podskripte od kojih jedna orkestrira OpenCV Haar klasifikatorom, druga Face++ API-jem, a treća FaceR API-jem.
Najvažniji dijelovi koda će biti prikazan i opisani u narednim potpoglavljima, a kompletan kod kometiran je i priložen je u folderu uz dokumentaciju.

3.1 SKRIPTA 1: face_recognition_GUI

Uloga ove skripte je stvaranje i prikaz GUI-ja.
Za izradu GUI-ja koristi se Tkinter, a osim njega importaju se podskripte facer i faceplusplus iz kojih će se pozivati opencv_detect funkcija.

```
from Tkinter import Tk, BOTH, IntVar
from ttk import Frame, Button, Radiobutton, Style
from PIL import Image, ImageTk
import tkMessageBox
import Tkinter
import facer
import faceplusplus
import cv2
```

U initUI(self) funkciji definiraju se sve vizualne komponente i povezuju s odgovarajućim metodama odnosno akcijama. Komponente su: dva radiobuttona (odabir API-ja), tri buttona („Start Recognition“, “ Help“, “Quit“).

```
def initUI(self):

        photo = ImageTk.PhotoImage(file='icon/main_bg_4.jpg')
        w = Tkinter.Label(self,image=photo)
        w.photo = photo
        w.place(x=0, y=0, relwidth=1, relheight=1)
        #ackground_label.pack()


        self.api = IntVar()
        self.parent.title("Face Recognition System")
        self.style = Style()
        self.style.theme_use("default")

        self.pack(fill=BOTH, expand=1)

        faceplus = Radiobutton(self, text="Face++", variable=self.api, value=1)
        faceplus.place(x=50, y=40)

        facer = Radiobutton(self, text="FaceR Animetrics", variable=self.api, value=2)
        facer.place(x=200, y=40)

        start = Button(self, text="Start Recognition",
            command=self.startRecognition)
        start.place(x=100, y=80)

        helpButton = Button(self, text="Help",
            command=self.giveHelp)
        helpButton.place(x=100, y=110)

        quitButton = Button(self, text="Quit",
            command=self.quitGUI)
        quitButton.place(x=100, y=140)
```

Funkcija startRecognition(self) pokreće jedan od dva moguća API-ja, ovisno o vrijednosti dobivenoj iz radiobuttona.

```
def startRecognition(self):

        if self.api.get() == 1:
                faceplusplus.start_recognition()
        if self.api.get() == 2:
                facer.start_recognition()
```

Funkcija giveHelp(self) otvara prozor s osnovnim uputama za korištenje programa.

```
def giveHelp(self):

        lines = ['1.) Choose one API', '2.) Click on "Start Recognition"', '3.) Type "q" or
press "Quit" to exit program.']
        tkMessageBox.showinfo('Instructions', "\n".join(lines))
```

Funkcija quitGUI(self) zatvara GUI i gasi program.

```
def quitGUI(self):

        self.destroy()
        exit()
```

3.2 SKRIPTA 2: opencv_detect

Ova podskripta se koristi za pozivanje cascade klasifikatora s kojim se vrši detekcija lica na ulaznom frameu koji je zadan argumentom img. Ukoliko se lice detektira, vraća se lista rects s koordinatama lica, u protivnom se vraća prazna lista.

```
import cv2

def detect(img):

        cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
        rects = cascade.detectMultiScale(img, 1.3, 4, cv2.cv.CV_HAAR_SCALE_IMAGE, (20,20))

        if len(rects) == 0:
                return []

        return rects
```

3.3 SKRIPTA 3: faceplusplus

Ovom podskriptom otvara se input kamere te se svaki 30. frame provjerava sadrži li lice pozivom opencv_detect(img) funkcije. Ukoliko sadrži, šalje se API recognition poziv Face++ servisu, koji vraća ime osobe i postotak pouzdanosti da se zaista radi o toj osobi.ž

Face++ posjeduje Python SDK, facepp.py, koji olakšava pisanje API poziva. Stoga se iz njega importaju određeni segmenti.

```
import glob
import unirest
import time
import os
import cv2
import opencv_detect
import ctypes
from facepp import API, File
```

Za slanje API poziva potrebne su određene kredencije, API_KEY i API_SECRET.

```
SERVER = 'http://api.us.faceplusplus.com/'
API_KEY = '4c764c549faf0186ba0377f829411722'
API_SECRET = 'P4E7XwSTt50RkX2JFyRuBRNIk2mPDWkg'

api = API(API_KEY, API_SECRET, SERVER)
```

Na početku funkcije otvara se input port kamere, definira se prag pouzdanosti (threshold) koji će nakon prepoznavanja odrediti radi li se zaista o toj osobi i importaju se dekorativne slike.

```
def start_recognition():

        cap = cv2.VideoCapture(0)
        frame_num = 0
        threshold = 70
        search_icon = cv2.imread("icon/waiting_2.jpg")
        access_icon = cv2.imread("icon/approvedfont.jpg")
        no_access_icon = cv2.imread("icon/notapprovedfont.jpg")
```

Beskonačna while petlja koristi se za čitanje frameova livefeed kamere te detekciju i eventualno prepoznavanje lica na frameu na svakom 30. frameu. Na taj način algoritam radi brže, dakle povećava se efikasnost, a pritom se ne smanjuje se efektivnost.

```
        while(1):
                frame_num += 1
                _, frame = cap.read()
                key = cv2.waitKey(5) & 0xFF
                cv2.imshow('Face Recognition System (livefeed)', frame)
                if frame_num == 30:
                        frame_num = 0
                        rects = []
                        rects=opencv_detect.detect(frame)
                        if len(rects):
                                cv2.imshow('Face Recognition System (livefeed)', search_icon)
                                key = cv2.waitKey(5) & 0xFF
                                print "Face detected!"
                                cv2.imwrite('frame.jpg', frame)
                                print "Starting recognition of the detected face."
                                response = api.recognition.identify(group_name='demo',
img=File(r'frame.jpg'), mode='oneface')
```

Ako je prepoznavanje vratilo rezultate, tada, ako je pouzdanost veća ili jednaka predefiniranom pragu, potvrđuje se da se zaista radi o toj osobi. U protivnom, rezultat se demantira. Prag pouzdanosti postavljen je na 70%, iako bi se trebao postaviti na 90% ili više. Razlog tome jest loša kvaliteta kamere koja se koristi za potrebe projekta, a samim time i nepredvidiv utjecaj osvjetljenja na sliku.

Nakon 3 sekunde, postupak kreće ispočetka.

```
                        if response['face']:
                                result_precision_rate =
float(response['face'][0]['candidate'][0]['confidence'])
                                if result_precision_rate >= threshold:
                                        print "Face recognized with confidence of
{0:.2f}%, access granted.".format(result_precision_rate)
                                        cv2.imshow('Face Recognition System (livefeed)',
access_icon)
                                        key = cv2.waitKey(10) & 0xFF
                                else:
                                        print "Face not recognized (confidence of
{0:.2f}%), access not granted.".format(result_precision_rate)
                                        cv2.imshow('Face Recognition System (livefeed)',
no_access_icon)
                                        key = cv2.waitKey(10) & 0xFF

                                time.sleep(3)
                        else:
                                print "Face not recognized, access not granted!"
                                cv2.imshow('Face Recognition System (livefeed)',
no_access_icon)
                                key = cv2.waitKey(10) & 0xFF

                if key == ord("q"):
                        break

        cap.release()
        cv2.destrovAllWindows()
```

3.4 SKRIPTA 4: facer

Identično prethodno, ovom podskriptom otvara se input kamere te se svaki 30. frame provjerava sadrži li lice pozivom opencv_detect(img) funkcije. Ukoliko sadrži, šalje se API detection poziv FaceR servisu, koji vraća image_id i niz parametara od kojih su bitne koordinate lica. Image_id i koordinate lica prosljeđuju se API recognition pozivu koji vraća postotak pouzdanosti da se radi o toj osobi.

Funkcija između ostalog zahtjeva importanje Unirest-a, seta HTTP knjižnica koji je stvoren i održavan od strane Mashape-a (API provider), preko kojeg se upravlja FaceR API-jem, a koji omogućuje stvaranje requesta odnosno komunikaciju s navedenim API-jem.

```
import glob
import unirest
import time
import os
import cv2
import opencv_detect
import ctypes
from facepp import API, File
```

Ukoliko Haar classifier detektira lice, šalje se API detect poziv koji vraća određeni broj parametara.

```
if len(rects):
        print "Face detected!"
        cv2.imshow('Face Recognition System (livefeed)', search_icon)
        key = cv2.waitKey(5) & 0xFF
        cv2.imwrite('frame.jpg', frame)

        try:
                response =
unirest.post("https://animetrics.p.mashape.com/detect?api_key=4adf13e6b744dee183476cc0574a7
d6f",
                 headers={"X-Mashape-Key":
"Ugdk0HPDgLmshEOPogyrGTxr9B4Dp1h7TSxjsng80cdpz6amiI"
                 },
                 params={
                        "image": open("frame.jpg", mode="rb"),
                        "selector": "FULL"
                 }
                )
                result = response.body
```

Od dobivenih parametara uzimaju se koordinate lica i image_id, koji predstavljaju ulazne argumente za API recognition poziv.

```
if result['images'][0]['faces']:

        image_id=result['images'][0]['image_id']
        width=result['images'][0]['faces'][0]['width']
        height=result['images'][0]['faces'][0]['height']
        topLeftX=result['images'][0]['faces'][0]['topLeftX']
        topLeftY=result['images'][0]['faces'][0]['topLeftY']

        try:
                print "Starting recognition of the detected face."
                response =
unirest.get("https://animetrics.p.mashape.com/recognize?api_key=4adf13e6b744dee183476cc0574a7d6f&galler
y_id=demo&height={0}&image_id={1}&topLeftX={2}&topLeftY={3}&width={4}".format(height,image_id,topLeftX,
topLeftY,width),
                 headers={
                        "X-Mashape-Key": "Ugdk0HPDgLmshEOPogyrGTxr9B4Dp1h7TSxjsng80cdpz6amiI",
                        "Accept": "application/json"
                 }
                 )
```

Konačno, ako je prepoznavanje vratilo rezultate, tada, ako je pouzdanost veća ili jednaka predefiniranom pragu, potvrđuje se da se zaista radi o toj osobi. U protivnom, rezultat se demantira. Nakon 3 sekunde, postupak kreće ispočetka.

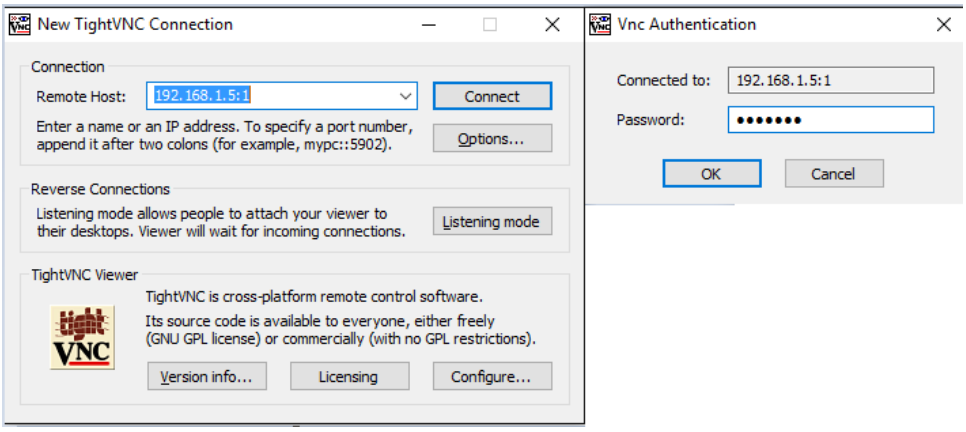
```
                result_precision_rate = float(response.body['images'][0]['candidates']['Dominik'])
                result_precision_rate *= 100

                                                # If recognition is positive, than if confidence is
>= to threshold, access is granted, match is confirmed, else access is not granted and match is not
confirmed.
                if result_precision_rate >= threshold:
                        print "Face recognized with confidence of {0:.2f}, access
granted.".format(result_precision_rate)
                        cv2.imshow('Face Recognition System (livefeed)', access_icon)
                        key = cv2.waitKey(10) & 0xFF
                else:
                        print "Face not recognized (confidence of {0:.2f}%), access not
granted.".format(result_precision_rate)
                        cv2.imshow('Face Recognition System (livefeed)', no_access_icon)
                        key = cv2.waitKey(10) & 0xFF

                time.sleep(3)
```

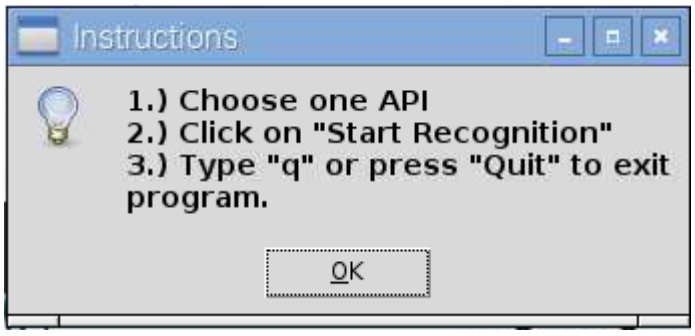
4 Prikaz rada algoritma

Za spajanje na Raspberry PI koristi se aplikacija TightVnc (slika 4.1). Potrebno je upisati IP adresu te predefiniranu lozinku.



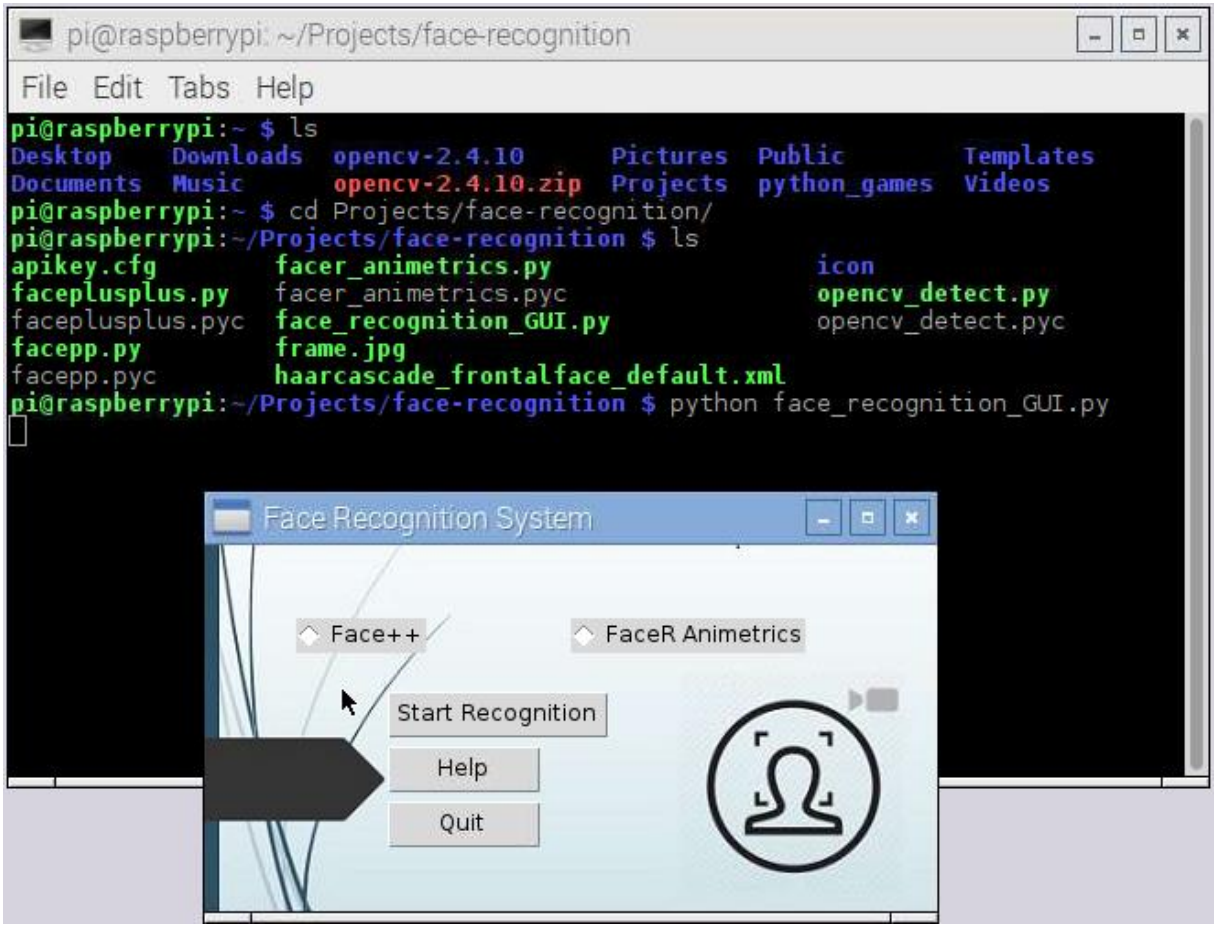
Slika 4.1: TightVNC - spajanje na Raspberry PI

Pokretanjem skripte face_recognition_GUI.py prikazuje se GUI. Klikom na „Help“ (slika 4.2) ispisuju se upute za korištenje. Na cloud servisima Face++ i FaceR nalazi se jedan istrenirani subjekt („Dominik“) unutar „demo“ galerije. U svrhu treniranja navedenog subjekta koristilo se par stotina slika s prikazom lica te osobe.



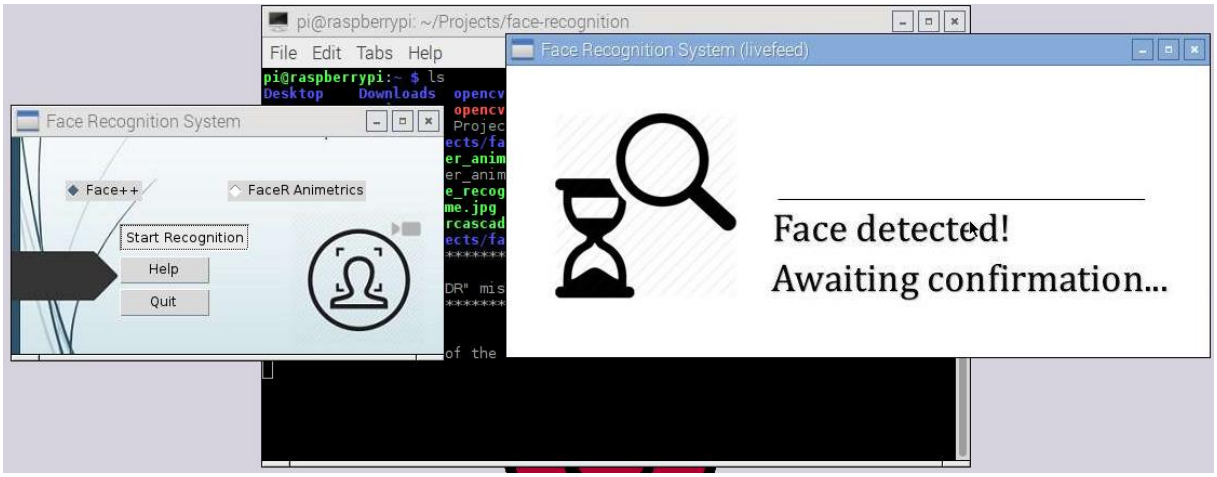
Slika 4.2: Prikaz uputa klikom na „Help“

Odabirom jednog od API-ja može se pokrenuti prepoznavanje. Prikaz GUI-ja može se vidjeti na slici 4.3.



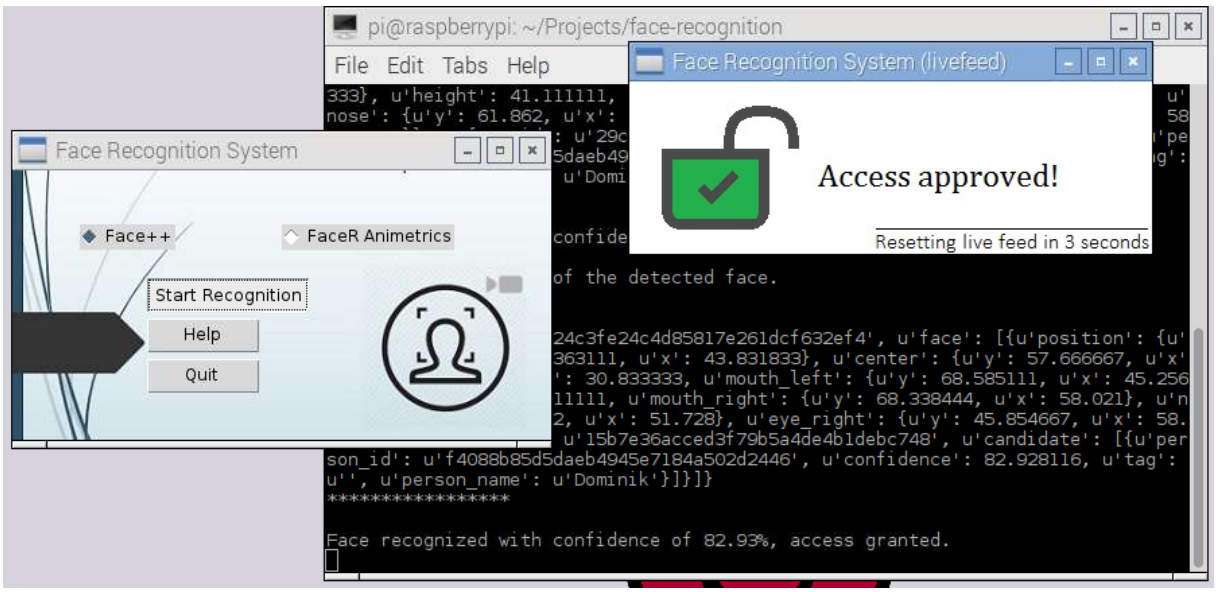
Slika 4.3:Prikaz GUI-ja

U trenutku kad OpenCV detektira lice, na zaslonu se pojavljuje slijedeća obavijest, kako je prikazano na slici 4.4. Potrebno je pričekati nekoliko trenutaka za upload slike i stizanje rezultata.



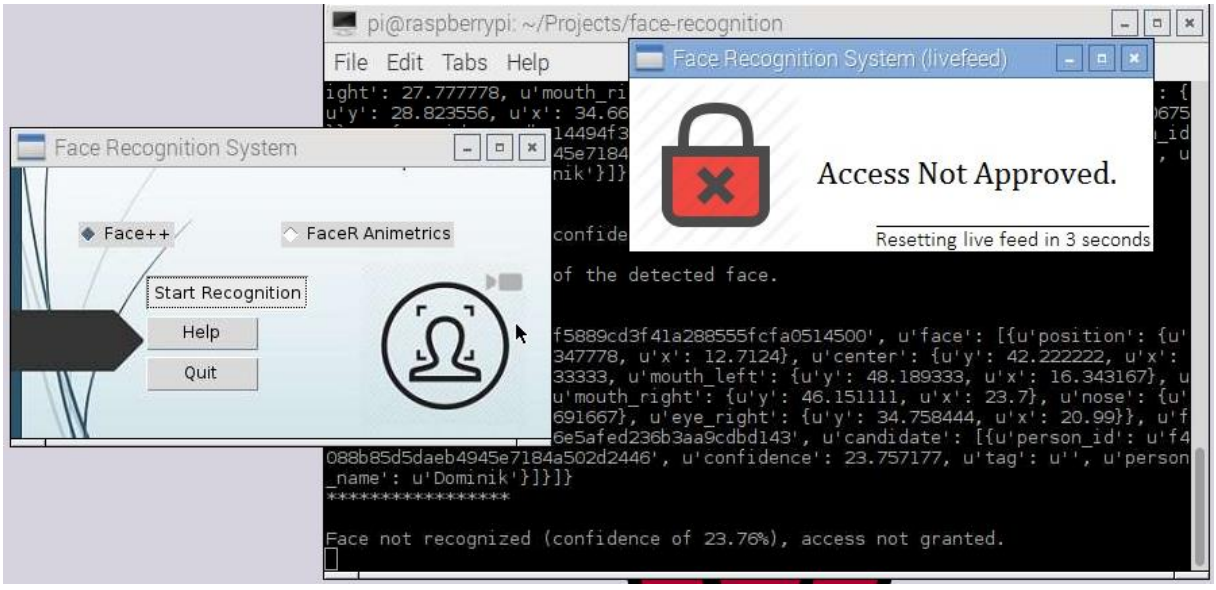
Slika 4.4: Uspješna detekcija lica

U slučaju uspješnog prepoznavanja subjekta, odnosno confidence level je veći od zadanog thresholda, prepoznavanje se smatra uspješnim (slika 4.5).



Slika 4.5: Uspješno prepoznavanje

U protivnom, prepoznavanje se smatra neuspješnim (slika 4.6).



Slika 4.6: Neuspješno prepoznavanje.

5 Zaključak

Realizirano rješenje primjer je mehanizma koji se u praksi može koristiti kao dodatni sigurnosni mehanizam za odobravanje pristupa nekoj prostoriji (otvaranje vrata). Lokalna detekcija lica, putem OpenCV knjižnica odnosno Haar Cascade Classifier pokazala se vrlo efikasnom. Detekcija lica preko cloud API-ja zahtjevala bi brzu internetsku konekciju za skoro trenutni upload frameova. Osim toga, na taj bi se način izvodio velik broj API poziva, što bi potencijalno rezultiralo puno većim troškom, s obzirom da FaceR API ima, kao i velik broj drugih online sustava za prepoznavanje, ograničen broj besplatnih API poziva.

U međusobnoj usporedbi dvaju korištenih Cloud servisa, Face++ je puno pristupačnija opcija, s obzirom da besplatna i i nudi velik broj API poziva (mjesečno 50.000 poziva), za razliku od FaceR-a, kod kojeg je besplatno samo 1000 API poziva mjesečno.
Kad se govori o kompleksnosti dvaju API-a, činjenica jest da je kod FaceR-a nespretno realizirano prepoznavanje lica – kako bi se prepoznavanje putem navedenog API-ja uopće i moglo pokrenuti, nakon OpenCV detekcije lica na frameu nužno je pokrenuti i detekciju na samom API-ju jer se tako generira jedinstveni image_id koji je jedan od obaveznih parametara za pokretanje prepoznavanja. Na taj način developerima se jako ograničava mogućnost kombiniranja navedenog API-ja s nekim drugim alatima za detekciju lica.
Facer stoga značajno zaostaje u brzini izvođenja u odnosu na Face++ jer za prepoznavanje uvijek izvodi 1 API poziv više od Face++ API-ja kojemu se za prepoznavanje lica jednostavno proslijede koordinate jednom detektiranog lica putem bilo kojeg drugog alata (u ovom slučaju OpenCV knjižnica).

Najviše se najviše vremena troši na upload frameova, dok je brzina izračuna operacija na uploadanom frameu zanemariva. Ovaj projekt pokazao je kako je tehnologija detekcije i prepoznavanja lica putem Cloud servisa vrlo uporabljiva, ali je pritom nužno imati dobru internet povezanost. Također, potvrđuje se kako komercijalni sustavi nisu nužno bolji u odnosu na dostupnu besplatnu alternativu.