VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

RASPBERRY PI: INTELIGENTNÍ DVEŘNÍ KUKÁTKO

RASPBERRY PI: INTELLIGENT DOOR PEEPHOLE

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Adam Miček

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Radomil Matoušek, Ph.D.

SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Adam Miček**
Studijní program:	Strojní inženýrství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**doc. Ing. Radomil Matoušek, Ph.D.**
Akademický rok:	2018/19

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Raspberry Pi: inteligentní dveřní kukátko

Stručná charakteristika problematiky úkolu:

S využitím prostředí Matlab/Simulink (MATLAB® Support Package for Raspberry Pi® Hardware) nebo přímým programováním bude vytvořena hw–sw platforma inteligentního dveřního kukátka postavenána platformě Raspberry Pi 3. Platforma Raspberry s příslušenstvím i sw jsou k dispozici.

Cíle diplomové práce:

1. Seznámení s problematikou programování platformy Raspberry Pi.
2. Po konzultaci se školitelem navrhnout HW platformu inteligentního dveřního kukátka.
3. SW realizace zahrnující: automatické snímání příchozí osoby, zpracování a ukládání získané informace, dálková zpráva zařízení.

Seznam doporučené literatury:

UPTON Eben; HALFACREE Gareth: Raspberry Pi, COMPUTER PRESS, 2016. Martin Stříž, Bučovice, 2015. 280 s. ISBN: 978-80-251-4819-8

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2018/19

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Cieľom tejto záverečnej práce je vytvorenie funkčného prototypu inteligentného priezoru (tzv. inteligentného video vrátnika) na platforme Raspberry Pi 3B+, ktorý bude umožňovať rozpoznávanie osôb na základe ich tváre, prípadne hesla vo forme QR kódu a vytvárať záznam vo forme fotiek, videa a textu. Ďalej bude umožňovať diaľkovú správu prostredníctvom Android aplikácie s možnosťou sledovania živého videa z Raspberry Pi, priamo v tejto aplikácií. Tvorbe projektu predchádza stručný úvod k platforme Raspberry Pi, jej alternatívam a procesu rozpoznávania tváre. Pri tvorbe softwaru pre Raspberry bol využitý primárne jazyk Python. Použité sú dva prístupy spracovávania obrazu priamo na Raspberry a na výkonnejšom počítači. Práca popisuje použitie viacerých moderných knižníc, platforiem a rámcov, implementovaných počas tvorby projektu, so snahou poskytnúť začiatočníkovi v tejto oblasti náhľad na tieto technológie, ich využitie a taktiež základ možnej ďalšej práce na nich.

Abstract

The goal of this diploma thesis is to create the functional prototype of intelligent peephole (video doorman) on Raspberry Pi 3B+ platform. It will support face recognition of the people, alternatively authorization by QR code and the record of the situations in form of picture, video, and text. Furthermore, it will allow remote control from Android application with possibility to see live stream of Raspberry Pi's video in this application. Creation of the project is preceded by brief introduction of Raspberry Pi platform, its alternatives and process of face recognition. Python programming language was primarily used for software development on Raspberry Pi platform. Two approaches were used for image processing, direct processing on Raspberry Pi and on more powerful computer. This work describes use of various modern libraries, platforms, and frameworks, used during the creation of the project, with goal to provide beginners in this field insight into this technology, way of its use, and offer starting point for further work.

Kľúčové slová

Raspberry Pi 3B+, jednodoskový počítač, inteligentný priezor, Python, rozpoznávanie tváre

Keywords

Raspberry Pi 3B+, single board computer, intelligent peephole, Python, face recognition

Bibliografická citácia

MIČEK, Adam. *Raspberry Pi: inteligentní dveřní kukátko* [online]. Brno, 2020 [cit. 2020-06-26]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/121628. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky. Vedoucí práce Radomil Matoušek.

Čestné prehlásenie

Prehlasujem, že túto diplomovú prácu som vypracoval samostatne, pod vedením doc. Ing. Radomil Matoušek, Ph.D a všetky použité zdroje sú uvedené v zozname použitej literatúry. Vyhlasujem, že podľa môjho najlepšieho vedomia a svedomia, sú všetky informácie v tejto práci pravdivé.

V Brne dňa 26.6.2020 ……

Bc. Adam Miček

Poďakovanie

Rád by som poďakoval vedúcemu mojej diplomovej práce, doc. Ing. Rodomilovi Matouškovi, Ph.D. za odborné vedenie, poskytnutý hardware, cenný čas, podporu a rady, ktoré viedli k úspešnému vytvoreniu tejto práce. Ďalej by som chcel poďakovať svojej rodine, za preukázanú podporu počas celého života, trpezlivosť a lásku.

Obsah

1 Úvod 15
2 Raspberry Pi 17
2.1 Vývoj 18
2.2 Raspbian 20
2.3 Alternatívy k Raspberry Pi 21
3 Analýza a návrh 27
3.1 Návrh systému 27
3.2 Hardware 28
3.2.1 Kamera 29
3.2.2 Reproduktor 30
3.2.3 Traffic HAT 31
3.3 Software 31
3.3.1 Python 32
3.3.2 OpenCV 32
3.3.3 Dlib 33
3.4 Výber ďalšieho softvéru 33
3.4.1 Flask 33
3.4.2 ZeroMQ 34
3.4.3 FireBase 35
4 Teoretický základ 37
4.1 Proces rozpoznávanie tváre 37
4.2 Detekcia 38
4.3 Natočenie tváre 39
4.4 Reprezentácia 40
4.5 Klasifikácia 42
4.6 Ladenie parametrov 43
5 Konfigurácia 45
5.1 Inštalácia Raspbianu 45
5.2 Práca s Raspberry 46
5.3 Virtuálne prostredia 49
5.4 Inštalácia OpenCV 50
5.5 Inštalácia Dlib 52
5.6 Inštalácia dodatočných knižníc 53
5.7 „Offloading“ 53
5.8 Statická IP adresa 54

6 Implementácia 57
6.1 Vytvorenie databázy a reprezentácií 57
6.2 Rozpoznávanie tvárí 59
6.3 Kontrola očí 61
6.4 Verifikácia kódom 65
6.5 Detekcia pohybu 65
6.6 Ukladanie dát 67
6.7 Streamovanie 69
6.8 Odosielanie videa na server 71
6.9 Prijatie videa na serveri 73
6.10 Komunikácia prostredníctvom „socket“ 73
6.11 „At Startup“ 74
6.12 Mobilná aplikácia 75
7 Realizácia a testovanie 79
8 Dosiahnuté ciele a možné pokračovanie 87
9 Záver 91
Bibliografická citácia 93
Zoznam skratiek, symbolov, obrázkov a tabuliek 101
Zoznam príloh 105
Príloha A : 106
Príloha B: 108

Úvod

V posledných rokoch sa otázka bezpečnosti stáva stále aktuálnejšou témou. Je to z dôvodov zvýšených nárokov na osobnú bezpečnosť, zabezpečenie osobných údajov a prístupu k nim, ochranu majetku, štátnu bezpečnosť alebo celkové zvýšenie pocitu bezpečia. Sofistikované bezpečnostné a kontrolné systémy sa stávajú stále bežnejšou súčasťou nášho každodenného života, s čím prirodzene súvisia zvýšené nároky na ich rýchlosť a spoľahlivosť. Snahou tiež je, aby tieto systémy čo najmenej obmedzovali svojich používateľov. Bližšie budú popísané systémy využívajúce rôzne metódy identifikácie a verifikácie osôb. Z hľadiska spôsobu určovania totožnosti možno rozdeliť tieto metódy na dve základné kategórie: biometrické a nebiometrické metódy (Zhao *et al.*, 2003).

Medzi nebiometrické metódy zaraďujeme používanie sériového čísla osobných dokladov, hesiel, PIN kódov, atď. Jedná sa o veľmi jednoduchú a rýchlu metódu, ktorej nevýhodou je práve možnosť jednoduchého zneužitia citlivých informácii v prípade odcudzenia, alebo straty týchto údajov (Abate *et al.*, 2007).

Druhou kategóriou sú biometrické metódy, ktoré sa zameriavajú na identifikáciu a verifikáciu osôb podľa jedinečných fyzických znakov alebo zvykových čŕt jedinca. Patrí sem skenovanie otlačkov prstov, tvaru dlane, rohovky, rozpoznávanie tváre, vlastnoručný podpis, hlasová biometria a ďalšie. Sľubné možnosti použitia, poskytnutie zvýšenej úrovne zabezpečenia a mnohé technologické výzvy spôsobili zvýšené množstvo pozornosti, ktoré je v posledných rokoch týmto metódam venované.
Ani biometrické metódy, ale neposkytujú odpovede na všetky výzvy, s ktorými sa môžeme na ceste za „dokole spoľahlivým“ systémom rozpoznávania stretnúť. Problém môže nastať v prípade, ak si niekto vytvorí napríklad maketu odtlačku nášho prsta, alebo 3D model našej tváre. „Zablokovanie“ biometrie totiž nie je také jednoduché ako zablokovanie pri odcudzenom občianskom preukaze.
Každý prístup má svoje výhody aj nevýhody. Skenovanie sietnice je veľmi spoľahlivý, ale často príliš obťažujúci a drahý spôsob verifikácie resp. identifikácie. Otlačky prstov sú spoločensky prijateľné, ale nie je možné ich používať ako spôsob vzdialenej identifikácie pri nespolupracujúcom subjekte (Abate *et al.*, 2007).

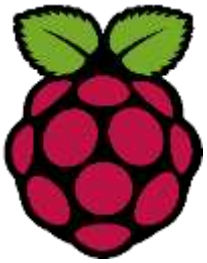
V posledných rokoch sa pozornosť vedcov z oblasti biometrie a rozpoznávania obrazu stále viac upriamuje na rozpoznávanie tvárí (Fernandez *et al.*, 2014). Je to spôsob rozpoznávania a kontroly, ktorý sa dá vykonávať na diaľku, neobťažuje a vďaka technologickému pokroku poskytuje stále lepšie výsledky. Samotné „rozpoznanie tváre“ je však až posledný krok v procese, ktorému predchádza prvotná detekcia, reprezentácia a ďalšie potrebné úkony (Zhao *et al.*, 2003). Jedná sa o komplexný proces, ktorý často komplikujú svetelné podmienky (najmä vo vonkajšom prostredí veľmi nestále), meniaci sa výraz tváre, starnutie, ochlpenie, okuliare. To je hlavný dôvod, prečo bolo v poslednej dobe vyvinuté veľké množstvo systémov, ktoré sa snažia čo najviac eliminovať tieto vplyvy (Lam, 2017).

Táto práca si kladie za cieľ vytvoriť systém rozpoznávania tváre na báze „inteligentného video vrátnika“ (priezoru, slang. kukátka), ktorý je použiteľný pre domáce, prípadne firemné účely, ale ktorý bude mať skôr informačný ako bezpečnostný charakter. Video vrátnika je možné považovať za súčasť bezpečnostných systémov, ktoré slúžia najmä k monitorovaniu svojho okolia, prichádzajúcich a odchádzajúcich osôb prípadne iného pohybu vo svojom okolí. Nakoľko v súčasnej dobe je možné ich považovať za bežnú súčasť zabezpečovacieho systému vynára sa otázka, či je daný systém spoľahlivý.

Cieľom tejto práce je vytvoriť zariadenie, ktorého prínosom, v porovnaní s bežnými „kukátkami“ (poskytujúcich video prípadne audio vstup a výstup s nutnou okamžitou reakciou zo strany prijímateľa informácie), je schopnosť samostatne spracovať a vyhodnotiť situáciu. Bude schopné automaticky monitorovať pohyb, povoliť, prípadne zamietnuť autorizáciu nepovolaných osôb, umožňovať autorizáciu na základe hesla vo forme QR kódu, vytvárať zoznam osôb a nahrávať fotky, videá a textové súbory automaticky do cloudovej služby. Vstup osôb, notifikácie, alarm, prehranie odkazu a ďalšie funkcie bude možné ovládať pomocou mobilnej aplikácie, umožňujúcej tiež sledovanie obrazu z kamery v reálnom čase.

.

Raspberry Pi



Obrázok 1: Logo Raspberry Pi (Wikipedia, 2019).

Táto kapitola sa venuje výpočtovému modulu Raspberry Pi (Obrázok 2). Stručne popisuje históriu a vývoj jednotlivých modulov, parametre a alternatívy ktoré sa na trhu nachádzajú.

Raspberry Pi je séria malých jednodoskových počítačov vyvinutých v Spojenom Kráľovstve, s cieľom poskytnúť jednoduchý a lacný prostriedok k výučbe programovania a rozvíjania počítačovej gramotnosti. Po pripojení klávesnice, myši a obrazovky sa z nich stáva plnohodnotný počítač ako každý iný.

V súčasnosti sa firma stojaca za Raspberry Pi delí na dve vetvy. Raspberry Pi Trading je zodpovedná za technologickú časť a Raspberry Pi Foundation je vzdelávacou charitou ktorá sa zameriava na výučbu informatiky v krajinách tretieho sveta (Cellan-Jones, 2011).

Obrázok 2: Model Raspberry Pi 3 B+ (Wikipedia, 2019).

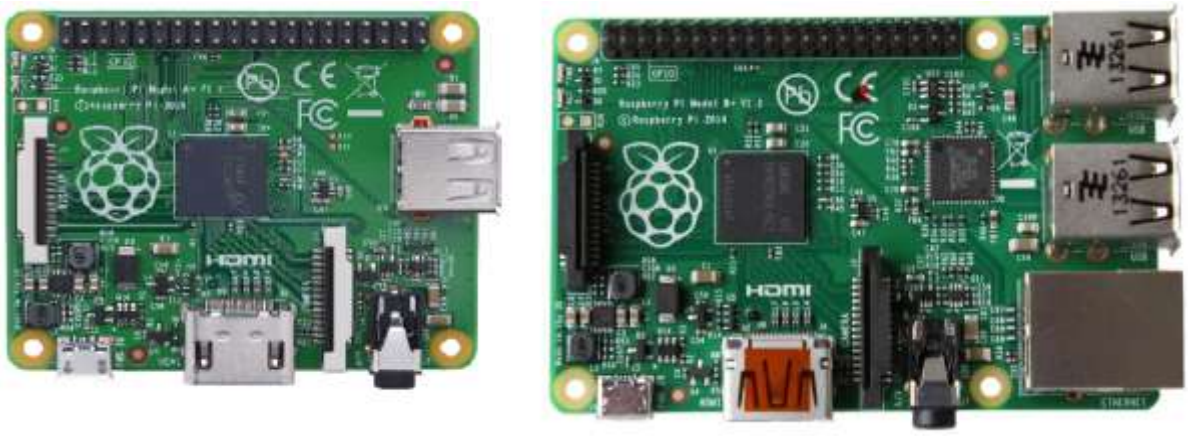
Čo sa začalo ako malý projekt sa postupne zmenilo na celosvetoví úspech, o čom svedčia aj čísla z predaja. V roku 2015 sa s počtom predaných kusov 5 miliónov Raspberry Pi stalo najlepšie predávaných britským počítacom (Gibs, 2015). V marci roku 2017 sa pri prekročení méty 12,5 milióna stalo 3. najpredávanejším univerzálnym počítačom na svete

(Miller, 2017). V marci roku 2018 bolo predaných 19 miliónov kusov (Upton, 2018) a k decembru roku 2019 hodnota prekročila 30 miliónovú métu (Tung, 2019).

1.1 Vývoj

Prvým predstaveným modelom, s ktorým sa začala písať história Raspberry, bol model Raspberry Pi 1, model B, predstavený v apríli roku 2012. Bol osadený Broadcom BCM2835 SoC s 700 MHz ARM1176JZF-S procesorom a VideoCore IV grafickou jednotkou s pamäťou 512MB. Cena bola stanovená na neuveriteľných 35$ čo bolo jedným z kľúčových faktorov úspechu tohto projektu. O rok neskôr bol predstavený model A s ešte nižšou cenou 25$ spôsobenou menšou operačnou pamäťou a len jedným USB portom, na rozdiel od dvoch pri modeli B (Lyons, 2015).

V roku 2014 bola prišla na trh vylepšená verzia oboch modelov: B+ a A+ (Obrázok 3) ktorá obsahovala 40PIN general-purpose input/output (GPIO) rozhranie namiesto predchádzajúcich 26PIN a ďalšie malé vylepšenia (Lyons, 2015).



Obrázok 3: Raspberry Pi 1 A+ (vľavo) a model B+ (vpravo)(Jarvan, 2017).

Následne s ročným odstupom vo februári roku 2015 bola uvedená na trh značne vylepšená verzia Raspberry Pi 2 model B so štvorjadrovým procesorom Cortex-A7 900 MHz a veľkosť operačnej pamäte vzrástla dvojnásobne na 1 GB zdieľaný s grafickou jednotkou. Centrálna procesorová jednotka je podľa slov výrobcu 4-6 krát výkonnejšia ako v predošlých modeloch (Fitzer, 2018).

Novým prírastkom do rodiny sa stalo v novembri 2015 Raspberry Pi Zero (Obrázok 4), ktoré prinieslo významnú redukciu rozmerov na jednu tretinu, jeden micro USB port a obrovskú výhodu nízkej spotreby energie (0,4-0,7 Watt). Pri svojej hmotnosti len 9 gramov sa naozaj jedná o minipočítač, ktorý je možné použiť v prípadoch, kde sú veľkosť a spotreba energie kľúčovými faktormi (Peterson, 2017).

Obrázok 4: Raspberry Pi Zero (Peterson, 2017).

Predposlednou predstavenou generáciou z februára roku 2016 je Raspberry Pi 3 (Obrázok 2). Disponuje navýšeným výkonom vďaka štvorjadrovému procesoru Cortex-A53 1.2 GHz a pridanou možnosťou Bluetooth a Wi-Fi pripojenia bez nutnosti použitia dodatočného hardwaru. Veľkou výhodou je taktiež plná kompatibilita s Raspberry Pi 2, čo znamená, že všetko príslušenstvo použiteľne pre Raspberry Pi 2 je použiteľne aj pre Raspberry Pi 3. V roku 2018 bolo vydané vylepšenie Raspberry Pi 3 model B+ a A+. Model B+ priniesol navýšenie eternetovej rýchlosti z predošlých 100 na 300 Mbit/s a vylepšenia bezdrôtového a Bluetooth modulu (Fitzer, 2018).

Posledným a najvýkonnejším modelom predstaveným v júli roku 2019 je Raspberry Pi 4 (Obrázok 5). Je to prvý model, ktorý sa predáva vo viacerých verziách s 1GB, 2GB alebo 4GB pamäťou RAM. Taktiež výkon tohto modelu je neporovnateľne vyšší ako u jeho predchodcov. Silnejší procesor, 2xHDMI port s možnosťou prehrávania 4K videa pri rýchlosti 60 snímok za sekundu, taktiež dva USB 3.0 porty + dva USB 2.0, Bluetooth 5.0 sú len niektoré z jeho predností.

Obrázok 5: Raspberry Pi 4 (Hattersley, 2019).

Pre bližšie špecifikácie modelov vid. (Tabuľka 1). Zoznam jednotlivých modelov firmy Raspberry Pi Trading v tejto kapitole nie je kompletný, ale pre účely tejto práce a ako forma stručnej histórie postačujúci. Pre túto prácu bola použitá druhá najnovšia generácia Raspberry Pi 3, model B+ , ktorý bol predurčený zadaním práce, z toho dôvodu je tomuto modelu a jeho špecifikáciám venovaná bližšia pozornosť v nasledujúcej kapitole.

Model	RPi 2 B	RPi 3 B	RPi 3 B+	RPi 4 B
SOC Type	Broadcom BCM2836	Broadcom BCM2837	Broadcom BCM2837B0	Broadcom BCM2711
CPU Clock	4 × Arm Cortex-A7, 900 MHz	4 × Arm Cortex-A53, 1.2 GHz	4 × Arm Cortex-A53, 1.4 GHz	4 × Arm Cortex-A72, 1.5 GHz
RAM	1 GB	1 GB	1 GB	1 GB/2 GB/4 GB
GPU	Broadcom VideoCore IV	Broadcom VideoCore IV	Broadcom VideoCore IV	Broadcom VideoCore VI
USB Ports	4	4	4	4 (2 × USB 3.0 + 2 × USB 2.0)
Ethernet	100 Mbit/s base Ethernet	100 Mbit/s base Ethernet	Gigabit Ethernet (max. 300 Mbps)	Gigabit Ethernet (no limit)
Power over Ethernet	No	No	Yes (requires separate PoE HAT)	Yes (requires separate PoE HAT)
WiFi	No	WiFi 802.11n	WiFi 802.11ac Dual Band	WiFi 802.11ac Dual Band
Bluetooth	No	4.1	4.2 BLE	5.0 BLE
Video Output	HDMI/3.5 mm Comp./DSI	HDMI/3.5 mm Comp./DSI	HDMI/3.5 mm Comp./DSI	micro-HDMI/3.5 mm Comp./DSI
Audio Output	I^2S/HDMI/3.5 mm Composite	I^2S/HDMI/3.5 mm Composite	I^2S/HDMI/3.5 mm Composite	I^2S/HDMI/3.5 mm Composite
Camera Input	15 Pin CSI	15 Pin CSI	15 Pin CSI	15 Pin CSI
GPIO Pins	40	40	40	40
Memory	MicroSD	MicroSD	MicroSD	MicroSD

Tabuľka 1: Špecifikácie vybraných modelov Raspberry Pi (Corral-Garcia et al., 2019).

1.2 Raspbian

Prevádzku Raspberry Pi zabezpečuje operačný systém. Rovnako ako pri bežných počítačoch, je to operačný systém, kto spravuje všetky zdroje počítača a poskytuje užívateľovi prostredie na prácu. Pre Raspberry Pi existuje viacero odporúčaných operačných systémov, z ktorých väčšina má Linux ako základňu. Avšak ani jeden z týchto operačných systémov nie je oficiálne vytvorený firmou, ktorá stojí za Raspberry Pi. Napriek tomu je od roku 2015 systém Raspbian oficiálne poskytovaný Raspberry Pi Foundation ako primárny a odporúčaný operačný systém.

Raspbian bol vytvorený ako nezávislý projekt a je neoficiálnou deriváciou Debianu Wheezy, prispôsobeného tak aby poskytoval optimalizovaný „hard float“ kód, (hardware floating point calculus- výpočty pri ktorých je použitá desatinná čiarka sú vykonávané hardwarovo), ktorý je schopný bežať na Raspberry Pi. V súčasnosti existujú mnohé verzie Raspbianu, z ktorých druhou najnovšou a taktiež použitou pre tento projekt je verzia Stretch. Posledná verzia bola vydaná vo februári 2020 a nesie názov Buster. Napriek tomu, že za vznikom Raspbianu primárne stoja Mike Thompson a Peter Green, veľkým prínosom je aj pomoc zo strany členov Raspberry Pi komunity, ktorým záleží na tom, aby mohli využívať svoje zariadenia na maximum (Raspbian, 2018).

Pre úplných začiatočníkov v programovaní je k dispozícii NOOBS (New Out Of the Box Software) software, ktorý zabezpečí jednoduchú inštaláciu požadovaného operačného systému. V prípade inej voľby ako Raspbian je daný software následne stiahnutý cez internet.

Vďaka mnohým firmám a nadšencom dnes existujú desiatky operačných systémov optimalizovaných pre Raspberry Pi. Dá sa povedať, že každý má nejaké výhody aj nevýhody a oblasti použitia, na ktoré sa hodí viac či menej. K dispozícií sú operačné systémy, ako pre začiatočníkov tak pre profesionálov a bližší zoznam so stručným popisom je dostupný z (Heath, 2018).

1.3 Alternatívy k Raspberry Pi

V oblasti jednodoskových počítačov je Raspberry Pi nepochybných šampiónom (minimálne čo sa týka počtu predaných kusov). Ale tak ako je to vo všetkých odvetviach aj v tomto prípade sa so vzrastajúcim záujmom o Raspberry Pi začali vynárať jeho alternatívy. Momentálne nie je Raspberry Pi ani najvýkonnejším ani najlacnejším jednodoskovým počítačom a aj keď sa vzhľadom na počet predných kusov zdá, že v pomere cena/výkon jednoducho nemá premožiteľa, je zrejmé, že mnohí užívatelia pri výbere zariadenia prihliadajú aj na ďalšie faktory (Locker, 2019).

Niektoré alternatívy sa hodia viac pre aplikácie kde sa vyžaduje väčší výpočtový výkon, iné zase vyžadujú minimálna spotrebu energie, malé rozmery a pod. Taktiež je dobré brať ohľad na kompatibilitu s operačnými systémami a netreba zabúdať ani na rôzne spôsoby konektivity (Moe, 2019). V nasledujúcej časti budú predstavené niektoré najznámejšie alternatívy spolu so stručným popisom prípadne porovnaním s referenčným Raspberry Pi 3, model B+.

Arduino Mega 2560

Obrázok 6: Arduino Mega 2560 (Locker, 2019).

- Procesor: ATmega2560
- GPU: /
- Pamäť: 256KB Flash ROM
- Cena: $ 33

Arduino je názov pre rodinu open-source jednodoskových mikrokontrolérov. Na rozdiel od Raspberry Pi založeného na mikroprocesore je Arduino založené na mikrokontroléroch, čo znamená že sa jedná o jednoduchý počítač, na ktorom dokáže iteratívne bežať jeden program. Aj napriek tomu, že Arduino výkonnostne nie je možne porovnávať s Raspberry Pi sa jedná o alternatívu čísla jedna. V oboch prípadoch je totiž jednoducho možné ovládať elektroniku pripojenú k ich GPIO (general-purpose input/output), Raspberry Pi je však navyše možné použiť ako plnohodnotný stolový počítač a celkovo je možné ho použiť na zložitejšie úlohy. Arduino na druhej strane je jednoduchšie aplikovať na vytváranie rôznych projektov z oblasti elektroniky, kde je výhodou práve možnosť rýchleho uvedenia do prevádzky pri jednoduchých úlohách, ktoré nevyžadujú výpočtový výkon. Ďalej je ho možné veľmi použiť napríklad pri vytváraní 3D tlačiarne, ovládaní pohybu robota prípadne ako náhradu za mikrokontrolér ako taký (Gutierrez, 2019).

Rock Pi 4 Model B

Obrázok 7: Rock Pi Model B (Locker, 2019).

- Procesor: Rockchip RK3328 (4x Cortex-A53 @ 1.5GHz)
- GPU: Mali T860MP4
- Pamäť: 64bit dual channel LPDDR4@3200Mb/s, 1GB/2GB/4GB voliteľné
- Cena: $ 39/70…

Rock Pi prišlo na trh ku koncu roka 2018. Jedná sa o kvalitný jednodoskový počítač so štvorjadrovým 1,5 GHz procesorom a dostatočne výkonnou Mali T860MP4 GPU. Pamäť RAM môže byť rozšírená až na 128 GB. Početné možnosti ukladania dát poskytujú vyššiu rýchlosť čítania a zápisu dát na externé zariadenia a tým celkovo vyššiu efektívnosť pri práci s dátami. Možnosť GPU akcelerácie je obzvlášť zaujímavá v prípade záujmu o algoritmy umelej inteligencie alebo strojové učenie. Rozmery, rozloženie a dizajn sú totožne s Raspberry Pi a pri veľmi podobnej cene sa rozhodne jedná o zaujímavú alternatívu (Locker, 2019).

Banana Pi M64

Obrázok 8: Banana Pi M64 (Long, 2019).

- Procesor: Allwinner A64 (4x Cortex-A53 @ 1.2GHz)
- GPU: Mali-400 MP2
- Pamäť: 2GB DDR3 RAM; 8GB to 64GB eMMC
- Cena: $ 61

Banana Pi M64 je vylepšenou verziou predchádzajúceho modelu Banana Pi. Beží na Androide, Ubuntu, Debian a ďalších operačných systémoch. Štvorjadrový procesor, Mali-400 MP2 GPU, 2 GB RAM a 8GB eMMC (embedded MultiMediaCard) modul rozšíriteľný pomocou microSD karty sú dobrou výbavou. S lepšími parametrami prichádza aj vyššia cena oproti Raspberry takmer o dvojnásobok. Wifi a Bluetooth sú tak ako u Raspberry Pi 3 samozrejmosťou avšak Banana Pi má iba dva USB porty oproti 4 u Raspberry Pi (Kimari, 2019).

Asus Tinker Board S

Obrázok 9: Asus Tinker Board S (Locker, 2019).

- Procesor: Rockchip RK3288 (4x Cortex-A17 @ 1.8GHz)
- GPU: Mali-T760
- Pamäť: 2GB LPDDR3 RAM; 16 GB eMMC
- Cena: $89

Asus vsadil na úspech Raspberry Pi a pre svoj jednodoskový počítač ako je už od pohľadu jasne použil rovnaké rozmery rozloženie komponentov a tým aj kompatibilitu so všetkým príslušenstvom pre Raspberry Pi 3 ako aj možnú náhradu týchto zariadení v predošlých projektoch. Tinker Board S má značne navrch, čo sa hardwaru týka, so svojim 4 jadrovým 1,8 GHz procesorom, výkonnou GPU, 2 GB RAM a Gigabitovým Ethernetom oproti Fast Ethernetu u Raspberry. Taktiež disponuje 16 GB eMMc pamäťou smart audio jackom ktorý poskytuje oveľa kvalitnejšie audio a ďalšími vylepšeniami oproti Raspberry Pi. Jedinou nevýhodou je snáď cena, ktorá je 2,5 násobne vyššia oproti Raspberry Pi. Aj napriek tomu sa však jedná o silnú konkurenciu (Kimari, 2019).

Odriod-XU4

Obrázok 10: Odroid-XU4 (Long, 2019).

- Procesor: Samsung Exynos5422 (4x Cortex-A15 @ 2.0GHz and 4x Cortex-A7 @ 1.4GHz)
- GPU: Mali-T628 MP6
- Pamäť: 2GB LPDDR3 RAM; voliteľná až do 64GB eMMC
- Cena: $85

Odroid – XU4 by sa dal nazvať ďalšou generáciou jednodoskových počítačov s výkonnejším a energeticky úspornejším hardwarom. Je jedným z najvýkonnejších jednodoskových počítačov aké je možné v súčasnej dobe kúpiť a už od pohľadu je zrejmé, že nekopíruje rozmery a rozloženie Raspberry Pi 3. Svojim výkonom a parametrami ho však jednoznačne prevyšuje. Okrem výkonného Samsung Exznos 5422 a Mali-T628 MP6 GPU disponuje aj 2 GB RAM a eMMC 5.0, USB 3.0 a Gigabitovým Ethernetom, ktoré umožňujú rýchly prenos dát. Ďalšou výhodou je podpora veľkého množstva operačných systémov ako Kali Linux, Arch Linux, Ubuntu, Android, RetroPie, Recalbox a mnohých ďalších. Napriek chýbajúcej Wifi a Bluetooth konektivite je pri cene 85$ výkonnejšou náhradou za Raspberry Pi 3 (Kimari, 2019).

Analýza a návrh

Nasledujúca kapitola sa venuje analýze a návrhu riešenia praktickej časti diplomovej práce. Na začiatku popisuje zvolený model Raspberry Pi 3B+ a jeho funkcionalitu, ďalej je zameraná na výber vhodných doplnkov k tomuto modulu a následne výberu vhodného softwaru. Poskytuje stručný popis jednotlivých hardwarových a softwarových komponentov, prípadné výhody, nevýhody a dôvody pre výber daného riešenia.

1.4 Návrh systému

Táto diplomová práca si kladie za cieľ vytvoriť tzv. „inteligentný priezor“. Jedná sa o zariadenie, ktoré bude možné umiestniť v blízkosti vstupu do miestnosti, domu, budovy a bude schopné zaznamenať a rozpoznať osobu, ktorá sa nachádza v jeho databáze a oznámiť úspešnú autorizáciu sprevádzanú svetelným a zvukovým signálom. Neznáme osoby, ktoré sa v databáze nenachádzajú sa budú môcť autorizovať pomocou QR kódu, obsahujúceho správne heslo, v inom prípade nebudú autorizované. Pri dlhšej dobe prítomnosti je osoba zaznamenaná prostredníctvom fotiek a videa. Treťou možnosťou je použitie tlačidla priamo na Raspberry ktoré odošle notifikáciu na Android zariadenie s nainštalovanou aplikáciou a následne bude môcť kompetentná osoba povoliť autorizáciu na diaľku prípadne využiť inú funkcionalitu aplikácie.

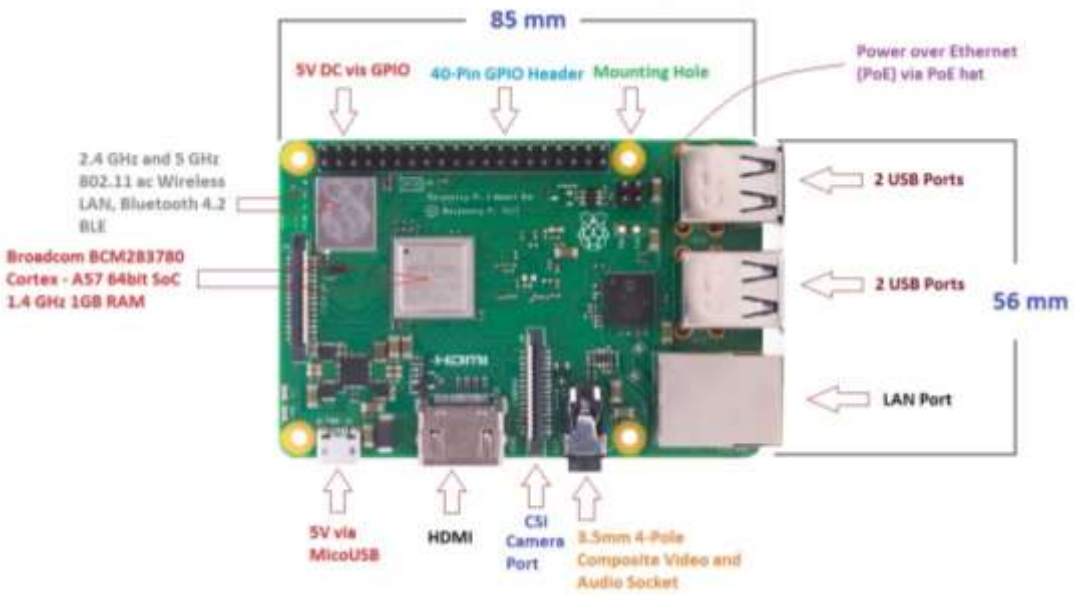
Z formy návrhu by sa dalo očakávať, že samotné zariadenie bude priamo spojené so zámkom dverí za účelom ich otvorenia. Pre nerealizovanie takéhoto riešenia existujú viaceré dôvody. Hlavným dôvodom je úroveň zabezpečenia, akú môže takýto systém s implementáciou rozpoznávanie tváre na Raspberry Pi poskytnúť. Aj napriek možnej vysokej úrovni presnosti a spoľahlivosti pri autentifikácií, je takýto systém náchylný na autentifikáciu osôb z fotografií alebo videa, ako bude podrobnejšie popísané neskôr. Z bezpečnostných dôvodov, sa preto nie je možné spoliehať, na takýto systém, ak sa jedná napríklad o prístup do miestnosti alebo priestorov. Návrh bol preto prispôsobený možnostiam, ktoré takýto systém vie poskytnúť a činnostiam, pri ktorých je možné sa na takéto riešenie spoľahnúť. V návrhu však boli ponechané niektoré prvky, ktoré sa môže zdať, nie úplne korelujúce s výsledným konceptom, avšak vhodne ho dopĺňajú, najmä v prípade budúceho rozširovania projektu. Ďalším dôvodom pre nerealizovanie prepojenia s fyzickými dverami môžeme poväzovať nemožnosť otestovať a realizovať takýto systém v prostredí kde bol projekt vyvíjaný. Celkovo sa bude sa jednať o systém s popísanou funkcionalitou, ktorý môže plne slúžiť ako forma inteligentného dochádzkového systému, monitorovacieho systému, prípadne formy bezpečnostného „smart“ kamerového systému so záznamom pohybu, tvárí, osôb atď. Výstupom bude svetelný a zvukový signál oznamujúci úspešnú autorizáciu prípadne zamietnutie.

Celý proces spracovania obrazu až po konečné rozpoznanie osoby bol prvotne implementovaný iba na Raspberry Pi. Za účelom lepšej kvality konečného riešenia, bolo toto rozšírené neskôr rozšírené aj na verziu, kde k spracovaniu obrazu prichádza na

vzdialenom výkonnejšom počítači a samotné Raspberry slúži na vykonávanie výpočtovo menej náročných úkonov. Bližšie informácie k postupu sú následne poskytnuté v ďalších kapitolách.

1.5 Hardware

Ako už bolo spomínané pre účely tejto diplomovej práce bol určený výpočtový modul Raspberry Pi 3B+ (Obrázok 11), ktorý je druhým najnovším a najvýkonnejším prírastkom do rodiny Raspberry (Gutierrez, 2019). Je osadený 64 bitovým 1.4 GHz quad-core ARM Cortex-A53 procesorom a RAM s približne 1GB pamäti, ktorá je zdieľaná s GPU. Tiež obsahuje už zabudovaný dual-band 802.11ac Wifi and Bluetooth 4.2 BLE (Bluetooth Low Energy). Oproti predošlým modelom poskytuje lepší odvod tepla a trojnásobný výkon. Ethernet port podporuje rýchlosť 300Mbit/s (100Mbit/s predošlé verzie) a taktiež funkciu PoE (Power over Ethernet) ktorá umožňuje zredukovať množstvo použitých káblov (Locker, 2018).



Obrázok 11: Popis modelu Raspberry Pi 3 B+(Aqueel, 2018).

Model 3B+ ako aj jeho predchodcovia disponuje 40 kolikovým konektorom (Obrázok 12). Z týchto 40 kolíkov je 26 použiteľných ako digitálny I/O (Input/Output, Vstup/Výstup) a zvyšných 14 kolíkov je určených pre napájanie, uzemnenie alebo iné účely. Na (Obrázok 11) je ďalej možné vidieť ešte 4 oddelené kolíky ktoré v prípade potreby slúžia na pripojenie prídavného modulu ktorý následne umožňuje využiť funkciu PoE (Aqueel, 2018).

Obrázok 12: „Pin-out" pre model 3B+(Aqueel, 2018).

1.5.1 Kamera

Medzi oficiálne príslušenstvo poskytovane k Raspberry Pi patria aj kamerové moduly. V súčasnosti je možné naraziť na štyri poskytované varianty, dve staršie s maximálnym rozlíšením 5MPx a dva novšie s rozlíšením 8MPx, vždy v štandardnej a NoIR verzii (Raspberrypi, 2016). Verzia NoIR neobsahuje infračervený filter a je ju možné použiť aj pre snímanie obrazu v tme, za predpokladu použitia dodatočného infračerveného osvetlenia. Značnou nevýhodou je však farebne skreslený obraz pri snímaní za normálnych svetelných podmienok (Gus, 2019). Pre účely tejto práce bol zvolený model Pi V2 NoIR (Obrázok 13) pre prípadnú možnosť použitia v noci. Dátová komunikácia a napájanie modelu je realizované pomocou ohybného plochého CSI (Camera Serial Interface) káblu o 15 vodičoch. Pre osadenie modulu bol použitý upravený kryt pre Raspberry.

Parametre podľa (Raspberrypi, 2016):

- IR filter: ÁNO
- Čip: 8Mpx Sony IMX219 CCD
- Foto: až 3280x2464 pixelov
- Video: 1080p @ 30fps; 720p @ 60 fps; 640x480p @ 90 fps
- Konektor: CSI
- Rozmery: 25 × 20 × 9 mm
- Spätná kompatibilita : ÁNO

Obrázok 13: Kamerový modul Pi V2 NoIR (Raspberrypi, 2016).

1.5.2 Reproduktor

Pre možnosť použitia zvukového výstupu či už pre oslovenia alebo usmernenie používateľa je dobré k Raspberry Pi pripojiť reproduktor. K tomuto účelu bol zvolený malý prenosný reproduktor (Obrázok 14), napájaný pomocou dvojice AAA 3V batérii. Na každej strane sa nachádza 1W reproduktor, celková hmotnosť je 40g pri rozmeroch 80x80x30 mm a pripojenie je realizované štandardným 3.5 mm audio konektorom (Kebidu, 2019). V prípade potreby väčšej hlasitosti je možné pripojiť akékoľvek iné kompatibilné zariadenie.

Obrázok 14: Prenosný reproduktor (Kebidu, 2019).

1.5.3 Traffic HAT

Ďalším doplnkom použitým pre túto diplomovú prácu je produkt firmy Raynteck tzv. Traffic HAT (Obrázok 15). Jedná sa o tri veľké diódy vo farbách svetelnej signalizácie (červená, žltá, zelená) spolu s jedným tlačidlom a zvončekom (Nuttall, 2016). Skratka HAT v názve znamená „Hardware Attached on Top“ čiže hardware ktorý sa umiestni priamo na Raspberry Pi. Traffic HAT bol zvolený najmä kvôli možnosti jednoduchej interpretácie výstupu programu pomocou farieb ktoré sú všeobecne dobre známe a taktiež možnosť vstupu od užívala pomocou tlačidla prípadne výstupu prostredníctvom jednoduchého zvukového znamenia zvončeka. Sada Traffic HAT je dodávaná ako polovične naletovaná kde konektor na pripojenie tlačidla, HAT EEPROM (Electrically Erasable Programmable Read-Only Memory) obvod a rezistor pre tlačidlo sú pred-letované. Diódy, zvonček a tlačidlo je však nutné priletovať (Rush, 2018).

Obrázok 15: Traffic HAT (Nuttall, 2016).

1.6 Software

Pri návrhu a výbere softwarovej výbavy je nutné brať ohľad na kompatibilitu dostupných programovacích jazykov a knižníc. Ako operačný systém bol zvolený už spomínaný Raspbian Stretch. Za najvhodnejší programovací jazyk bol zvolený jazyk Python v dôvodu dobrej kompatibility s potrebnými knižnicami. Na dodatočné spracovanie obrazu a ďalšie funkcie spojené s počítačovým videním boli použité primárne knižnice OpenCV, Dlib a neskôr spomenuté dodatočné knižnice. V ďalšej kapitole je taktiež popísaný software slúžiaci na sprístupnenie videa online, prenos videa na výkonnejší počítač a taktiež posielanie notifikácií do Android aplikácie.

1.6.1 Python

Obrázok 16: Logo Python (Python, 2019).

Python je interaktívny, interpretovaný programovací jazyk, vytvorený ako open-source projekt Guidom van Rossumom. Jeho najaktuálnejšou verziou je v súčasnosti verzia 3.8.2. Podporuje objektovo orientované programovanie, štruktúrované aj funkcionálne programovanie. V poslednej dobe sa veľmi často používa práve v oblasti „Data Science“ (Veda o Dátach), do ktorej spadá napríklad strojové učenie, analýza a vizualizácia dát atď. Jeho veľkou výhodou je práve vysoký výkon, ktorý dosahuje pri použití v rôznych aplikáciách a to práve vďaka tomu že knižnice pre náročné numerické operácie ktoré Python využíva, nie sú písané v jazyku Python ale sú písané v jazykoch ako C++ a Fortran. Ďalšou výhodou je jednoduchá možnosť rozširovania Python pomocou modulov písaných napríklad v C alebo C++ a taktiež možnosť použitia ako rozširovacieho jazyka pre už existujúce moduly a aplikácie. Pre túto diplomovú prácu bol vybraný ako vhodná alternatíva s dobrou kompatibilitou (Python, 2019).

1.6.2 OpenCV

Obrázok 17: Logo OpenCV (OpenCV, 2019).

OpenCV (Open Source Computer Vision Library) je softvérová knižnica s otvoreným zdrojovým kódom (open-source), pre strojové učenie a počítačové spracovanie obrazu. Je vydaná pod tzv. BSD licenciou ktorá je jedna z najpoužívanejších pre open-source softvér a znamená že použitie OpenCV je zdarma ako pre akademické tak komerčné účely. Má rozhranie pre C++, Python a Javu a podporu pre Windows, Linux, Mac OS, iOS a Android. OpenCV je C/C++ knižnica navrhnutá s dôrazom na aplikácie v reálnom čase. Obsahuje viac ako 2500 optimalizovaných algoritmov zameraných na spracovanie obrazu a strojové

učenie. Pre túto prácu boli využité práve niektoré algoritmy zamerané na najmä na detekciu tvárí, pohybu a pod (OpenCV, 2019).

1.6.3 Dlib

Obrázok 18: Logo Dlib (King, 2019a).

Knižnica Dlib je sada moderných nástrojov C++, ktorá obsahuje algoritmy strojového učenia a ďalšie nástroje na vytváranie komplexného softvéru. Jedná sa o open-source licenciu, takže ako v predošlom prípade, je jej použitie bezplatné čo je veľkou výhodou a aj vďaka tomu si aj táto knižnica získala veľkú obľubu medzi používateľmi. Knižnica ako taká poskytuje mnohé užitočné funkcie ako rôzne učiace algoritmy, numerické algoritmy, funkcie spracovania obrazu, kompresné algoritmy atď. (King, 2019a).

Pre túto diplomovú prácu bola knižnica Dlib zvolená hlavne z dôvodu kvalitných algoritmov na spracovanie obrazu. Jej použitie by sa mohlo zdať zbytočné, keďže OpenCV disponuje veľkým množstvom obdobných algoritmov, avšak niektoré algoritmy z tejto knižnice sú implementovateľné iným, lepším, alebo jednoduchším spôsobom.

1.7 Výber ďalšieho softvéru

Pri OpenCV a Dlib bol výber zjednodušený tým, že sa jedná o velmi kvalitné a dostupné riešenia, za ktoré sa iba ťažko hľadá náhrada s obdobnou funkcionalitou a s možnosťou použitia na Raspberry Pi. Nasledujúca časť iba stručne popisuje výber aj v prípadoch, kde bolo podobných alternatív viac. V takom prípade je nutné otestovať viaceré riešenia za účelom objavenia tej najvhodnejšej. Takýto proces je často časovo náročný, pretože dokumentácia nie je vždy dostatočne prehľadná a je potrebný značný čas na zorientovanie sa vo funkcionalitách a samotné otestovanie riešenia. Avšak prináša aj možnosť zvoliť si také riešenie, ktoré najlepšie zodpovedá projektovým požiadavkám.

1.7.1 Flask

Obrázok 19: Logo Flask (Flask, 2019).

Nakoľko je pre potreby mobilnej aplikácie a vzdialenej správy je nevyhnutné sprístupniť obraz z kamery na Raspberry Pi v reálnom čase online, bolo potrebné vybrať vhodný nástroj, ktorý túto funkcionalitu sprostredkuje. Na výber je viacero možností, z ktorých sú niektoré lepšie, iné horšie, čo sa týka najmä rýchlosti prenosu, jednoduchosti implementácie atď. Medzi najpoužívanejšie „Pythnovské“ webové rámce (anglicky framework) jednoznačne patria Django, Flask, Tornado, Web2py, Bottle a iné (Petlovana, 2020). Django je jeden z „najrozvinutejších“ (full-stack framework), s veľkým množstvom funkcionalít, z ktorých drvivá väčšina nie je potrebná pre túto diplomovú prácu. Ako vhodnejšia voľba sa javili tzv. micro-framerowky, medzi ktoré partí Flask, Bottle, Cherry Pi. Jednoznačne najobľúbenejším z nich je práve Flask, ktorý sa osvedčil ako vhodná voľba aj v tomto prípade. Oproti „full-stack frameworkom“ je extrémne „ľahký“ a tým pádom značne jednoduchší na prvotné zavedenie, zároveň však jednoducho škálovateľný v prípade potreby rozširovania a pridávania funkcionalít (Flask, 2019). Samotné sprístupnenie videa online sa preto jednoducho a rýchlo dá implementovať pomocou Flasku ako pri Raspberry tak aj na výkonnejšom počítači.

1.7.2 ZeroMQ

Obrázok 20: Logo ZeroMQ (Hintjens, 2020).

Ďalej bolo potrebné zvoliť software, pomocou ktorého je možné preposielať obraz z Raspberry Pi kamery na výkonnejší počítač. Ako v predošlom prípade aj tu existuje množstvo kvalitných knižníc a protokolov slúžiacich na posielanie obrazu medzi počítačmi. Medzi alternatívy zvažované pre túto prácu patria napríklad: FFMPEG, RabbitMQ, AMQP, ROS, GStreamer a iné. Po dôslednom zvážení, pre aj proti, viacerých knižníc, bola pre účely tejto diplomovej práce zvolená knižnica ZeroMQ (známa tiež ako ZMQ,0MQ). Jedná sa o open-source univerzálnu asynchrónnu vysoko výkonnú knižnicu s podporou väčšiny operačných systémov. Ako sa uvádza na oficiálnych stránkach, „Zero“ („nula“ po anglicky) v názve znamená nulového sprostredkovateľa (zero broker), nulovú latenciu, nulovú cenu a nulovú administráciu (Hintjens, 2020). K odosielaniu obrazu zachyteného kamerou na Raspberry je v tejto práci využité „imageZMQ“. Jedná sa o skupinu tried v jazyku Python ktoré umožňujú posielať obraz z OpenCV na druhý počítač pomocou PyZMQ, knižnice ktorá obsahuje väzby medzi jazykom Python a už spomínaným ZMQ (Pyzmq, 2020) (Bass, 2019).Veľkou výhodou ZMQ oproti ostatným zvažovaným knižniciam na posielanie správ, je už spomínaná „ľahkosť“ a výkonnosť. ZMQ nevyžaduje sprostredkovateľa správ čo je veľké plus najmä pri posielaní väčších súborov ako napríklad fotky. Celý proces je vďaka tomu oveľa svižnejší a práve vysoký prietok a malá latencia, čo sú parametre dôležité pre túto diplomovú prácu, boli dôvodom pre výber tejto knižnice.

1.7.3 FireBase

Obrázok 21: Logo Firebase (Firebase, 2020).

Pre prácu na mobilnej aplikácií bolo využité Android Studio. Viac detailov k aplikácií a jej funkciám sa nachádza v ďalších kapitolách. Jedna funkcionalita však stojí za samostatnú zmienku. Jedná sa o FireBase, mobilnú platforma od Google, ktorá umožňuje relatívne rýchlo a jednoducho vytvárať kvalitné mobilné aplikácie s množstvom funkcionalít. Ide o tzv. Backend as a Service (BAAS). Podporuje iOS, Android, C++ a iné. V tejto diplomovej práci bola FireBase implementovaná do Android aplikácie za účelom posielanie notifikácií používateľovi tejto aplikácie v prípade, že niekto mechanicky stlačí tlačidlo na Raspberry Pi (čiže inteligentnom priezore) a v tom momente je potrebné informovať povolanú osobu/y. Nakoľko Firebase nepoužíva hypertextový prenosový protokol ale tzv. WebSocket, je komunikácia takmer bez oneskorenia, čo je veľkou výhodou aj pri použití v tejto diplomovej práci (Firebase, 2020).

Teoretický základ

Jedným z hlavných pilierov, okolo ktorých je vystavaná táto diplomová práca je rozpoznávanie tváre. Preto je tejto téme v nasledujúcej kapitole venovaná značná pozornosť s dôrazom najmä na metódy neskôr implementované v tejto práci. Táto kapitola zámerne nasleduje až po predstavení návrhu systému.

Cieľom nasledujúcej kapitoly nie je vyčerpávajúci rozbor jednotlivých stupňov rozpoznávania tváre, ale výstižný popis, ktorý dostatočne opisuje jednotlivé kroky a trendy v týchto oblastiach. Podkapitoly poskytujú stručné odôvodnenie použitých a nepoužitých metód v tejto práci. Tieto odporúčania sa vzťahujú k použitiu na Raspberry Pi 3B+ prípadne obdobnom zariadení.

1.8 Proces rozpoznávanie tváre

So slovným spojením rozpoznávanie tváre sa dnes môžeme stretnúť na každom kroku ako aj so systémami ktoré toto rozpoznávanie implementujú. Často sa však môžeme stretnúť s neúplným pochopením príbuzných výrazov, ktorými sú identifikácia a verifikácia (nazývaná aj ako autentifikácia) človeka pomocou tváre. Rozpoznávanie tváre je všeobecný pojem, ktorý zahŕňa aj identifikáciu aj verifikáciu, avšak jedná sa o odlišné procesy ku ktorým treba aj odlišne pristupovať (Huang, 2019).

Pri verifikácií sa jedná o systém, ktorý sa snaží zodpovedať otázku: „Je táto osoba tou osobou za ktorú sa vydáva?“ V takom prípade je následne tvár človeka, ktorý stojí pred kamerou porovnáva s tvárou človeka s odpovedajúcim označením v databáze. Systém rozhodne o zhode prípadne nezhode. Tieto systémy sa všeobecne označujú ako 1:1 systémy. Verifikácia je preto proces rýchlejší a presnejší ako identifikácia a zväčšovanie databázy nemá priamy vplyv na rýchlosť vyhodnotenia.

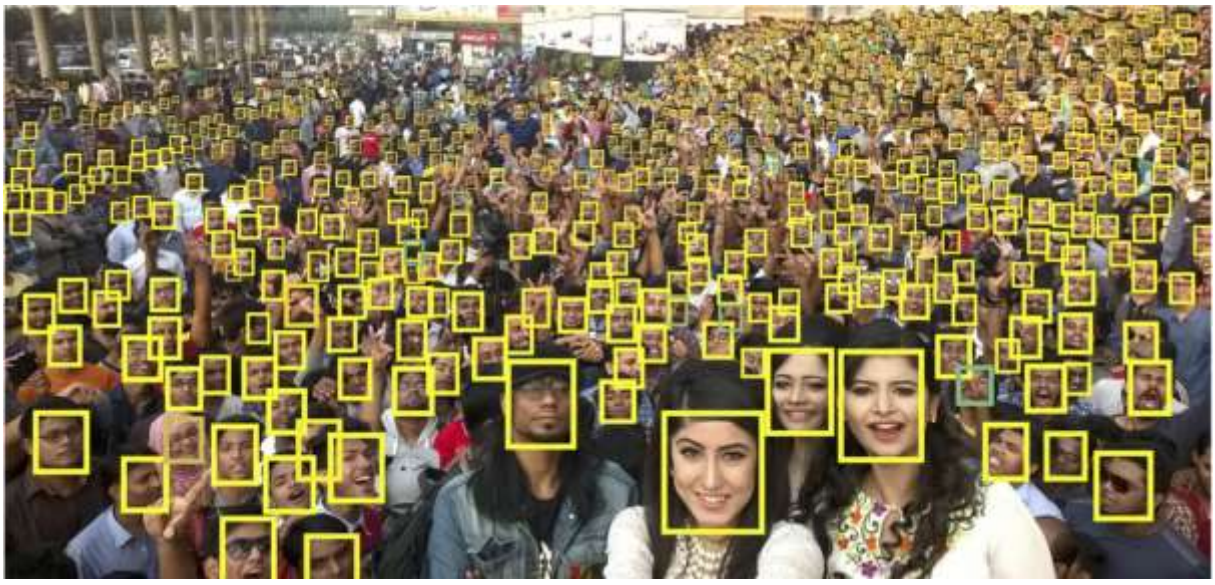
Identifikácia je odlišná od verifikácie tým, že sa snaží zodpovedať otázku: „Kto je táto osoba?“ K tomu je nutné, aby systém porovnal tvár osoby pred kamerou so všetkými údajmi v databáze. Preto sú tieto systémy popisované ako 1:n, kde „n“ je počet záznamov, s ktorými je nutné vyhotovenú fotku porovnať. Z tohto faktu je zrejmé, že identifikačné systémy pracujú pomalšie ako systémy verifikačne a ich rýchlosť sa značne odvíja aj od veľkosti databázy (Marcel, 2006). V tejto diplomovej práci sa využíva práve identifikácia osôb.

V skutočnosti ale ani v jednom z vyššie spomínaných systémov nedochádza k porovnaniu skutočných tvárí. Celý proces rozpoznávania je oveľa zložitejší a predchádza mu niekoľko krokov. Najčastejšie sa rozčleňuje na: **Detekciu**, **Zrovnávanie**, **Reprezentáciu**, **Klasifikáciu** (Taigman *et al.*, 2014).

Všetky tieto kroky sú predmetom častých výskumov a vylepšení s jedným a tým istým zámerom, a to vytvoriť systém, ktorý čo najspoľahlivejšie rozpozná ľudskú tvár a úspešnosť správnej klasifikácie sa bude blížiť ľudskej úspešnosti (prípadne bude lepšia), za rôznych svetelných podmienok, prostredia, veku, etnicity a pod. (Taigman *et al.*, 2014).

1.9 Detekcia

K rozpoznávaniu tvárí môžeme pristúpiť až po ich úspešnej lokalizácií v obraze (Obrázok 22). Dá sa povedať, že najväčší boom detekcii tváre bol zaznamenaný od roku 2001, po tom čo Paul Viola a Michael Jones vynašli algoritmus, ktorý bol schopný detegovať objekty v reálnom čase a bol implementovateľný aj na bežne doma používané zariadenia. Jedná sa o tzv. Haar Cascades klasifikátor. Tento algoritmus je možné použiť na detekciu rôznych typov objektov aj keď primárnou motiváciu pri vytváraní bola pravé detekcia tváre (Viola and Jones, 2001). V súčasnosti už existujú mnohé spoľahlivejšie metódy detekcie, ako napríklad metóda vynájdená v roku 2005 nazvaná Histogram of Oriented Gradients (HOG) a taktiež dnes často skloňované a jedno z najkvalitnejších riešení súčasnosti, ktorým sú Convolutional Neural Networks (CNN) (Dalal and Triggs, 2005) (Kumar *et al.*, 2019).



Obrázok 22: Ukážka výstupu systému na detekciu tváre (Khazri, 2019).

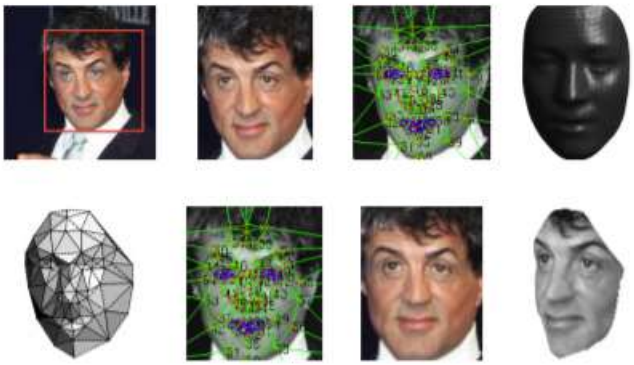
Pri výbere správnej metódy je nutné brať ohľad na zariadenie, na ktorom bude detekcia a ďalšie fázy rozpoznávania tváre prebiehať. V prípade Raspberry Pi nie je možné použiť CNN nakoľko nemá dostatočný výpočtový výkon. Metóda HOG bola v tejto práci použitá na detekciu tvárí v databáze osôb, ktoré majú mať povolený vstup. Tento proces je predprípravou preto nie je nutné klásť také veľké nároky na rýchlosť spracovania a môžeme uprednostniť vyššiu presnosť detekcie, ktorú poskytuje. Keď však kladieme dôraz na dostatočnú rýchlosť pri detekcii, jedinou dostatočne spoľahlivou a rýchlou metódou,

poskytujúcou uspokojivé výsledky na Raspberry Pi v reálnom čase je práve Haar Cascades od Viola-Jones.

1.10 Natočenie tváre

Po správnom detegovaní tváre v obraze, je pre dosiahnutie čo najväčšej presnosti a spoľahlivosti, nutné sa popasovať aj s ďalším bežným problémom. Je ním natočenie tváre. Takáto tvár pre počítač vyzerá úplne inak ako tvár, ktorá v obraze smeruje priamo na kameru. Výkon jednotlivých rozpoznávacích algoritmov vo všeobecnosti závisí na presnom orezaní časti tváre z obrazu a správnom nasmerovaní tváre do šablóny tak, že pozíciu jednotlivých oporných bodov na tvári je vzhľadom na pevný súradnicový systém možné podrobiť štandardnej analýze. Mnohé „zrovnané“ databázy, ktoré sa používajú na trénovanie rozpoznávacích algoritmov ako napríklad Labeled Faces in a Wild-a (LFW-a) poskytujú normalizovaný vstup a tak zlepšujú učenie týchto algoritmov (Huang, 2019).

Vyriešiť natočenie tváre v obraze sa však stále považuje za nie úplne ľahkú úlohu. V súčasnosti existujú rôzne metódy, ktoré sa úspešne využívajú jednu alebo viaceré spôsoby zrovnávania, za účelom dosiahnutia čo najlepších výsledkov (Taigman *et al.*, 2014). Najbežnejšími a najpoužívanejšími spôsobmi sú: vytvorenie 3D analytického modelu tváre; hľadanie podobného rozostavenia referenčných bodov v externej databáze z ktorých sa následne vychádza; metódy ktoré samé hľadajú transformáciu na základe podobnosti, pre pixely na obrázku; (Dong *et al.*, 2018) (Hassner and Ieee, 2013; Yi *et al.*, 2013) (Huang *et al.*, 2007).



Obrázok 23: Obrázok zobrazuje postup zrovnávania tváre (zľava doprava, z hora, dole) podľa (Taigman et al., 2014). Detekcia so 6 počiatočnými referenčnými bodmi; 2D orezanie; 67 referenčných bodov na 2D výreze so zodpovedajúcimi Delaunayho trianguláciami; 3D referenčný tvar pretransformovaný do 2D zrovnaného roviny; (druhý riadok) viditeľnosť jednotlivých častí na 3D-2D modely; referenčné body vypočítane na základe 3D modelu využité na zrovnanie a napriamenie modelu tváre; finálny zrovnaný výrez; dodatočne vytvorený pohľad. Takto vytvorený model dokáže následne vytvoriť aj nový pohľad na tvár, ktorý môže byť užitočný v ďalšom rozpoznávaní (viď posledný obrázok).

Zakomponovanie zrovnávania tvárí do algoritmu nie je potrebné vždy a záleží od koncového použitie a implementácie. V našom prípade treba brať do úvahy obmedzený zdroje, ktoré Raspberry Pi poskytuje a aj charakter problému. V rámci prípravy tejto diplomovej práce boli otestované oba algoritmy ako s, tak aj bez zrovnávania tváre. Algoritmus s implementovaných jednoduchým zrovnávaním tvárí spôsoboval značné spomalenie spracovávania obrazu a neposkytoval žiadne dodatočné výhody a to najmä z dôvodu, že sa jedná o systém vedomej identifikácie a efekt zrovnávania je preto minimálny, keďže osoby pristupujú k systému väčšinou vzpriamene a vhodne. Z toho dôvodu nie je zrovnávanie tváre zakomponované v algoritme použitom pre túto diplomovú prácu. Navyše po viacerých pokusoch ako s modelmi na rozpoznávanie tváre z knižnice OpenCV tak aj s Dlib (King, 2017), bola pre túto prácu (ako bude upresnené neskôr) vybraná knižnica Dlib. Práve Dlib sa po viacerých testoch na menších statických databázach javila ako menej závislá na tom či je na spracovávaných fotkách použité zrovnávanie tváre alebo nie.

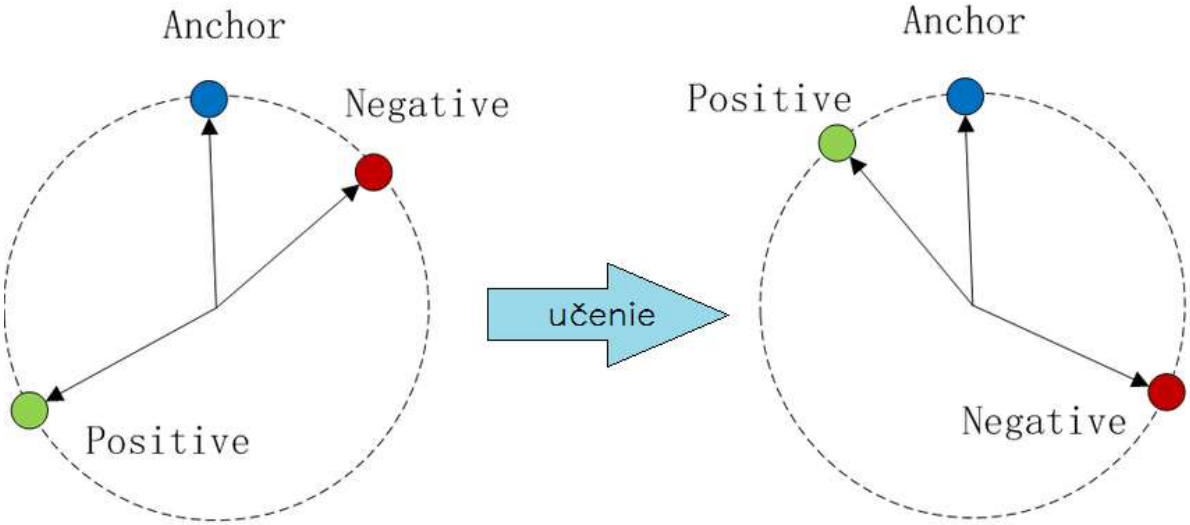
1.11 Reprezentácia

Správna reprezentácia a extrakcia čŕt tváre je dôležitým krokom v procese rozpoznávania. Z výpočtového hľadiska je totiž nutné fotky zredukovať takým spôsobom, ktorý by umožňoval ich čo najrýchlejšie, ale aj najspoľahlivejšie porovnanie. Pri 2D rozpoznávaní tváre sa môžeme stretnúť s dvoma najčastejšími reprezentáciami: geometrická reprezentácia a reprezentácia založená na vzhľade (appearance-based representation (ABR)), prípadne ich kombinácia. V súčasnosti, sa však ako najpraktickejšie a najúspešnejšie javia práve metódy založené na vzhľade. Výskumy potvrdzujú, že rovnaký popis obrazu môže byť využitý na rozpoznávanie objektov ako aj tvárí preto niet divu, že práve ABR sú dominujúcou reprezentáciou (Ding *et al.*, 2008) (Meyers and Wolf, 2008). Tieto metódy sú založené na zaznamenávaní rôznych štatistických údajov o jednotlivých pixeloch v rámci tváre (vytvárajú tzv. šablóny, anglicky „templates“), na rozdiel od geometrických metód ktoré využívajú napríklad meranie vzájomnej vzdialenosti oči, šírky obočia, atď. (Li, 2009).

Vzhľadom na svoju dôležitosť, je práve oblasť reprezentácie tváre predmetom skúmania mnohých odborníkov. Snahy boli dlho upriamené na vytváranie čo najlepšieho spôsobu vyjadrenia tváre. Čo však zostalo pri vytváraní stále nových spôsobov popisu v úzadí je fakt ,že nech už sa jedná o akýkoľvek spôsob popisu (skonštruovanej reprezentácie) použitého pri rozpoznávaní tváre, zväčša tento popis používa rovnaké operátory na všetky oblasti tváre (Taigman *et al.*, 2014) (Yavuz *et al.*, 2016). Avšak práve v posledných rokoch ruka v ruke so zväčšujúcim sa množstvom dostupných dát sa metódy využívajúce hlboké učenie prekonávajú metódy skonštruovaných reprezentácií, pretože sú schopné, na základe učenia nájsť a optimalizovať črty najdôležitejšie pre rozpoznanie tváre podľa vlastných potrieb (Krizhevsky *et al.*, 2017). Inými slovami „necháme to na počítač“ aby si vybral, ktoré údaje z tváre sú pre neho dôležité pri rozpoznávaní, a ktoré nie.

Riešenie použité v tejto diplomovej práci vychádza z (Schroff *et al.*, 2015). Podstata je v natrénovaní Hlbokej Konvolučnej Neurónovej Siete (Deep Convolutional Neural Network) na na-mapovanie obrazu tváre do kompaktného Euklidovského priestoru, kde vzdialenosti priamo zodpovedajú miere podobnosti jednotlivých tvárí (v angličtine sa pre takúto kvantifikáciu často používa výraz embedding). Výhodou oproti mnohým iným prístupom vyžívajúcim hlboké učenie je, že sieť je natrénovaná, aby priamo optimalizovala takúto kvantifikáciu tváre. Výhodou tohto prístupu je velmi vysoká efektivita reprezentácie. Jednotlivé „embeddings" (kvantifikátory, zakódovania) tvárí v tejto diplomovej práci teda podľa vzoru (Schroff *et al.*, 2015) zaberajú iba 128-bytov.

Proces učenia spočíva v načítaní troch fotiek. Prvá a druhá fotka („Anchor" a „Positive") sú jednej a tej istej osoby, tretia fotka odlišnej osoby („Negative"). Následne sa algoritmus snaží upraviť váhy neurónovej siete tak aby kvantifikácia (v našom prípade 128D vektor) prvej a druhej fotky, čiže tej istej osoby, boli čo najpodobnejšie, a kvantifikácia druhej a tretej fotky, čiže odlišných osôb, čo najodlišnejšie. Hovoríme o tzv. Triplet loss funkcii (Obrázok 24). Tento proces sa následne opakuje milióny-krát s množstvom fotiek, až kým nepríde k spoľahlivému nastaveniu váh (Schroff *et al.*, 2015).



Obrázok 24: Triplet Loss funkcia minimalizuje vzdialenosť medzi „Anchor" a „Positive" (rovnaká osoba) a maximalizuje vzdialenosť medzi „Anchor" a „Negative" (odlišné osoby) (Zhu, 2017).

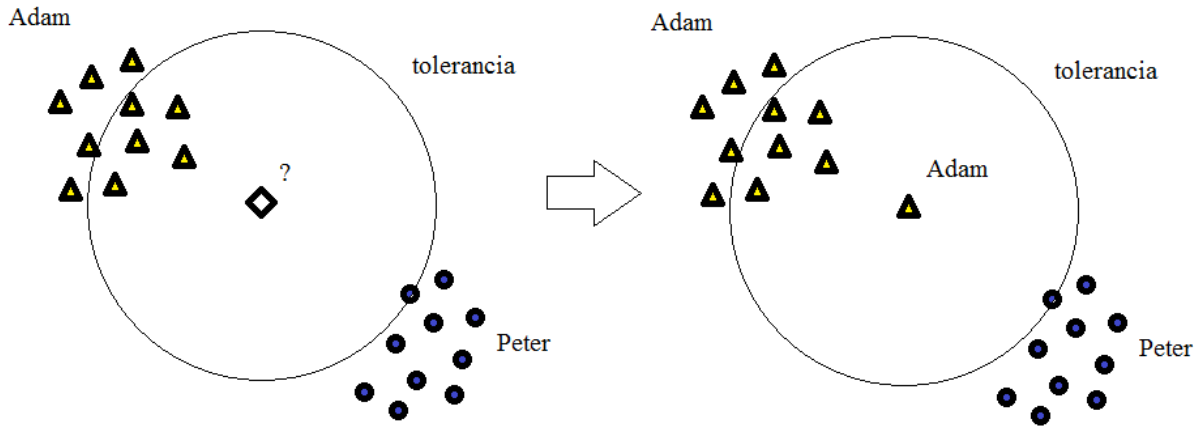
Trénovanie takejto Hlbokej CNN na bežných zariadeniach môže trvať rádovo stovky až tisíce hodín (Schroff *et al.*, 2015). Je zrejmé, že tréning takejto neurónovej siete nie je možné vykonávať na Raspberry Pi. Neurónová sieť využitá pre túto prácu, bola natrénovaná vyššie uvedeným spôsobom, na približne 3 miliónoch fotkách tvárí. Je súčasťou knižnice Dlib, ktorej autorom je (King, 2017). Jedná sa o obdobu ResNet-34 (He *et al.*, 2016), je však tvorená 29 konvolučnými vrstvami a oproti pôvodnej ResNet-34 má taktiež počet filtrov na jednej Konvolučnej vrstve zredukovaný na polovicu. Dá sa povedať,

že pre potreby tejto diplomovej práce je kvalita reprezentácie, ktorú poskytuje spomínaná Hlboká CNN viac ako postačujúca.

1.12 Klasifikácia

Posledným krokom v procese rozpoznávania tváre je klasifikácia. Ako už bolo spomínané, podľa charakteru problému, môže ísť najčastejšie o identifikáciu, autentifikáciu alebo zhlukovanie. Po tom čo máme k dispozícií „embeddings“ jednotlivých tvárí, musíme vybrať a natrénovať dnes už jeden z „typických“ modelov strojového učenia, ktorý bude schopný jednotlivé osoby rozlíšiť. Môže sa jednať o Support-Vector Machine (SVM), Random Forest, k-Nearest Neighbor (k-NN). Výber správneho klasifikátora môže mať značný vplyv na presnosť výsledkov celkového rozpoznávania a v snahe zvýšiť presnosť sa môžeme pokúsiť natrénovať rôzne, „lepšie“ klasifikátory. Výhodou k-NN, ktorý bol v upravenej forme použitý pre účely tejto diplomovej práce, je práve jeho jednoduchosť. V prípade, že máme pár vzorových fotiek, je totiž ťažké natrénovať dobrý klasifikátor. Naopak pri veľkom počte fotiek v databáze sa môže zdať tento klasifikátor ako menej výhodný z dôvodu porovnávanie aktuálnej hodnoty so všetkými údajmi v databáze. Ak máme v databáze viac fotiek toho istého človeka, porovnanie prebieha so všetkými fotkami (Xu and Iop, 2018) (Singh *et al.*, 2016).

Ďalšou výhodou tohto modelu je napríklad aj to, že pridanie novej osoby do databázy nevyžaduje žiadnu dodatočnú úpravu existujúcich reprezentácií, kedy napríklad pri SVM je nutné znovu natrénovať celý model, čo môže byť pri veľkom počte fotiek v databáze časovo náročné a nepraktické (Singh *et al.*, 2016) (Luo, 2016).



Obrázok 25: Ukážka klasifikácie pri dvoch osobách v databáze. V tolerancií sa nachádza viac fotiek osoby „Adam“ z toho dôvodu je vstup vyhodnotený ako osoba „Adam“.

V našom prípade, sa však upravený k-NN kvalifikátor javí ako najlepšia voľba, práve vzhľadom na svoju jednoduchosť, rýchlosť a dostatočnú presnosť. V k-NN použitom pre túto prácu nie je zadávaný počet najbližších susedov (nearest neighbor) ale tzv. tolerancia

(vzdialenosť) medzi fotkami v databáze a vstupnou fotkou. Jedná sa o euklidovská vzdialenosť. Po prekročení tolerancie medzi porovnávanými fotkami sa započíta jeden hlas pre danú osobu, jedná sa o zhodu. Osoba („trieda“), ktorá má po prejdení celej databázy najviac hlasov, je vyhodnotená ako tá daná osoba. V prípade, že tolerancia nie je prekročená pri žiadnej fotke z databázy, je rozpoznávaná osoba označená ako neznáma. Pri výskyte málo pravdepodobnej zhody hlasov sa vyberie podľa abecedného poradia.

1.13 Ladenie parametrov

Tak, ako je nutné prispôsobiť jednotlivé metódy a algoritmy výpočtovým možnostiam Raspberry Pi, aby sme dosiahli čo najuspokojivejší a funkčný výsledný produkt, je nutné prispôsobiť aj jednotlivé parametre týchto algoritmov s cieľom dosiahnuť čo najvyššiu presnosť rozpoznávania osôb.

Jedná sa o ladenie parametrov jednotlivých metód, po anglicky tzv. „hyperparameter tuning“. V mnohých prípadoch sa aj dnes používa prístup pokus-omyl, ktorý môže často a nemusí poskytovať uspokojivé výsledky. Cieľom je však objaviť kombináciu parametrov, pre ktoré sú výsledky čo najlepšie. V prípade zložitých algoritmov je takýto proces výzvou, nakoľko si môže vyžadovať desiatky hodín a množstvo výpočtového výkonu. Napríklad CNN majú vo všeobecnosti veľký počet parametrov ktoré sú nastavované ešte pred procesom trénovania (Luo, 2016). V prípade, že je potrebná vyššia presnosť je nutné zvoliť iné hodnoty parametrov a proces učenia opakovať znovu. Neexistujú metódy ktoré by ustanovovali ako efektívne nastaviť tieto parametre (Wang *et al.*, 2018). Pre algoritmy strojového učenia však existujú viaceré prístupy, ktoré slúžia na nájdenie najlepšej kombinácie týchto parametrov. Patria sem: Mriežkové hľadanie, Náhodne hľadanie, Optimalizácia pomocou gradientu, Genetické algoritmy, Bayesovská optimalizácia (Pedregosa *et al.*, 2011) (Hao and Ho, 2019) (Guo *et al.*, 2019) (Thiede and Parlitz, 2019) (Cornejo-Bueno *et al.*, 2018). Metódy využívajúce Bayesovskú optimalizáciu sú vo všeobecnosti považované za jedny z najefektívnejších (Wang *et al.*, 2018).

V prípade tejto diplomovej práce, s ohľadom na skutočnosti a metódy spomínané v predchádzajúcich podkapitolách, takéto extenzívne hľadanie nebolo potrebné nakoľko nedochádzalo priamo k ladeniu týchto algoritmov, pre parametre vyžadujúce pozornosť bolo postačujúce využitie metódy ručného ladenia.

Konfigurácia

Táto kapitola sa venuje príprave na realizáciu projektu, jednotlivým krokov a prípadne problémom, ktoré pri tvorbe nastali. Stručne popisuje prípravu prostredia, inštaláciu a konfiguráciu. Jedná sa o kroky, ktoré je nutné vykonať predtým, než sa pristúpi k samotnej implementácií zdrojového kódu. Nejde však iba obsiahly popis postupu, snahou je naopak oboznámiť čitateľa aj s odôvodnením niektorých krokov, pri riešení projektu a poskytnúť tak hlbší náhľad na zvolené technológie a riešenia. Nasledujúci postup a rady sa vzťahujú k operačným systémom: Windows 10, Raspbian Stretch a Ubuntu 18.04.

1.14 Inštalácia Raspbianu

Operačný systém Raspbian je možné nainštalovať viacerými spôsobmi, v každom prípade je však potrebný počítač s prístupom na internet a čítačka microSD kariet.

Obrázok 26: Ukážka NOOBS na oficiálnej stránke www.raspberrypi.org.

Jednou z možností je použite NOOBS (New Out Of the Box Software) (Obrázok 26), ktorý po spustení Raspberry Pi zabezpečí inštaláciu operačného systému. Pri zvolení tejto možnosti stiahneme NOOBS z oficiálnej stránky *www.raspberrypi.org*. Na výber sú dve verzie: NOOBS, ktorá podporuje možnosť inštalácie off-line a možnosť NOOBS Lite, pri ktorej je pre inštaláciu nutné pripojenie k internetu. Stiahnutý *.zip* súbor je potrebné rozbaliť a kompletne prekopírovať na naformátovanú microSD kartu. Odporúčaná minimálna kapacita je 8GB (Raspbian, 2018), pre potreby tejto diplomovej práce bolo však nutné použiť vyššiu kapacitu. Karta je vložená do Raspberry Pi a následne sa postupuje podľa pokynov na obrazovke, až do úspešného nainštalovania operačného systému. Vyššie popísaný postup bol zvolený aj pre túto diplomovú prácu.

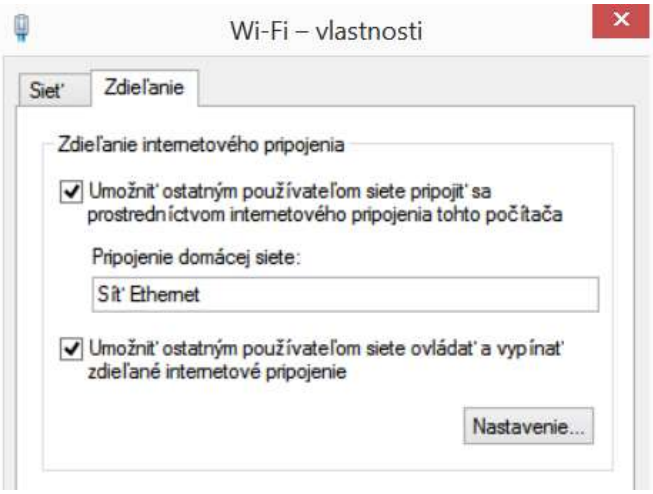
Druhý postup, o niečo zložitejší, vyžaduje stiahnutie Raspbian *image* súboru, taktiež z *www.raspberrypi.org*. Nakoľko je súbor skomprimovaný (formát ZIP64), je nutné ho rozbaliť špeciálnym softwarom pre Windows, napríklad 7-zip, ktorý je možné zdarma získať z *www.7-zip.org*. Pre zápis *image* súboru na SD kartu je opäť potrebný dodatočný software, napríklad Win32 Disk Imager, dostupný z *sourforce.net*. Po zápise *image* súboru na microSD kartu, stačí vložiť kartu do Raspberry Pi, ktoré sa spustí s funkčným operačným systémom.

1.15 Práca s Raspberry

Na prácu s Raspberry Pi je nevyhnutné použitie dodatočného hardwaru. Či už sa jedná o pripojenie k počítaču napríklad pomocou Eternetového káblu, vzdialené pripojenie, alebo jednoduché pripojenia displeja klávesnice a myši priamo k Raspberry Pi. Pri tvorbe tejto diplomovej práce boli použité všetky vyššie spomenuté postupy.

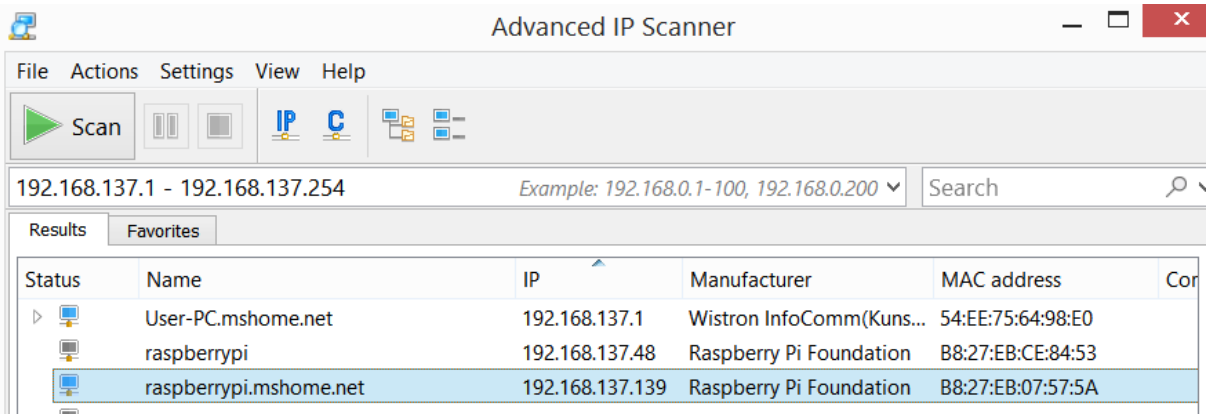
Pre vzdialený prístup k príkazovému riadku Raspberry Pi v rámci rovnakej siete bola využitá technológia SSH (Secure Shell). Jedná sa o počítačový program a zároveň aj súvisiaci sieťový protokol, ktorý umožňuje bezpečnú šifrovanú komunikáciu medzi dvoma neautentifikovanými počítačmi v nezabezpečenej sieti (Maheshwari, 2017). Veľmi rozšírenou SSH implementáciou, dnes nielen na platforme Microsoft Windows, využitou pre tuto prácu, je PuTTY (klient protokolov SSH, Telnet, rlogin, TCP a terminál pre spojenia cez sériový port) (Tatham, 2019).

Prvým krokom je pripojenie Raspberry k počítaču pomocou štandardného eternetového kábla a spustenie samotného Raspberry Pi. Následne bolo nutné povoliť zdieľanie pripojenia aby sa Raspberry mohlo pripojiť k internetu prostredníctvom použitého počítača (Obrázok 27).



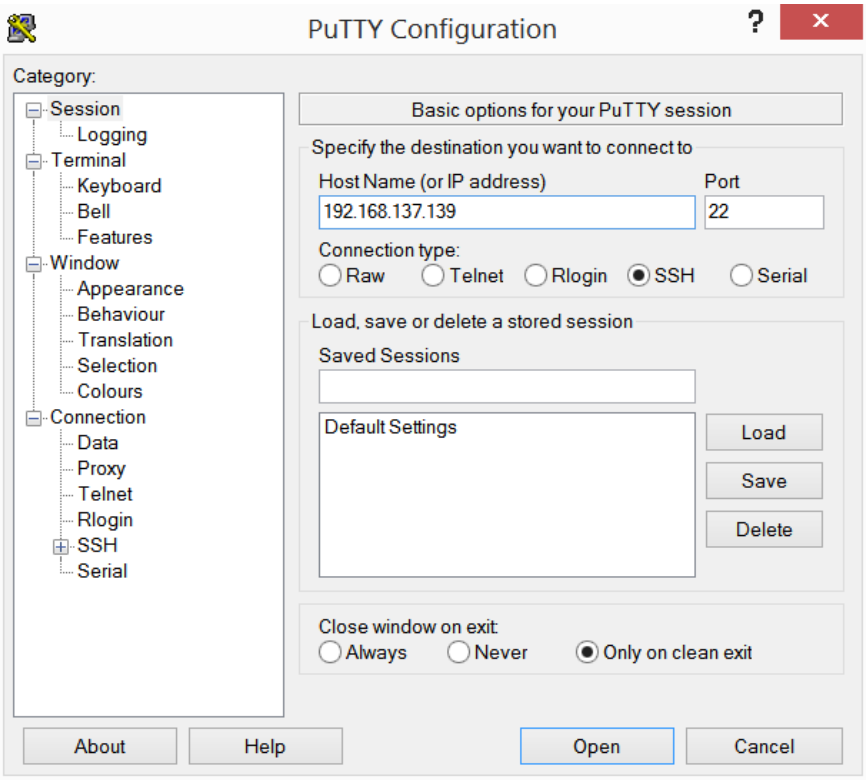
Obrázok 27: Zdieľania internetového pripojenia.

Na pripojenie sa k Raspberry Pi je nutné zistiť jeho IP adresu. Na jej zistenie bol použitý bezplatný program Advanced IP Scanner (Obrázok 28), dostupný z *www.advanced-ip-scanner.com*.



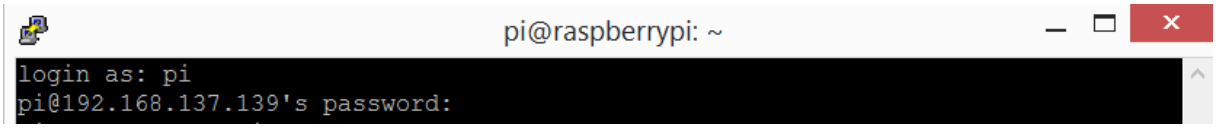
Obrázok 28: Zvýraznený riadok s modrou ikonou odkazuje na IP adresu Raspberry Pi.

Zistená IP adresa bola zadaná do prostredia *PuTTY* (Obrázok 29). *SSH* server štandardne počúva na porte 22 z toho dôvodu je táto hodnota pred-vyplnená.



Obrázok 29: Užívateľské rozhranie PuTTY.

Po kliknutí na tlatíčko „open“ dôjde k nadviazaniu spojenia a pre prístup je nutné sa prihlásiť užívateľským menom (štandardne: „pi“) a heslom (štandardne: „raspberry“) (Obrázok 30). Následne je možné ovládať Raspberry Pi pomocou príkazového riadku.



Obrázok 30: Prihlásenie.

Nevýhodou tohto spojenia je však to, že umožňuje prístup iba k príkazovému riadku, čo nebolo vždy úplne postačujúce. Navyše, od novembra roku 2016 je *SSH* od výroby deaktivovaná (Magpi, 2017). Pre aktiváciu bolo nutné pripojiť monitor a klávesnicu a následne manuálne povoliť túto možnosť. Taktiež by bolo možné povoliť túto funkcionalitu pomocou umiestnenia súboru s názvom *ssh* bez prípony do *boot* oddielu SD karty pomocou iného počítača.

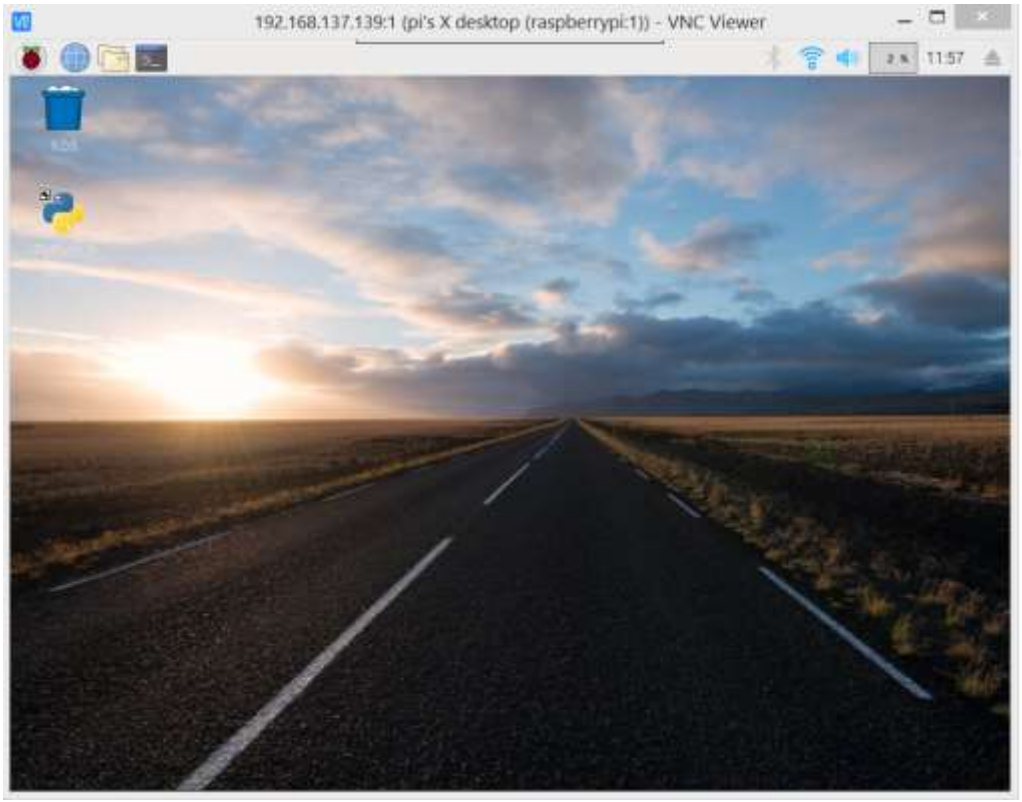
Pre prístup k plnej pracovnej ploche Raspberry bol využitý systém *VNC* (*Virtual Network Computing*), dostupný z *www.realvnc.com*. Jedná sa o systém zdieľania pracovnej plochy, ktorý umožňuje vzdialene ovládať pracovné prostredie jedného počítača (na ktorom beží *VNC* server), z iného počítača (na ktorom beží *VNC Viewer*) (Rouse, 2006). K nainštalovaniu a spusteniu *VNC* serveru na Raspberry bola použitá vyššie popísaná technológia *SSH* a nasledujúce príkazy. Je dobrým zvykom uistiť sa, že pred inštaláciou nového softwaru je naše Raspberry Pi aktuálne.

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install tightvncserver
```

Následne spustíme VNC server príkazom

```
$ tightvncserver
```

Zobrazí sa pridelená adresa *VNC* serveru napríklad: 192.168.137.139:1. Túto adresu zadáme do programu *VNC Viewer* a následne po prihlásení, sa otvorí okno zobrazujúce pracovnú plochu Raspberry Pi (Obrázok 31), ktoré umožňuje ovládať Raspberry Pi, ako keby sme pracovali priamo na ňom. Je však nutné poznamenať, že nie všetky funkcie a príkazy, najmä čo sa týka práce s obrazom zaznamenávaným Raspberry Pi kamerou, fungujú správne, a je teda dobré si pri prípadných problémoch overiť funkčnosť pripojením sa k Raspberry Pi iným spôsobom.



Obrázok 31: Pracovná plocha Raspberry Pi v prostredí VNC Viewer.

1.16 Virtuálne prostredia

Virtualnv je nástroj na vytváranie izolovaných prostredí v programovacom jazyku Python. Základným problémom, ktorý použite virtuálnych prostredí rieši je problém závislostí a verzií potrebných pre rôzne aplikácie. Ak jedna aplikácia potrebuje verziu 2.0 a iná verziu 1.0 istého programu je možné práve vďaka rôznym virtuálnym prostrediam používať tieto aplikácie na jednom zariadení. Taktiež je týmto spôsobom možné používať rôzne verzie programu Python. Rovnako v prípade funkčnej aplikácie nemusíme riskovať, že pri náhodnej zmene nejakej knižnice, počas práce na inom projekte príde k jej znefunkčneniu. Vo všetkých týchto a mnohých ďalších prípadoch môže Virtualnv značne uľahčiť prácu. Vytvára prostredie, ktoré má svoje vlastné inštalačné adresáre, ktoré nezdieľa knižnice s inými virtuálnymi prostrediami a voliteľne nepristupuje ani ku globálne nainštalovaným knižniciam (Docs, 2019). Aj keď použitie virtuálnych prostredí nie je nevyhnutné, osobne ho vrelo odporúčam najmä pri rozsiahlejšej práci. Jedná sa o velmi užitočný nástroj, ktorý môže ušetriť často nemalé problémy a čas. Virtuálne prostredia boli použité aj pri tejto diplomovej práci, pre inštaláciu je potrebné použiť príkaz:

```
$ sudo pip install virtualenv virtualenvwrapper
```

Zložku *.profile* prípadne *.bash_profile* je nutné upraviť nasledovne, či už z terminálu alebo v editore.

```
$ echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.profile
```

```
$ echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.profile
$ echo "export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3" >> ~/.profile
$ echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.profile
```

Následne použijeme príkaz `source` ktorý načíta a umožní používať príkazy z tejto zložky.

```
$ source ~/.profile
```

Ďalším krokom je vytvorenie virtuálneho prostredia. Pre projekt tejto diplomovej práce bolo vytvorené napríklad prostredie s názvom *opcv4*.

```
$ mkvirtualenv opcv4 -p python3
```

Pre prácu v tomto prostredí použijeme príkaz.

```
$ workon opcv4
```

1.17 Inštalácia OpenCV

OpenCV bolo stručne popísané v prechádzajúcej kapitole. Nasledujúca časť popisuje ako nainštalovať OpenCV na Raspberry Pi a integrovať túto knižnicu do jazyka Python. Ak nie je spomenuté inak, inštalácia tejto aj všetkých ďalších spomínaných knižníc, a taktiež následná práca prebieha v rámci virtuálneho prostredia (aj v prípade inštalácie na výkonnejšom počítači).

V súčasnosti je možné OpenCV jednoducho nainštalovať pomocou príkazu `pip install z` príkazovej riadky. PIP je správca balíkov (modulov) pre Python. Pomocou tohto príkazu by bolo možné nainštalovať „pre-build“ (pred-vytvorené) OpenCV balíky v priebehu niekoľkých minút. Nejedná sa o oficiálne balíčky od OpenCV, to však nie je dôvod na ich nevyužitie. Je však nutné poznamenať, že tento spôsob vo väčšine prípadov (pri 3 z 4 vykonaných pokusov v rámci tejto diplomovej práce) skončí internou chybou kompilátora. Operačný systém jednoducho ukončí kompiláciu nakoľko sa začne používať príliš veľa pamäte, preto je nutné, aj v tomto prípade, podniknúť niektoré z nižšie uvedených krokov. Ďalšou nevýhodou tohto spôsobu je, že neposkytuje žiadnu kontrolu nad kompiláciou, pretože balíčky už sú pred-kompilované, čo neumožňuje zahrnúť žiadne prídavne optimalizácie ani dodatočné algoritmy. Z toho dôvodu, bola pre túto diplomovú prácu zvolená metóda kompilácie zo zdroja, ktorá na rozdiel od predchádzajúcej možnosti zaberie rádovo hodiny, avšak umožňuje inštaláciu potrebných optimalizácií.

Pred začatím práce na inštalácii je dobré sa uistiť že je Raspberry aktuálne.

```
$ sudo apt-get update && sudo apt-get upgrade
```

Následne je nutné nainštalovať CMake. Jedná sa o open source, multiplatformovú rodinu nástrojov, ktoré poskytujú kontrolu nad inštaláciou softvéru (Cmake, 2019).

```
$ sudo apt-get install build-essential cmake unzip pkg-config
```

Pre prácu s obrazom a videom je nutné nainštalovať potrebné knižnice. Taktiež ďalšie užitočné knižnice ktoré sa môžu zísť pri práci s OpenCV a ďalej knižnice ktoré obsahujú numerické optimalizácie. Ako posledné nainštalujeme vývojárske hlavičky pre Python.

```
$ sudo apt-get install libjpeg-dev libpng-dev libtiff-dev
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
$ sudo apt-get install libxvidcore-dev libx264-dev
$ sudo apt-get install libgtk-3-dev
$ sudo apt-get install libcanberra-gtk*
$ sudo apt-get install libatlas-base-dev gfortran
$ sudo apt-get install python3-dev
```

Následne stiahneme *opencv* a *opencv_contrib* súbor a rozbalíme ich.

```
$ wget -O opencv.zip https://github.com/opencv/opencv/archive/4.0.0.zip
$ wget-O opencv_contrib.zip \
https://github.com/opencv/opencv_contrib/archive/4.0.0.zip
$ unzip opencv.zip
$ unzip opencv_contrib.zip
```

Vstúpime do vytvoreného alebo vytvoríme nové virtuálne prostredie. Bližšia rozprava k virtuálnych prostrediam v jazyku Python je v predchádzajúcej kapitole.

```
$ source ~/.profile
$ workon cv
$ pip install numpy
```

Vytvoríme zložku *build.*

```
$ mkdir build
$ cd build
```

Pomocou príkazu cmake nakonfigurujeme *build* nasledovne (týmto spôsobom môžeme nainštalovať dodatočné algoritmy a optimalizácie).

```
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-4.0.0/modules \
    -D ENABLE_NEON=ON \
    -D ENABLE_VFPV3=ON \
    -D BUILD_TESTS=OFF \
    -D OPENCV_ENABLE_NONFREE=ON \
    -D INSTALL_PYTHON_EXAMPLES=OFF \
    -D BUILD_EXAMPLES=OFF ..
```

Pred kompiláciou je vhodné (a často nevyhnutné) zväčšiť swap-size pamäťovej karty, čo minimalizuje pravdepodobnosť zamrznutia kompilácie z dôvodov nedostatku pamäte. Tento krok je vhodné vykonať aj pri inštalácií už prekompilovaných balíčkov. Z dôvodu životnosti pamäťovej karty, je však po inštalácii potrebné vrátiť swap-size na pôvodnú hodnotu, aby nedošlo k predčasnému zlyhaniu pamäťovej karty (Pfaffly, 2015).

```
$ sudo sed -i 's/CONF_SWAPSIZE=100/CONF_SWAPSIZE=1024/g' /etc/dphys-swapfile
$ sudo /etc/init.d/dphys-swapfile stop
$ sudo /etc/init.d/dphys-swapfile start
```

Potom spustíme kompiláciu (tento krok trvá často viac ako hodinu). Pre kompiláciu využijeme všetky štyri jadrá Raspberry (`-j4`).

```
$ make -j4
```

Po úspešnej kompilácii môžeme nainštalovať OpenCV

```
$ sudo make install
```

Posledným a dôležitým krokom je vytvorenie prepojenia medzi OpenCV a našim virtuálnym prostredným.

```
$ cd ~/.virtualenvs/opcv4/lib/python3.5/site-packages/
$ ln -s /usr/local/python/cv2/python-3.5/cv2.cpython-35m-arm-linux-
gnueabihf.so cv2.so
```

1.18 Inštalácia Dlib

Podobne ako v prípade inštalácie OpenCV si aj knižnica Dlib vzhľadom na obmedzené zdroje Raspberry Pi, vyžaduje predprípravu (King, 2019b). Je potrebné zvýšiť swap-size (na 1024 prípadne 2048) podobne ako v predošlej inštalácii. Obdobným spôsobom je nutné po ukončení vrátiť swap-size na pôvodnú hodnotu.

```
$ sudo sed -i 's/CONF_SWAPSIZE=100/CONF_SWAPSIZE=1024/g' /etc/dphys-swapfile
$ sudo /etc/init.d/dphys-swapfile stop
$ sudo /etc/init.d/dphys-swapfile start
```

Následne je vhodné vykonať ďalšie kroky s cieľom získať čo najviac dostupnej pamäte potrebnej pre kompiláciu. Pomocou nasledujúceho príkazu sa dostaneme do hlavných nastavení, kde v sekcii *Boot Options* zvolíme možnosť *Console Autologin*.

```
$ sudo raspi-config
```

Týmto spôsobom sa vyhneme načítavaniu užívateľského prostredia, a po reštarte Raspberry sa otvorí priamo terminál. To ušetrí značné množstvo pamäte RAM.

Ďalší krok k ušetreniu pamäte je zmenšenie veľkosti pamäte používanej GPU. Zostávajúc v hlavných nastaveniach sa v sekcii *Advanced Options* (rozšírené nastavenia) presunieme na *Memory Split* (rozdelenie pamäte) kde so štandardných 64MB zmeníme hodnotu na 16MB.

Následne reštartujeme Raspberry, aby vykonané kroky nabrali účinnosť.
Pre inštaláciu Dlib je nutné mať nainštalované nasledujúce balíky.

```
$ sudo apt-get update
$ sudo apt-get install build-essential cmake
$ sudo apt-get install libgtk-3-dev
$ sudo apt-get install libboost-all-dev
```

Do otvoreného virtuálneho prostredie následne nainštalujeme knižnicu Dlib.

```
$ pip install dlib
```

V prípade, že je grafická karta v počítači CUDA kompatibilná (Compute Unified Device Architecture) (čo nie je prípad Raspberry ani následne využívaného výkonnejšieho počítača), je vhodné nainštalovať Dlib priamo zo zdroja, čím značne urýchlime a zefektívnime následné používanie, pretože umožníme pri rozpoznávaní tvárí využívať aj výkon grafickej jednotky (Cuda, 2018).

1.19 Inštalácia dodatočných knižníc

Predchádzajúce prípady inštalácie sa môžu na prvý pohľad zdať komplikované pre neskúseného používateľa. Množstvo balíkov, programov a knižníc sa dá na Raspberry nainštalovať oveľa jednoduchšie. Väčšina ostatných knižníc pre jazyk Python bola nainštalovaná pomocou jednoriadkového príkazu `pip install <názov knižnice>` . Napríklad spomínaný Flask je možné nainštalovať nasledovne:

```
$ pip install flask
```

Jednoducho sa dajú nainštalovať aj balíky, ktoré sú súčasťou balíkovej knižnice pre Debian. Napríklad *festival* ktorý slúži na prevod textu na reč (tzv. TTS= Text To Speech) je možné nainštalovať pomocou príkazu:

```
$ sudo apt-get install festival -y
```

1.20 „Offloading“

Ako už bolo načrtnuté v predchádzajúcej kapitole, pre zlepšenie celkového výsledku, zvýšenie rýchlosti a kvality rozpoznávania, bola ako druhá alternatíva spracovávania obrazu použitá verzia, kde je využitý výkonnejší počítač. Raspberry Pi disponuje

obmedzenými výpočtovým výkonom, a preto použitie na výpočtovo náročnejšie úlohy môže byť občas výzvou. Jednou z možností ako Raspberry použiť a zbytočne ho nepreťažovať, bolo práve využitie výpočtového výkonu silnejšieho počítača.

Za účelom prenosu obrazu na druhý počítač bola použitá knižnica ImageZMQ. Je ju jednoducho možné nainštalovať pomocou príkazu:

```
$ pip install imagezmq
```

Na spracovanie bol použitý počítač *HP Pavilion dv6,* so 4-jadrovým procesorom Intel Core i5 a pamäťou RAM 6 GB. Operačný systém počítača je Windows 10. Na tomto počítači tak, ako na Raspberry Pi, bolo nutné nainštalovať potrebné knižnice. Inštalácia prekompilovaných balíčkov OpenCV je jednoduchá a je ju možné zvládnuť v niekoľkým krokoch. Snahou však bolo nainštalovať OpenCV priamo zo zdroja. Je nutné poznamenať, že inštalácia OpenCV na Windows 10 sa stretla s viacerými problémami a chybovými hláseniami. Po viacnásobných extenzívnych pokusoch nainštalovať OpenCV na Windows 10 sa knižnica nakoniec dala do prostredia Python importovať.

Druhou možnosťou, ktorá sa v konečnom dôsledku ukázala ako výhodnejšia, bola inštalácia Ubuntu 18.04 ako druhého operačného systému popri systéme Windows (Ubuntu je kompletná distribúcia operačného systému Linux pre pracovné stanice založená na linuxovej distribúcií Debian) (Ubuntu, 2020). Následne je možné používať počítač ako „dual-booth systém“ (systém s dvoma operačnými systémami na jednom pevnom disku). Popis inštalácie Ubuntu popri systéme Windows by bol mimo rozsah tejto diplomovej práce. Následná inštalácia OpenCV a ďalších knižníc sa z väčšej časti zhoduje s popisovanou inštaláciou na systéme Raspbian (okrem krokov špecifických pre Raspberry ako zmena veľkosti swapu a pod.), z tohto dôvodu nebude opakovane popisovaná.

1.21 Statická IP adresa

IP adresa (Internet Protocol adress) je unikátny číselný identifikátor zariadenia (uzla) v sieti komunikujúceho s ostatnými zariadeniami (uzlami) prostredníctvom IP protokolu. V súčasnosti je počet zariadení väčší ako počet dostupných Ipv4 adries (preto je nevyhnutný prechod na Ipv6) a z tohto dôvodu, nie je možné každému zariadeniu prideliť jedinečnú IP adresu. Riešením je využitie smerovača ktorý skupine zariadení pridelí tzv. neverejnú IP adresu zatiaľ čo on využíva adresu verejnú (pridelenú sprostredkovateľom internetového pripojenia). Následne smerovač prekladá tieto adresy na verejné. V jednom čase nemôžu v sieti internet existovať dve zariadenia s rovnakou verejnou IP adresou. Ako neverejné adresy sa používajú adresy zo stanovených intervalov (10.0.0.0– 10.255.255.255, 172.16.0.0– 172.31.255.255, 192.168.0.0 – 192.168.255.255).

Nastavením statickej IP adresy v rámci tejto diplomovej práce sa rozumie nastavenie statickej neverejnej adresy, v rámci domácej siete, aby bolo možné sa k Raspberry pripojiť bez problémov aj v prípade, že by z dôvodu dynamického

prideľovania IP adries po istom čase prišlo k jej zmene. Iba veľmi málo používateľov internetu disponuje statickou verejnou IP adresou. Jej získaniu predchádza väčšinou diskusia s poskytovateľom internetové pripojenia a nutnosť priplatiť si, nakoľko drvivá väčšina poskytovateľov internetového pripojenia statické adresy spoplatňuje. Statická verejná IP adresa by umožnila pristupovať k obrazu z Raspberry Pi, z akejkoľvek a nie len domácej siete.

Alternatívne by sa na tento účel dala využiť služba Dynamic DNS, ktorá udržuje aktuálne hodnoty dynamických adries. V oboch prípadoch by následne po správnom nastavení presmerovania portov na našom domácom routery, bolo možné pristupovať k videu z Raspberry z akejkoľvek siete. Dôvodov pre nevyužitie týchto možností je viacero. Okrem spoplatnenia služby a nutnosti obdržania statickej verejnej IP adresy, je ním aj potreba disponovať znalosťami konfigurácie siete. Splnenie takéhoto predpokladu je u koncového užívateľa málo pravdepodobné. V neposlednom rade je to aj nutnosť adekvátneho zabezpečenia takéhoto prístupu z vonkajšej siete, nakoľko sa jedná o citlivé dáta (živý prenos).

Na nastavenie statickej IP adresy v domácej sieti bola využitá priamo jednotka TP-link Deco M9 Plus. Prostredníctvom mobilnej aplikácie alebo webového rozhrania je možné rezervovať adresu pre zariadenie (Obrázok 32). Rezervovaná bola IP adresa ako pre Raspberry Pi tak pre HP Pavillion dv-6.

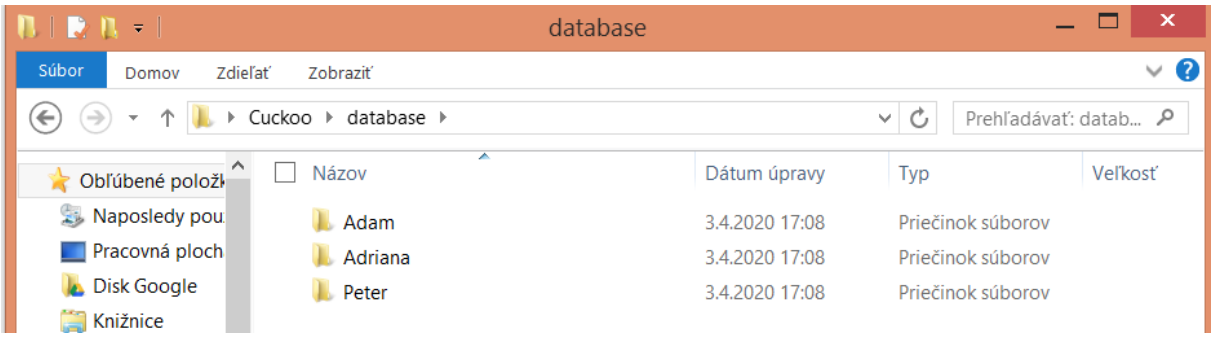
Obrázok 32: Ukážka rezervovanej adresy pre Raspberry Pi (CuckooRPi).

Implementácia

Po úspešnom nainštalovaní a príprave jednotlivých prvkov projektu je možné pristúpiť k fáze implementácie samotného riešenia. Táto kapitola popisuje jednotlivé algoritmy a použitie už spomínaných hardwarových alebo softwarových prvkov.

1.22 Vytvorenie databázy a reprezentácií

Za prvý krok môžeme považovať vytvorenie databázy s fotkami ľudí, ktorým má byť povolený vstup pomocou rozpoznania ich tváre. Fotky sú vložené do zložky „database“ (Obrázok 33), v ktorej má každá osoba vlastnú zložku, nazvanú jej menom. Skript následne práve podľa názvu zložky priradí fotke označenie.



Obrázok 33: Osobné zložky jednotlivých osôb. Cuckoo je názov hlavnej zložky projektu (z anglického cuckoo= kukučka). Názov bol použitý aj ako meno mobilnej aplikácie kvôli názornej podobnosti na slovo „kukátko“.

Na zakódovanie jednotlivých tvárí je použitý skript pod názvom *faces_database.py*. Ako argumenty mu predáme cestu k databáze a cestu kam uložiť súbor *faces_encoded.pickle* so zakódovanými tvárami. Nasledujúca časť kódu ukazuje cyklus pomocou ktorého sú jednotlivé fotky zakódované pre budúce použitie. Cyklus *for* prechádza všetky fotky a pri každej extrahuje meno z názvu zložky. Knižnica *face_recognition* slúži na jednoduchšie používanie funkcií z knižnice Dlib (Geitgey, 2017). Tváre sú na fotkách lokalizované pomocou funkcie *face_locations()*. Predvolený model tejto funkcie na lokalizáciu je HOG (Histogram of Oriented Gradients), keďže je to model ktorý chceme využiť, nie je nutné nič špecifikovať. Pre každú tvár sa následne vytvorí reprezentácia (zakódovanie), ktoré sa nakoniec uloží aj s menom do samostatných zoznamov.

V prípade, že je na obrázku lokalizovaných viac tvárí, skript predpokladá, že sa jedná a jednu a tú istú osobu. Je to najmä z dôvodu možnej druhej falošnej lokalizácie, ktorá by bez tejto funkcie mohla spôsobiť vynechanie skutočnej tváre zo zakódovania, prípadne možnú chybu programu, čo by mohlo spôsobiť nutnosť opakovať celý proces aj napriek tomu, že samotné následné rozpoznávanie, týmto nie je nijak negatívne ovplyvnené. Počas vypracovávania tejto diplomovej práce taká situácia nenastala, ale v prípade kódovanie tisícov fotiek je nutné počítať aj s takýmto variantom.

```
# looping all image paths
```

```
for (i, PathsIMG) in enumerate(PathsIMG):
       # get the name from path
       print("Encoding",i + 1)
       name = imagePath.split(os.path.sep)[-2]

       # preprocess so that dlib can be used
       img = cv2.imread(PathIMG)
       rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

       # coordinates od localized faces
       localization = face_recognition.face_locations(rgb)
       # creation of face embeddings
       embeddings = face_recognition.face_encodings(rgb, localization)

       # loop over the embeddings, in case there are more faces
       for embedding in embeddings:
              # update lists
              facesEncoded.append(encoding)
              Names.append(name)
```

Model *HOG* bol použitý pri lokalizácií tvarí na Raspberry, použitie výkonnejšieho počítača umožňuje použitie výpočtovo náročnejšieho modelu *CNN* (*Convolutional Neural Network*) na zakódovanie jednotlivých tvárí.

```
...
boundingbox = face_recognition.face_locations(rgb,
               model="cnn")
...
```

Následná reprezentácia vyzerá podľa očakávaní nasledovne (Obrázok 34). Jedná sa o 128-bytový vektor.

```
        array([-0.06829347,  0.0403478 ,  0.05692631, -0.00711204, -0.03908385,
 0.01985077, -0.02288917, -0.04681916,  0.12782972, -0.02808866,
 0.25592899, -0.07889175, -0.21189477, -0.03892235,  0.0039623 ,
 0.09861147, -0.17991988, -0.09202938, -0.10659409, -0.12000347,
 0.11923119, -0.05611509, -0.02590239,  0.06579811, -0.25840005,
-0.29361823, -0.07291283, -0.16521527,  0.02971418, -0.04740055,
 0.01456314,  0.01381437, -0.11709445, -0.0006626 ,  0.04362732,
 0.04033753, -0.0013619 , -0.06382751,  0.18273813, -0.07670397,
-0.13606642,  0.14775534,  0.06835556,  0.22728595,  0.14334014,
 0.02477019,  0.11344254, -0.06318123,  0.14123075, -0.25970051,
 0.08998965,  0.18218802,  0.11113218,  0.08964648,  0.1595962 ,
-0.18742618, -0.0594664 ,  0.21518102, -0.10296884,  0.18545374,
 0.02179764, -0.00245963,  0.00147767, -0.1475517 ,  0.15457225,
 0.12038326, -0.19904923, -0.13533641,  0.10952065, -0.10010333,
-0.06094731,  0.06930198, -0.17465235, -0.19602323, -0.3033438 ,
 0.09109566,  0.46022585,  0.1394289 , -0.25748476,  0.0071908 ,
-0.08618882,  0.01711479,  0.01712966,  0.10120574, -0.04772914,
-0.04626248, -0.0812799 ,  0.02867967,  0.18301077,  0.0015154 ,
 0.00795677,  0.22776134, -0.00547237,  0.00606308, -0.05070718,
 0.04407106, -0.17992672, -0.05742128, -0.05420078, -0.00825219,
-0.025886  , -0.06623113, -0.00613969,  0.08162875, -0.19433302,
 0.16943051, -0.052636  ,  0.00931905, -0.05911341,  0.1007545 ,
-0.03919786,  0.07175939,  0.15096582, -0.31124231,  0.15664159,
 0.18767041,  0.09653929,  0.12427544,  0.04457551,  0.04771402,
 0.00483198,  0.05554433, -0.18652362, -0.06275128, -0.04346445,
-0.03968121, -0.03259256,  0.03934694])],
```

Obrázok 34: Reprezentácia tváre vo forme 128-bytového vektoru.

1.23 Rozpoznávanie tvárí

K samotnému rozpoznávaniu tvárí dochádza v skripte *faces_recognize.py*. V tomto skripte sa nachádza taktiež väčšina funkcií, ktoré táto diplomová práca využíva a nižšie budú postupne popísane. Pre názornosť sú niektoré hodnoty predávané skriptu ako argumenty v nasledujúcich ukážkach doplnené napevno.

Rozpoznávanie tvárí v tomto prípade prebieha v reálnom čase takže na Raspberry Pi je nutné použiť funkcie také, aby bol zabezpečený dostatočne plynulý beh programu. Nie je možné používať model *HOG* ako tomu bolo pri kódovaní tvári v skripte *faces_database.py* ale je využitý Haar Cascades klasifikátor.

Na začiatok je potrebné načítať známe tváre a inicializovať klasifikátor. Následne je detekcia, ktorú vykonáva funkcia *detection()* spustená ako samostatné vlákno (tzv. thread) . Vo funkcii *detection()* je pomocou cyklu *while* vykonávaná detekcia jednotlivých obrázkov z kamery. Na rozpoznávanie tvárí sú predpripravené dve verzie pôvodnej snímky, čierno-biela a Red-Green-Blue (RGB, červená-zelená-modrá). Čierno-biela slúži na lokalizáciu tváre. Tvár na snímke lokalizovaná funkciou *detectMultiscale()*. Red-Green-Blue (RGB) verzia slúži na vytvorenie kódovaní a následné porovnávanie s databázou známych tvárí. Porovnanie prebieha v cykle *for* pomocou funkciu *comapare_faces()*. Fotky sú porovnávané na základe vzdialenosti ich zakódovaných vektorov. Ak je pri porovnávaní prekročená tolerancia, fotka je označená ako zhoda. Toleranciu je možné upravovať pomocou dodatočného argumentu napríklad „*tolerance=0.5*“. Najlepšie výsledky boli docielené s ponechaním tejto hodnoty bez zmeny na východiskovej hodnote „*0.6“*. Možnosťou by bolo taktiež zaznamenávať priamo vzdialenosť jednotlivých tvárí, to sa však neosvedčilo ako výhodnejšia metóda, nakoľko spôsobuje zbytočné spomalenie programu bez viditeľnej pridanej hodnoty. Záznam vzdialeností by mohol mať význam pre systém rozpoznávajúci stovky osôb, bez kvalitných databáz a v zhoršených podmienkach, kde by sa tak dalo predísť niektorým nesprávnym autentifikáciám.

Po porovnaní detegovaných tvárí so všetkými tvárami v databáze je tvár vyhodnotená ako tvár osoby, s ktorou je počet zhôd najvyšší. V prípade žiadnej zhody je tvár považovaná za neznámu.

```
...
# loading face embeddings and classfier from the main folder (Cuckoo)
data = pickle.loads(open("encodings.pickle", "rb").read(), encoding="latin1")
classifier = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
...

...
# thread performing detection
d = threading.Thread(target=detection)
d.start()
...
```

```
def detection():
    ...

    while True:

            # reading the frame, resizing to 500px for faster processing, keeping h/w
ratio
            time= datetime.datetime.now()
            frame = vs.read()
            #frame= cv2.flip(frame,-1)
            frame = imutils.resize(frame, width=500)
            ...

            #preparing the image for detecting face (gray) and for recognizing (rgb)
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

            # faces being detected and localized in the image
            rects = classifier.detectMultiScale(gray, scaleFactor=1.2,
                    minNeighbors=5, minSize=(30, 30),
                    flags=cv2.CASCADE_SCALE_IMAGE)
            ...

            # creating the facial embeddings for all faces in the image
            embeddings = face_recognition.face_encodings(rgb, boxes)
            names = []

            for embedding in embeddings:
                    # comparing the faces from database with current ones
                    # if the treshold is overcome the match is recorded
                    matches = face_recognition.compare_faces("encodings.pickle",
                            encoding)
            ...
```

Použitie výkonnejšieho počítača umožňuje využiť na lokalizáciu tváre model *HOG* z knižnice Dlib, ktorý poskytuje vyššiu presnosť a nespôsobuje prílišné spomalenie behu programu.

```
...
boxes = face_recognition.face_locations(rgb,
               model="hog")
...
```

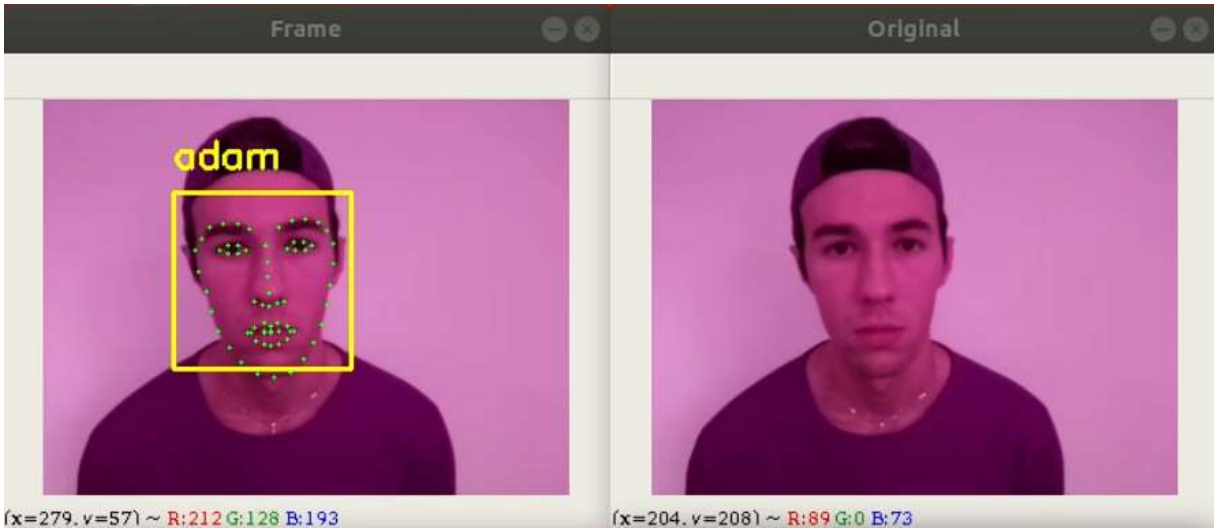
Výstup detekcie tváre je nasledovný (Obrázok 35).

Obrázok 35: Výstup detekcie tváre metódou HOG. Naľavo obraz s oblasťou detegovanou ako tvár, napravo vstupný obraz.

1.24 Kontrola očí

Ako dodatočný prvok k rozpoznaniu tváre, bola v snahe o zvýšenie celkovej vierohodnosti systému, pridaná funkcia sledovania regiónu očí, konkrétne zmeny veľkosti otvorenia očí, ktoré nastávajú napríklad pri žmurkaní tzv. Eye Aspect Ratio (EAR). Systém bol upravený tak, že autorizácia konkrétnej osoby nastane až po tom, čo sa zaznamenaná dostatočná zmena otvorenia očí, ku ktorej môže relatívne dôjsť nevedome prípadne po vynútenom žmurknutí. Takto sa systém stáva oveľa menej náchylný na identifikáciu osôb z fotiek. Podľa požiadaviek je možné parametre pre zmeny tvaru regiónu očí upraviť, aby vhodne zodpovedali kritériám, ktoré od systému vyžadujeme. Táto funkcionalita využíva detekciu kľúčových bodov tváre z knižnice *Dlib* (Obrázok 36). Na tvári je detegovaných 68 rôznych bodov, podľa potreby je ich možné zobraziť (Obrázok 37).

Obrázok 36: Zobrazenie kľúčových bodov tváre s ich označením z knižnice Dlib (Houssaini et al., 2019).



Obrázok 37: Umiestnenie kľúčových bodov na oblasť tváre v reálnom čase.

Funkcia *predictor()* je pridaná do cyklu *for* vo vnútri funkcie *detection()* a slúži na lokalizáciu siete kľúčových bodov na tvári (Obrázok 37). Ako je možné vidieť, cyklus *for* teraz okrem zakódovaných tvári (ako tomu bolo v predošlej kapitole), prechádza aj jednotlivé lokácie tvarí uložené v premennej *boxes*, na základe nich a čiernobielej verzie snímky, dokáže následne vytvoriť sieť kľúčových bodov na tvári. Je však nutné predávať snímku a detegovanú lokáciu tváre, nie zakódovanú tvár. Premenná *boxes* s uloženými lokáciami tvárí, má však údaje uložené vo forme jednoduchých štvoríc, a pre vytvorenie *dlib* objektu *rect*, ktorý dokáže funkcia *predictor* spracovať, je nutné použiť funkciu *css_to_rect*. Zvyšok ukážky sa venuje získaniu hodnôt z funkcie *predictor()*. Výsledná používaná hodnota EAR je priemerom hodnôt EAR ľavého a pravého oka, ktoré získame pomocou nižšie popísanej funkcie.

```
def detection():
    ...

    while True:

            ...
            for embedding, box in zip(embeddings,boxes):
                    # comparing the faces from database with current ones
                    # if the treshold is overcome the match is recorded
                    matches                                                 =
face_recognition.compare_faces("embeddings.pickle",
                            encoding)

                    rect = css_to_rect(box)
                    shape = predictor(gray, rect)
                    shape = face_utils.shape_to_np(shape)

                    # coordinates to compute the eye aspect ratio for both eyes
                    L_Eye = shape[lStart:lEnd]
```

```
                    R_Eye = shape[rStart:rEnd]
                    L_EAR= ea_ratio(L_Eye)
                    R_EAR = ea_ratio(R_Eye)

                    # average the eye aspect ratio together for both eyes
                    ear = (L_EAR + R_EAR) / 2.0
```

Výpočet EAR vychádza z (Houssaini *et al.*, 2019). Je to súčet vzdialeností dvoch párov nad sebou sa nachádzajúcich bodov oka podelená dvojnásobkom vzdialenosti dvoch bodov na horizontálnej línií. Tieto body môžeme vidieť v oblasti oka (Obrázok 36). Výpočet vykonáva funkcia *ea_ratio().*

```
def ea_ratio(eye):

#  distances between vertical eye points (4 points each eye)
    X = dist.euclidean(eye[1], eye[5])
    Y = dist.euclidean(eye[2], eye[4])

    # distances between horizontal eye points (2 points each eye)
    Z = dist.euclidean(eye[0], eye[3])

    # eye aspect ratio
    ear = (X + Y) / (2.0 * Z)


    return ear
```

Nasledujúca ukážka (Obrázok 38) zobrazuje proces detekcie zmeny otvorenia očí, ktorý je podmienkou pre autorizáciu danej osoby.

Obrázok 38: Najvyšší rad zobrazuje detegovanú rozpoznanú (identifikovanú) tvár osoby, ktorá však čaká na autorizáciu podmienenú zmenou otvorenia očí. Stredný rad zobrazuje moment zatvorenia očí, avšak autorizácia čaká na opätovné otvorenie. Podľa potreby je možné nastaviť dobu, počas ktorej, musia byť oči zatvorené aby autorizácia prebehla. Ak by nedošlo k následnému otvoreniu, hneď v nasledujúcom momente, alebo by oči neboli zatvorené dostatočne dlho, algoritmus by osobu neautorizoval. To taktiež minimalizuje možnosť použitia fotografie s otvorenými a zatvorenými očami pri snahe o autorizáciu. Posledný rad zobrazuje následné otvorenie očí a autorizáciu, ktorú vidieť na zmene farby a textovom výstupe. Taktiež je možné vidieť, že po úspešnej autorizácií sa zastaví nahrávanie videa, na cloud sa tak nahrá iba fotografia. Video by sa nahralo v prípade že by k úspešnej autorizácií neprišlo.

1.25 Verifikácia kódom

Jednou z funkcionalít, ktoré boli taktiež implementované, bola možnosť verifikácie pomocou *QR kódu*. Za týmto účelom je importovaná knižnica Pyzbar, ktorá umožňuje rozpoznanie aj iných typov kódov. Ak sa na obrázku nachádza *QR kód*, tak je dekódovaný, podľa potreby je ho možné uložiť do databázy a pod. Následne je porovnaná začiatočná časť kódu, ak sa zhodu s požadovaným heslom (v tomto prípade „test123“) je vstup povolený.

```
...
codes = pyzbar.decode(frame)


for code in codes:
    ...

    barcodeData = code.data.decode("utf-8")
    ...

if barcodeData[:7]== "test123":
    ...
```

Ukážka detegovaného kódu (Obrázok 39).

Obrázok 39: Obrázok z detekcie QR kódu. Červenou je zvýraznený samotný kód a jeho dekódovaná forma (test123). Keďže sa jedná o správne heslo, používateľ je následne informovaný o úspešnej autorizácií krátkym zvukovým signálom a rozsvietením zelenej diódy.

1.26 Detekcia pohybu

Predchádzajúce kapitoly popisovali spracovanie obrazu za účelom rozpoznania a spracovania regiónu tváre. Funkcia *detection()*, však obsahuje aj detekciu pohybu, ktorá zabezpečí spustenie nahrávania videa, ktoré sa v prípade, že počas stanoveného limitu

nepríde k úspešnej autentifikácií a následnej autorizácií, nahrá do cloudového úložiska. Za týmto účelom je využitá substrakcia pozadia. Jedná sa o substrakciu (zjednodušene odčítanie), súčasného obrazu od obrazu, ktorý považujeme za pozadie. Tento proces je jednoduchý, ak disponujeme fotografiou pozadia, ktoré je nemenné. V prípade tejto diplomovej práce, je však vysoko pravdepodobné, že je pozadie premenné, nakoľko sa počas dňa menia svetelné podmienky, môžu sa meniť predmety v zornom poli, nasmerovanie kamery. OpenCV disponuje niekoľkými algoritmami, ktoré je možné využiť na substrakciu pozadia. Pre túto diplomovú prácu boli implementované dva algoritmy: GMG a MOG.

GMG algoritmus, pomenovaný podľa svojich autorov a predstavený v roku 2012 (Godbehere *et al.*, 2012). Tento algoritmus kombinuje štatistický odhad pozadia a Bayesovskú segmentáciu jednotlivých pixelov. Na modelovanie pozadia sa používa prvých niekoľko snímok. Využíva pravdepodobnostný algoritmus segmentácie v popredí, ktorý identifikuje možné objekty v popredí pomocou Bayesovskej inferencie. Novšie pozorovania majú väčšiu váhu ako staré pozorovania, aby sa prispôsobili premenlivému prostrediu a osvetleniu. Na odstránenie nežiaduceho hluku sa vykonáva niekoľko morfologických filtrovaní.

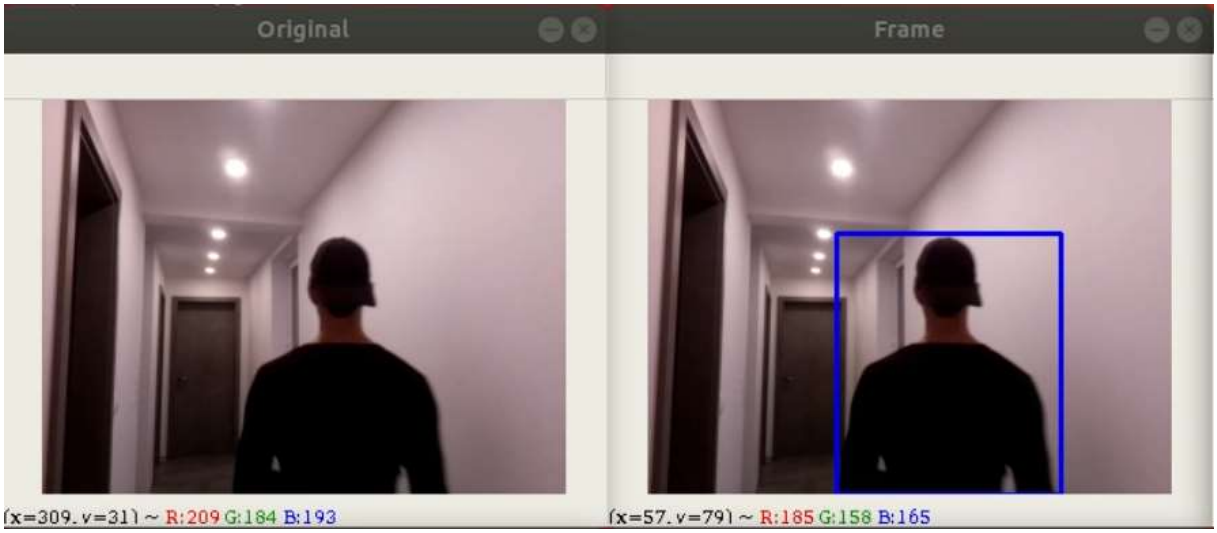
MOG je gaussovský algoritmus na báze segmentov pozadia a popredia. Predstavený v roku 2001 autormi P. KadewTraKuPong-om a R. Bowden-om. Metóda používa na modelovanie každého pixelu pozadia kombinácie K gaussovských distribúcií (kde K= 3 až 5). Váhy týchto kombinácií predstavujú časové pomery, počas ktorých tieto farby zostávajú na scéne. Pravdepodobné farby pozadia sú tie, ktoré zostávajú dlhšie a sú statickejšie.

V oboch prípadoch po prekročený hraničnej hodnoty „pohybu“ vo funkcií *treshold()* sa toto zaznamená a následne sú funkciou *findContours()* zaznačené kontúry týchto oblastí. Cyklus *for* tieto kontúry postupne prechádza, a ak je oblasť dostatočne veľká, tak vyhodnotí, že sa jedná o pohyb. Ukážka zobrazuje použitie algoritmu MOG. Implementáciu algoritmus GMG je vidieť ako komentované časti.

```
#kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))
#fgbg = cv2.bgsegm.createBackgroundSubtractorGMG()
fgbg = cv2.bgsegm.createBackgroundSubtractorMOG()

def detect_obj():

        while True:

            ...


            fgmask = fgbg.apply(frame)
            #fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernel)

            ret, thresh = cv2.threshold(fgmask, 126, 255,cv2.THRESH_BINARY)
            thresh = cv2.dilate(thresh, None, iterations=2)
            moving_regions = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
```

```
            cv2.CHAIN_APPROX_SIMPLE)
        moving_regions = imutils.grab_contours(moving_regions)


        for region in moving_regions:
            # if the contour is too small, ignore it
            if cv2.contourArea(region) < conf["min_area"]:
                continue
            # compute the bounding box for the contour, draw it on the frame,
            # and update the text
            (x, y, w, h) = cv2.boundingRect(region)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            recording=True Online úložisko
```

Pri spustenej detekcií pohybu môžeme výstup detekcie zobraziť v prípade potreby aj na výstupnom obrázku kde je vidieť ako modro ohraničená zóna (Obrázok 40).



Obrázok 40: Ukážka z detekcie pohybu v reálnom čase. Naľavo vstupný obrázok, napravo obrázok s modro ohraničenou zónou vyhodnotenou ako pohyb pomocou MOG.

1.27 Ukladanie dát

Údaje ako história vstupov, fotografie z úspešných aj neúspešných verifikácií, textové dáta o ďalších udalostiach sú sprístupnené online. Na túto funkciu bol využitý Dropbox, obľúbená cloudová služba, ktorá umožňuje bezplatné použitie svojej API (Aplication Program Interface) a umožňuje tak programátorsky rozšíriť možnosti Dropboxu do vytvorenej aplikácie (Dropbox, 2020). Platformu Dropbox Python SDK je jednoducho možné nainštalovať pomocou príkazu `$pip install dropbox`. Dokumentáciu k použitiu API je možné nájsť na (Dropbox, 2020). Vďaka Dropboxu majú všetky osoby s prístupom k účtu možnosť vzhliadnuť nahrané súbory online a narábať s nimi. Zároveň sa nemusia báť, že by ich dáta boli uložené na nezabezpečenej alebo inak nespoľahlivej platforme.

Po načítaní konfiguračného súboru, kde sa nachádza aj prístupový token pre Dropbox (`dropbox_access_token`) následne vytvoríme premennú *user*. Funkcia *dbox()* slúži na nahrávanie súborov do účtu, v prípade tejto diplomovej práce sú to fotky, videá a textový súbor so zoznamom všetkých povolených a zamietnutých prístupov. Popísaná je funkcia nahrávania fotografií a textového súboru. Vstupov do tejto funkcie je fotografia a čas kedy bola daná fotka vytvorená, ktorý je zakomponovaný do názvu fotky. Postup je obdobný aj pri funkciách nahrávajúcich video.

```
...

conf = json.load(open("conf.json"))
user=None

    if conf["use_dropbox"]:
        user=dropbox.Dropbox(conf["dropbox_access_token"])
        print("Dropbox client created")
    ...

def dbox(frame,time):
    ...

    #tries to upload the files to Dropbox account
    try:
        user.files_upload(f.read(), path2, mode=WriteMode('overwrite'))
        user.files_upload(open(t.path, "rb").read(), path)
        t.cleanup()
    except ApiError as err:
        # Listening for the error of not enough space in the account
        if (err.error.is_path() and err.error.get_path().error.is_insufficient_space()):
            sys.exit("!!! Not enough space to upload the files !!!")
        elif err.user_message_text:
            print(err.user_message_text)
            sys.exit()
        else:
            print(err)
            sys.exit()

    ...

    return
```

Fotografie a videá uložené do cloudovej služby môžu potom vyzerať nasledovne (Obrázok 41).

Obrázok 41: Súbory nahrané do cloudovej služby. Fotografie sú nahrávane vo formáte jpg, videá vo formáte avi

Podľa potreby sa môžu ukladať záznamy z detekcie kódov, textové súbory a pod. (Obrázok 42)

Obrázok 42 : Zoznam povolených prístupov LOG.txt a fotografie z detekcie kódu.

1.28 Streamovanie

Streamovanie videa prebieha pomocou už spomínaného webového micro-frameworku Flask. Na začiatku inicializujeme premennú *picUpload*, ktorá obsahuje obrázok, ktorý má byť viditeľný z webu. Pomocou funkcie *threading.Lock()* zabezpečíme bezproblémový beh programu. Funkcia totiž zabraňuje čítaniu obrázka, ktorý je práve nahrávaný iným „*thread-om*“ (pozn. thread= vlákno). Taktiež je tým zabezpečená možnosť vzhliadania živého videa viacerým užívateľom súčasne respektíve vo viacerých oknách prehliadaču naraz. Následne je inicializovaná aplikácia *Flask*.

```
...
# lock is useful for more apps accessing the stream and for the exchange of picUpload is
# therfore safe and stable

picUpload = None
lock = threading.Lock()
```

Jednoduchá funkcia *index()* nám vráti šablónu webovej stránky na základe súboru *index.html*.

```
app = Flask(__name__)

# simple webpage template loaded
@app.route("/")
def index():
    return render_template("index.html")
    ...
```

Funkcia *generate()* slúži na zakódovanie vstupného obrázku do formátu *.jpeg*. Vďaka premennej *lock* je zabezpečené bezpečné správanie jednotlivých vlákien programu ako už bolo spomínané.

```
    ...

def generate():
    # takes global picUpload and lock from function detection()
    global picUpload, lock

    # looping over the pictures to be uploaded
    while True:
        # this ensures loops waits for the lock
        with lock:
            # makes sure there is picture
            if outputFrame is None:
                continue

            # create JPEG from picture
            (correct, picJPEG) = cv2.imencode(".jpg", picUpload)

            # check of correct encoding
            if not correct:
                continue

        # bite format of piture for upload
        yield(b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' +
            bytearray(picJPEG) + b'\r\n')
        ...
```

Bližšie informácie k definovaniu *routes* (ciest) a *views* (pohladov) je možné získať na (Birchard, 2019). Funkcia *video_feed()* slúži na volanie funkcie *generate()*. Výstupom z tejto funkcie je v jednoduchosti živé video zakódované ako bytové pole. Toto je následne zobrazené prehliadačom.

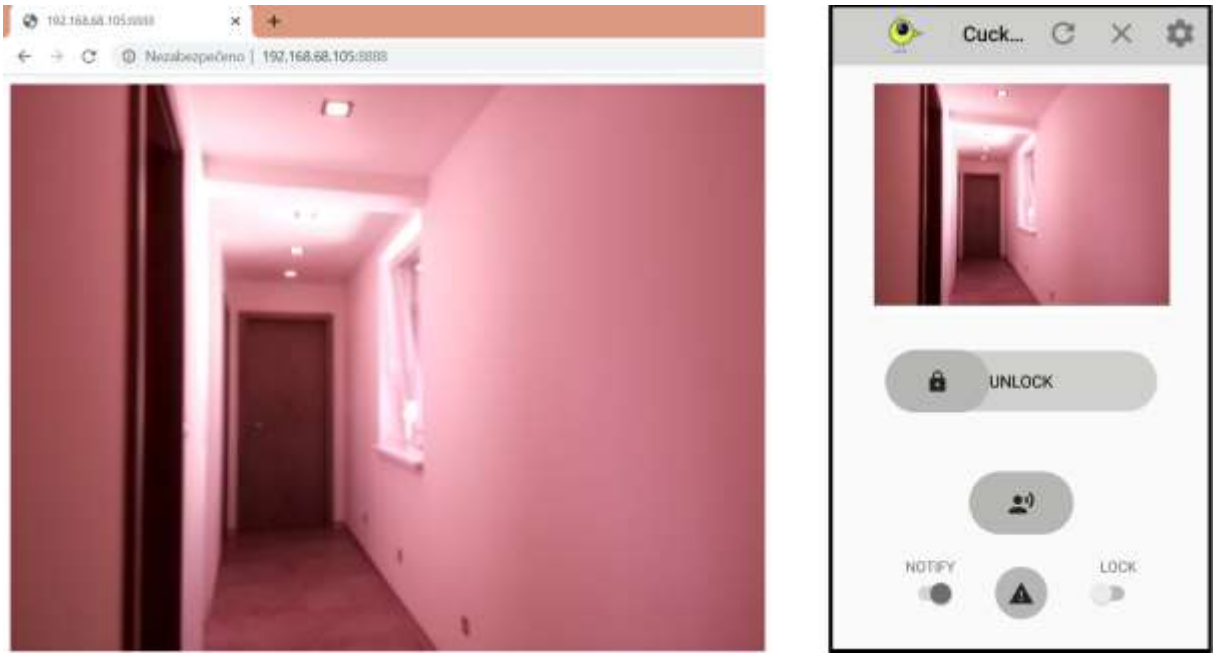
```
...

@app.route("/video_feed")
def video_feed():
    # return the response generated along with the specific media
    # type (mime type)
    return Response(generate(),
```

```
        mimetype = "multipart/x-mixed-replace; boundary=frame")
...
```

Posledným krokom je spustenie aplikácie *Flask*.

```
...
#runs the Flask app
app.run(host="192.168.68.105"], port="8000", debug=True,
        threaded=True, use_reloader=False)
...
```

Streamované video je následne možné vzhliadnuť či už priamo v prehliadači alebo v Android aplikácií (Obrázok 43).



Obrázok 43: Zobrazenie streamovaného videa v prehliadači a Android aplikácií.

1.29 Odosielanie videa na server

Všetky predchádzajúce popisované ukážky boli implementované na Raspberry Pi. Takto bolo možné dosiahnuť uspokojivého výsledku rozpoznávania s rýchlosťou výstupu okolo ~2 až 3 snímok za sekundu (bližšie informácie sú poskytnuté v ďalšej kapitole).

Pre potreby využitia serveru bolo však nutné vytvoriť nové skripty, prípadne upraviť existujúce. Ukážky popisované v predchádzajúcich podkapitolách boli v rovnakej forme (okrem spomínaných vylepšení) implementované na server. Nasledujúca časť popisuje implementáciu knižnice *ZMQ* na preposielanie obrazu.

Na Raspberry bol vytvorený skript *sending.py*, ktorý má za úlohu odosielať obraz na server. Na začiatku inicializujeme objekt *ImageSender*, so „socket“ adresou serveru cez ktorú

prebieha následná komunikácia. Pomocou cyklu *while* následne zachytávame obraz z kamery a spolu s vlastným menom Raspberry ho odosielame (skript je upravený aby bolo jednoducho možné pridať v prípade potreby viacej Raspberry Pi). V cykle je taktiež podmienka ktorá čaká na vstup od tlačidla na Raspberry, po ktorého stlačení je používateľ mobilnej aplikácie notifikovaný prostredníctvom funkcie *sendPush()* a môže vykonať potrebné kroky.

```
...
sender = imagezmq.ImageSender(connect_to="tcp://192.168.0.105:5555")

# getting the own hostname and initializing the camera stream
name = socket.gethostname()
vs = VideoStream(usePiCamera=True).start()
...

while True:
    # capture picture and send it along with the name (useful for more senders)
    picture = vs.read()
    sender.send_image(name, picture)

    # waiting for the button press, send notification to mobile app
    if(th.button.is_pressed) and (datetime.datetime.now() - click).seconds >=1 :
        click=datetime.datetime.now()
        sendThread= Thread(target=sendPush)
        sendThread.start()
```

Funkcia *sendPush()* na nižšie popísaný formát. Odošle notifikáciu s názvom *BELL* a popisom „Niekto je pri dverách“. Notifikovaný budú všetci odoberatelia témy „*bell*“ k odoberaniu ktorej, sa môžu prihlásiť prostredníctvom mobilnej aplikácie. V prípade, že notifikovaný byť nechcú alebo nechcú byť rušený jednoducho sa v aplikácii od notifikácií odhlásia. Použitie takzvaných „odoberateľov“ je značne vhodnejšia verzia zvolenia cieľového publika notifikácií pre potreby tejto diplomovej práce.

Druhou možnosťou je ukladanie registračných identifikátorov zariadení používajúcich aplikáciu. Tento identifikátor sa vytvorí napríklad pri jej stiahnutí a inštalácií a následne je napríklad možné cielene kontaktovať jednotlivé zariadenia. Tento typ implementácie je však vhodný pre iné typy projektov.

```
def sendPush():

    data_msg = {
        "title" : "BELL",
        "body" : "Someone is at the door.",
    }
    notify = FCMNotification(api_key="AA ~(deleted for security reasons)~ k2nQO")
    print("Notification will send")
    res= notify.notify_topic_subscribers(topic_name="bell", content_available=True,
        data_message=data_msg)
```

1.30 Prijatie videa na serveri

Príkazy zabezpečujúce prijatie fotiek na serveri sú implementovaná do skriptu *faces_recognize.py*. Nahradíme časť kódu, ktorá predtým získavala obraz z kamery, potrebnými príkazmi. Inicializujeme *ImageHub()*, následne v rámci cyklu *while* vo funkcií *detection()* snímok získame a odošleme odpoveď pre Raspberry.

```
...

imageHub = imagezmq.ImageHub()
...

def detection():
    ...

    while True:

            (name, frame) = imageHub.recv_image()
            imageHub.send_reply(b'OK')
            ...
```

1.31 Komunikácia prostredníctvom „socket“

„Socket“ (voľný preklad zásuvka, pätica – pozn. autora) je takzvaný koncový bod (ednpoint) spojenia. Je definovaná kombináciou IP adresy a číslom portu (port je identifikátor konkrétnej služby alebo programu). „Sockets“ existuje viacej druhov, v tejto práci sa budeme venovať iba tzv. INET Sockets (InterNetwork~ IPv4, (INET6~ IPv6)) a iba STREAM Sockets (používajúcim Transmission Control Protocol (TCP)). Jednoducho povedané, Socket umožňuje komunikáciu medzi dvoma rôznymi procesmi, v rámci jedného alebo rôznych zariadení.

V rámci tejto diplomovej práce, boli využité na komunikáciu medzi Android aplikáciou a Raspberry Pi, ako aj medzi počítačom HP Pavilion dv6 a Raspberry Pi. Posielané sú príkazy, ktoré má Raspberry vykonať, a tak umožniť vzdialenú správu, keďže po „offloadovaní“ rozpoznávania na druhý počítač, je to práve on, kto je zodpovedný za autentifikáciu a autorizáciu. Nasledovný program *listening.py* je implementovaný na Raspberry Pi, za účelom prijímania príkazov.

Nižšie je prezentovaná funkcia *start_server*. Príkazy na začiatku jej tela vytvoria a nastavia *socket* podľa požadovaných argumentov. Nasleduje pokus spojiť *socket* s predanou hosťovskou adresou a portom. Argument `host = ,‘` umožňuje pripojenie k *socket* pomocou akejkoľvek adresy, ktorou zariadenie disponuje. Následne príkaz `soc.listen(5)` určí, že počet možných žiadostí o pripojenie v rade je 5 (štandardne používaná maximálna hodnota), až potom dôjde k odmietnutiu ďalších pripojení. V cykle *while*, dôjde k nadviazaniu spojenia s druhým zariadením (v prípade tejto práce s počítačom HP alebo Android aplikáciami). Vytvorí sa vlastné vlákno pre každé spojenie,

v rámci ktorého je volaná funkcia *client_thread()*, ktorá zabezpečí prijatie príkazu a jeho následné vykonanie.

```
Def start_server():
    host = ,‘
    port = 5555
    soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    soc.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    print(„Socket created, server started“)

    try:
        soc.bind((host, port))
    except:
        ...
        sys.exit()

    soc.listen(5)       # up to 5 requests are qued
    print(„Socket listening“)

    # infinite loop
    while True:
        connection, address = soc.accept()
        ip, port = str(address[0]), str(address[1])
        print(„Connected with „ + ip + „:“ + port)

        try:
            Thread(target=client_thread, args=(connection, ip, port)).start()
        except:
            print(„Thread did not start.“)
            traceback.print_exc()

    soc.close()
```

1.32 „At Startup“

Pri používaní Raspberry Pi ako inteligentného priezoru, nie je žiaduce, aby sme museli pri každom spustení, alebo v prípadne výpadku elektrického prúdu, pripájať toto zariadenia k monitoru a ručne spúšťať potrebné programy. Z toho je dôvodu je potrebné, po úspešnom naprogramovaní zariadenia, zabezpečiť automatické spúšťanie požadovaných programov po každom zapnutí (reštarte).

Za týmto účelom vytvoríme tzv. „shell script“ *startup.sh* s nasledujúcim obsahom (viď nižšie), ktorý bude slúžiť na spúšťanie programov *sending.py* a *listening.py*. Aby to bolo možné, je potrebné najskôr vstúpiť do Python virtuálneho prostredia, ktoré bolo využívané pri tejto práci.

```
Source /home/pi/.profile
workon opcv4
```

```
cd /home/pi/cuckoo
python listening.py
python sending.py
```

Aby bol skript spustiteľný použijeme príkaz `chmod`.

```
$ chmod +x startup.sh
```

Otvoríme zložku *contrab*.

```
$ sudo crontab -e
```

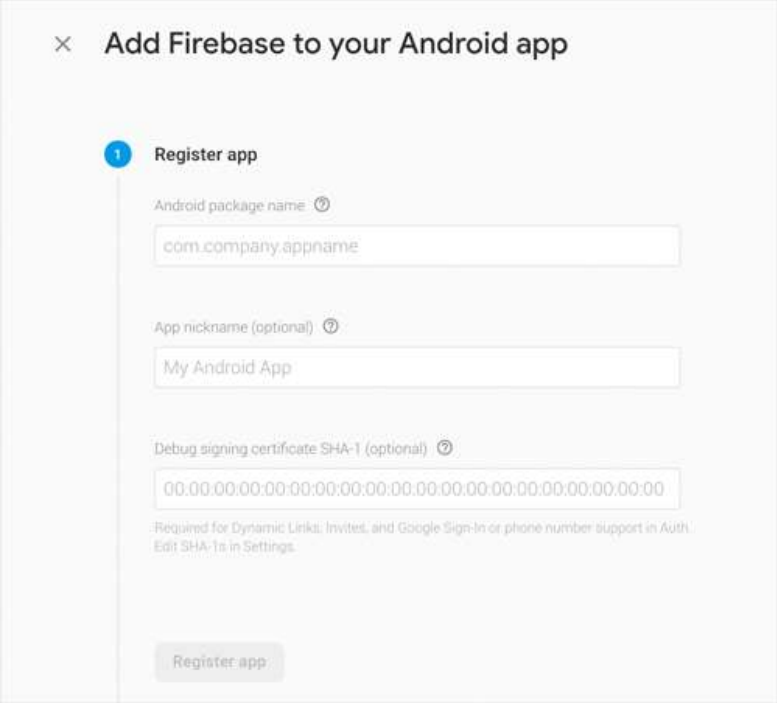
Na jej koniec pridáme nasledujúci príkaz.

```
@reboot /home/pi/startup/startup.sh
```

1.33 Mobilná aplikácia

Mobilná aplikácia bola vytvorená na umožnenie vzdialenej správy zariadenia, aby mal používateľ možnosť na diaľku povoliť autorizáciu osobe, ktorá o takéto povolenie požiadala. Mobilná aplikácia bola vytvorená pre operačný systém Android prostredníctvom Adroid Studia. Aplikácia (pomenovaná *Cuckoo*) komunikuje s Raspberry prostredníctvom „socket“. Nakoľko sa nejedná o predmet tejto diplomovej práce vytvorenie aplikácie a jej funkcionalít, nie je v tejto práci bližšie popísané. Komunikácia cez socket prebieha na obdobnom princípe ako komunikácia medzi Raspberry a počítačom a z dôvodu obmedzeného rozsahu práce, nie je bližšie popisovaná. Podrobnejšie popísaná je iba implementácia platformy FireBase, ktorá môže z hľadiska funkcionalít a jednoduchej škálovateľnosti byť pre čitateľa zaujímavá.

Pred implementáciou FireBase do mobilnej aplikácie je dobré sa uistiť že používaná verzia Android Studio, je tá najaktuálnejšia, a že sú splnené predpoklady použitia, bližšie popísané online, dostupné aj s ďalšími informáciami, napríklad k implementácií pre Android z *www.firebase.google.com/docs/android/setup* . Pri vytvorení projektu sa ako prvé zobrazí nasledujúce okno (Obrázok 44).

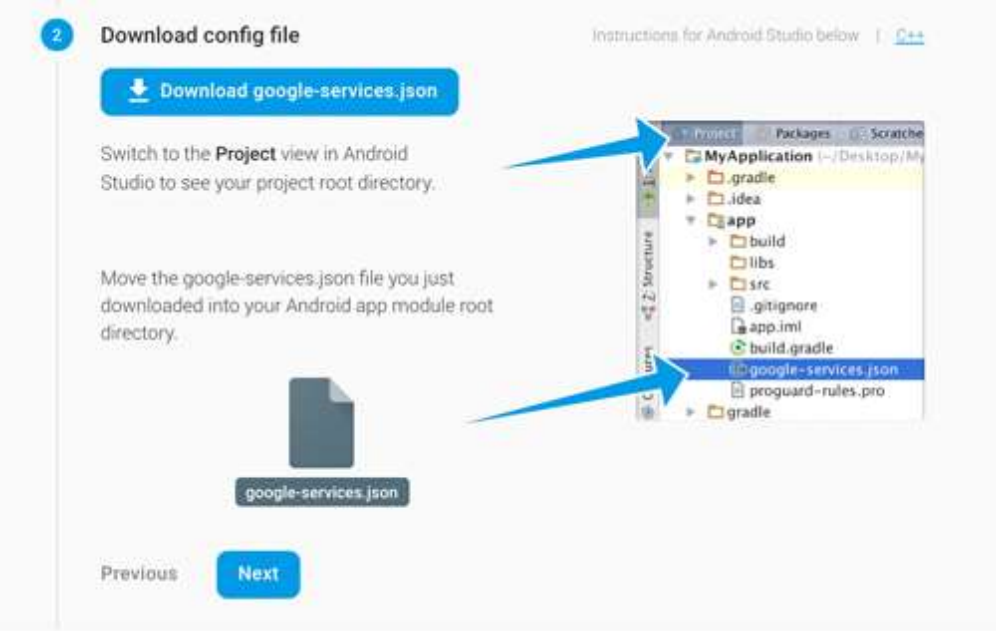


Obrázok 44: Prvý krok vytvorenia projektu.

Aby sme SHA-1 vygenerovali je nutné použiť nasledujúce príkazy.

```
$ cd C:\Program Files\Java\jdk1.8.0_201\bin>

$ keytool -list -v -alias androiddebugkey -keystore C:/Users/User/.android/debug.keystore
 -list -v

$ keytool -exportcert -list -v -alias androiddebugkey -keystore
C:/Users/User/.android/debug.keystore
```

Skopírujeme SHA-1 a zadáme ho do formuláru. Ďalej doplníme ostatné požadované informácie a po stiahnutí umiestnime súbor *google-service.jsom* ako to zobrazuje ukážka (Obrázok 45).



Obrázok 45: Druhý krok vytvorenia projektu.

Upravíme *build.gradle* súbor aplikacie, tak že na začiatok pridáme nasledovný riadok

```
apply plugin: ‚com.google.gms.google-services‘
```

Následne pridáme potrebné závislosti

```
dependencies {
    ...
    implementation ‚com.google.firebase:firebase-messaging:20.1.2‘
    implementation ‚com.google.firebase:firebase-analytics:17.2.3‘
    }
```

Do závislostí v projektovom *build.gradle* pridáme

```
dependencies {
    ...
    classpath ‚com.google.gms:google-services:4.3.3‘
    }
```

Vytvoríme tzv. „service súbor“ napríklad *MyFirebaseMessagingservice*. Začiatok súboru je nasledovný. Ďalej špecifikujeme správanie a požadované funkcie.

```
Public class MyFirebaseMessagingService extends FirebaseMessagingService {

    @Override
    public void onNewToken(String s) { ...
```

Taktiež je nutné *MyFirebaseMessagingservice* špecifikovať v súbore *Android manifest.*

```
<service
    android:name=“.MyFirebaseMessagingService“
    android:exported=“false“>
    <intent-filter>
        <action      android:name=“com.google.firebase.MESSAGING_EVENT“      />
    </intent-filter>
</service>
```

Ukážka implementácia spomínanej funkcie na odoberanie notifikácií (subscription) vyzerá nasledovne (viď nižšie). Výhodou použitia odoberania rozličných tém (v tomto prípade téma „bell“) je následná rýchla správa takýchto odberateľov a bez údržbové používanie. Charakteru tohto projektu vyhovuje práve možnosť prihlasovania a odhlasovania odberu.

```
Subscribe.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
{
        if (isChecked) {
            Log.d(TAG, „Subscribing to notifications“);
            // Subscribing
FirebaseMessaging.getInstance().subscribeToTopic(„bell“)
                    .addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            String msg = „Notifications ON“;
                            if (!task.isSuccessful()) { ...
```

Realizácia a testovanie

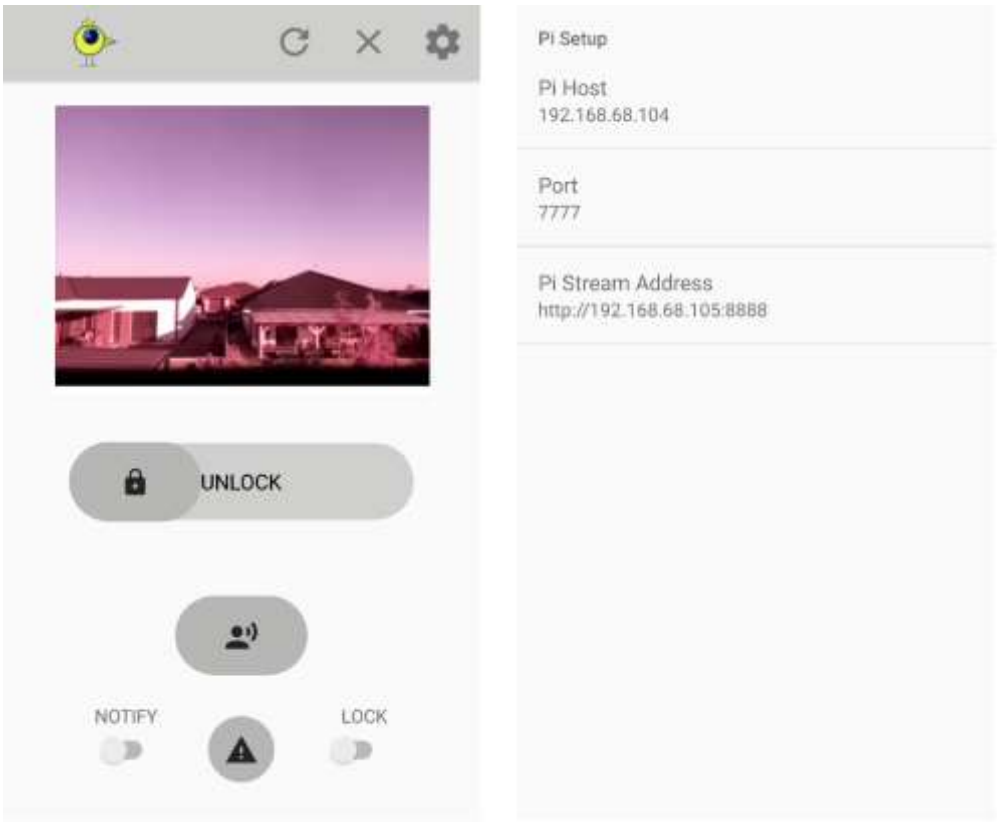
Po úspešnej príprave a vyhotovení praktických záležitostí, je možné pristúpiť k realizácií hmotného výstupu projektu. Raspberry Pi bolo umiestnené do prispôsobeného krytu, ktorý umožňuje súčasné pripojenie kamery a jej uzavretie priamo do puzdra a pripojenie Traffic HAT aby bolo možné sledovať výstup zo zariadenia (Obrázok 46). Z vonkajšej časti pripojený reproduktor slúži na informovanie osoby o výsledku autorizácie prípadne prehrá správu po stlačení tlačidla v mobilnej aplikácii.

Výstup z rozpoznávania je realizovaný ako svetelný signál diód, zvukový signál a grafický výstup na streamovanom obraze. Súčasná verzia inteligentného priezoru môže plne slúžiť ako forma inteligentného dochádzkového systému, kontroly a záznamu vstupu osôb do budovy (domu, firmy, objektu), bezpečnostný kamerový systém s detekciou pohybu a tvárí, atď.

Obrázok 46: Raspberry Pi umiestnené v kryte, upevnené pri vstupe do miestnosti.

Úvodná stránka mobilnej aplikácie obsahuje viaceré funkcie (Obrázok 47). Autorizácia (spustenie zelenej diódy a zvukového signálu) je umožnená po aktivovaní posuvného tlačidla v stredovej časti. Tlačidlo je posuvné z dôvodu, aby nedošlo k otvoreniu po náhodnom stlačení. Tlačidlo pod ním (s ikonou osoby) umožňuje prehrať správu „Please wait, I will be there in a minute“ (v slovenskom znení „Prosím počkajte, o chvíľu prídem.“). Spodné tlačidlo v strede (trojuholník s výkričníkom), slúži ako výstražný signál

v prípade, že chceme upozorniť osobu pohybujúcu sa v okolí priezoru. Výstražný signál znie tak dlho, ako dlho je tlačidlo stlačené.



Obrázok 47: Mobilná aplikácia, hlavná stránka (vľavo) a nastavenia (vpravo). Pohľad kamery smeruje do diaľky z okna izby, nejedná sa o umiestnenie pri použití.

Posuvné tlačidlo „NOTIFY“ slúži na aktivovanie alebo deaktivovanie notifikácií, ktoré užívateľ dostáva po stlačení tlačidla na priezore. Funkcia „LOCK“ slúži na „uzamknutie“ (autorizácia rozpoznaním tváre alebo prostredníctvom *QR kódu* nie je možná) a odblokovanie je možné, iba prostredníctvom mobilnej aplikácie. Nastavenie hodnôt potrebných pre pripojenie aplikácie k priezoru a zobrazenie videa z kamery slúži sekcia „Pi Setup“ (Obrázok 47), ktorú je možné otvoriť pomocou ikony „*Settings*“ (Nastavenia) v pravom hornom rohu.

Aplikácia bola navrhovaná aj s ohľadom na použitie na tabletoch a väčších obrazovkách, čo by pri strategickom umiestnení tabletu s permanentne bežiacou aplikáciou vo vnútorných priestoroch slúžilo ako centrálny systém vzdialenej správy bez nutnosti použitia mobilného zariadenia. Z toho dôvodu sú aj tlačidlá „NOTIFY“ a „LOCK“ umiestnené na hlavnej obrazovke, aby bol ich stav vždy viditeľný a prístup k nim rýchly.

V rámci testovania bolo Raspberry Pi bolo umiestnené v blízkosti dverí, v dosahu elektrickej zásuvky a boli zaznamenávané údaje v priebehu niekoľkých dní. Nakoľko sa nejedná o obyčajný systém záznamu obrazu ale o rozpoznávanie tvárí, je nutné niektoré parametre na základe pozorovaných výstupov prispôsobiť umiestneniu priezoru. Na základe umiestnenia v priestore je dobré upraviť veľkosť sekcie, ktorú algoritmus ešte

môže považovať za tvár. Týmto spôsobom sa môžeme vyvarovať rozpoznávania z veľkej diaľky, zbytočným dvojitým autorizáciám, nepresným rozpoznaniam a pod, najmä pri použití metódy Haar Cascades na detegovanie tvárí.

Pre dosiahnutie čo najvyššej presnosti pri rozpoznávaní je vhodné fotky osôb do databázy známych tvári nahrávať v svetelných podmienkach podobných svetelným podmienkam prostredia, kde sa rozpoznávanie vykonáva. Zvýšiť spoľahlivosť môžeme aj nahraním väčšieho množstva fotiek, tej istej osoby (ak je to možné viac ako 10). Dodržaný by mal byť aj približne rovnaký počet nahraných fotiek u jednotlivých osôb. Za ideálne sa dá považovať, ak sú fotky už pri ich vytváraní robené so zámerom použitia v databáze, čiže tvár je centralizovaná v obraze, približne zaberajúca rovnakú oblasť na každej fotke.

Na základe umiestnenia priezoru, je dobré zvoliť vhodnú metódu detekcie pohybu, ktorá zabezpečí spustenie nahrávania. Metóda GMG je citlivá aj na tie najmenšie zmeny a tiene, takže je vhodná do väčších priestorov, vonkajšieho prostredia a monitorovanie z väčšej diaľky, kde nedochádza k výrazným odrazom tieňov napríklad na stenách, skle alebo náhlym zmenám osvetlenia. Vo vnútorných priestoroch je lepšie využiť metódu MOG, ktorá nedeteguje tiene a celkové zmeny tak citlivo (Obrázok 48). Táto metóda poskytuje dobrú kvalitu detekcie a podľa potreby je vhodné upraviť argumenty funkcií aby nedochádzalo k príliš citlivej alebo necitlivej detekcií a taktiež parameter, ktorý definuje ako dlho po zotrvaní v obraze je objekt považovaný za pozadie. Keďže bol priezor umiestnený vo vnútorných priestoroch hlavná využívaná metóda detekcie je MOG.

Obrázok 48: Výstup pri detekcií pohybu v reálnom čase. Vstupný obrázok je vidieť v hornej časti. V dolnom rade naľavo výstup v GMG detekcie a napravo MOG detekcie. Ako je možné vidieť GMG deteguje značné množstvo tieňov a zmien ktoré nastanú pri pohybe, dokonca aj bez pohybu v zornom poli kamery.

Nasledujúca tabuľka (Tabuľka 2) zobrazuje porovnanie rýchlosti spracovávania obrazu formou počtu spracovaných snímok za sekundu, pri použití rôznych metód detekcie tváre. Práve detekcia, sa dá považovať za kľúčový krok pri rozpoznávaní tváre v rámci tohto projektu, nakoľko práve presne a rýchlo lokalizovaná tvár v obraze mala veľký vplyv na celkovú kvalitu rozpoznávania. Nie je možné porovnávať tvár, ktorá v obraze nebola detegovaná, prípadne bola detegovaná zle. Údaje vhodne vykresľujú výpočtové rozdiely, ktoré nastávajú pri použití Raspberry Pi v porovnaní s výkonnejším počítačom. Z údajov vyplýva značné spomalenie, ktoré nastáva pri použití výpočtovo náročnejších metód na lokalizáciu tváre ako je HOG alebo CNN, ale v tomto prípade ide spomalenie ruka v ruke so zvýšením kvality lokalizácie.

Je nutné poznamenať, že v prípade sledovania videa online sa rýchlosť spracovania zastaví na hodnote mierne nad ~10 FPS, ktorá predstavuje pri danom rozlíšení a implementácií pomyselnú hranicu rýchlosti nahrávania na server. Pre udržanie stability programu pri nahrávaní obrazu online je totiž potrebná správna práca s funkciou *Lock()*. Sledovanie videa online nespôsobuje takmer žiadne spomalenie pri rýchlostiach spracovania nižších ako spomínaných ~10 FPS.

	HAAR	HOG (FPS)	CNN
RPi	2,6	0,98	/
HP	20,3	10,5	0,6

Tabuľka 2: Porovnanie rýchlosti spracovávania obrazu na Raspberry Pi a počítači, pri použití rôznych detekčných metód. Porovnanie bolo vykonané pri rovnakom video vstupe a pri 24 fotografiách v databáze. Priemerná rýchlosť veľmi záleží na veľkosti databázy (z dôvodu porovnávania s jednotlivými fotografiami) a pomeru času tváre v obraze. Daná rýchlosť zodpovedá rýchlosti pri výskytu tváre v obraze 40% z celkového sledovaného času.

Cieľom tejto práce nie je detailné hodnotenie presnosti a kvality jednotlivých algoritmov na lokalizáciu, vytváranie reprezentácií alebo detegovania oblastí tváre, nakoľko používané algoritmy sú veľmi dobre zdokumentované a ich slabiny a silné stránky sú vopred známe. Existuje o nich veľké množstvo rôznych publikácií, za zmienku stoja napríklad (Mantoro *et al.*, 2018) (Guennouni *et al.*, 2017) (Sharifara *et al.*, 2014) (Osowski and Siwek, 2020) (Li and Huo, 2016) (Khammari, 2019) (Kumar *et al.*, 2019) (Yang and Wang, 2015) a mnohé ďalšie. V stručnosti by sa porovnanie použitých metód dá zhrnúť nasledovne.

Haar Cascades Classifiar je stále velmi často používaným „štandardom“ pri detekcii tvárí a s mnohými vylepšeniami v posledných rokoch. Jeho hlavnými výhodami sú rýchlosť, jednoduchosť a detekcia aj malých tvárí v obraze. Nevýhodou je však vyššie percento

„false positive“ (nesprávne detegovaných) tvárí. Ďalej je to malý rozsah uhlov, v ktorých algoritmus tvár deteguje (podľa použitého detektora či už predných tvárí, bočných atď.) a neschopnosť detegovať príliš prekrytú tvár.

HOG je v súčasnosti veľmi častou používanou metódou detekcie tváre, ktorá poskytuje niektoré výhody oproti vyššie spomínanému Haar Cascades. HOG dokáže detegovať aj tváre, ktoré sú natočené ku kamere mierne bokom a taktiež viac prekryté. Ide o model ktorý vďaka svojej „ľahkosti“ vykazuje dobrú rýchlosť spracovávania. Nevýhodou ktorá môže byť citeľná pri iných projektoch je detekcia iba tvárí, veľkosti väčšej ako 80x80 pixelov čo by v iných aplikáciách mohlo byť potrebné (v takom prípade je možné natrénovanie vlastného detektoru).

CNN je najpokročilejšia metóda detekcie požitá v tejto práci. Výhodou je detekcia aj značne natočených a prekrytých tvárí. Pri spracovávaní obrazu čisto na CPU však táto metóda nie je vhodná pre detekciu v reálnom čase. Podobne ako v prípade HOG, nedeteguje príliš malé tváre v obraze a ako je možné vidieť z ukážky detegovaná oblasť tváre je menšia ako tomu bolo pri metóde HOG. To môže mať vplyv na využiteľnosť pri niektorých aplikáciách tejto metódy.

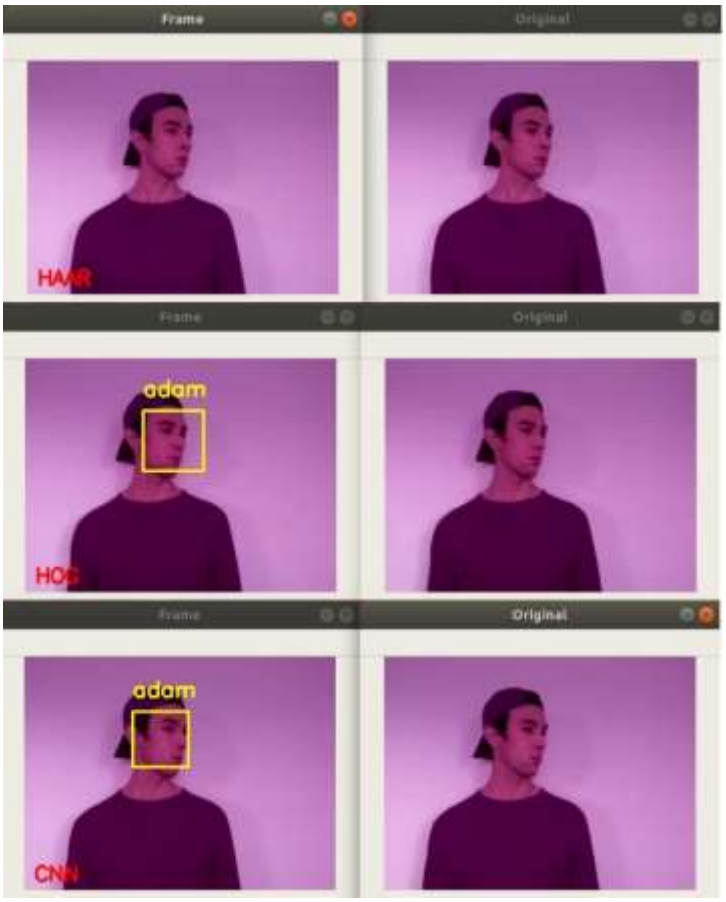
Nasledujúca ukážka zobrazuje porovnanie detekcie pri bočnom pohľade (Obrázok 49). Podrobnejšie porovnania je možné nájsť v prílohe A a B.

Samotný proces vyhodnotenia obrazu z priezoru prebieha nasledovne; Ak priezor zaznamená pohyb, spustí zaznamenávanie obrazu. Tento záznam pokračuje, kým nedôjde k jednej z dvoch podmienok. Platnej autentifikácií niekoho v obraze, prípadne prekročeniu požadovaného času nahrávania, kedy sa nahrávka automaticky nahrá do cloudovej služby aj spolu s fotografiou, ktorá spustila nahrávanie. Ak počas nahrávania dôjde k úspešnej autentifikácií, tak sa nahrávanie zastaví a nahrávka odstráni, nakoľko sa považuje sa nepotrebnú a na cloud je nahraná iba fotografia autorizovanej osoby a tiež je aktualizovaný zoznam príchodov.

V prípade prístupu heslom sa nahrávanie nepreruší a na cloud je nahrané video aj fotografia z autorizácie kódom. Autorizáciu je možné nahradiť použitím Android aplikácie, kde sa očakáva, že osoba požadujúca autorizáciu stlačí tlačidlo (zvonček), a zariadenie s nainštalovanou aplikáciou následne notifikuje používateľa o tejto skutočnosti. Následne je po vzhliadnutí prenosu z priezoru možné postupovať podľa vlastného uváženia. Ak sa v obraze nachádza neznáma osoba po stanovenej chvíli dôjde k nahraniu fotky tejto osoby, ako aj videa (nahrávanie ktorého bolo spustené detekciou pohybu) do cloudovej služby.

Je na konečnom používateľovi, aby vhodne navrhol čas nahrávania medzi jednotlivými fotkami, aby neprichádzalo k početnému nahrávaniu jednej a tej istej neznámej osoby, prípadne, v akej situácií má byť nahrávanie pozastavené, znovu spustené, vymazané atď. Ohľad musí byť braný aj na to, či sa daná známa alebo neznáma osoba bude v zornom poly kamery nachádzať dlhší čas (príchod po chodbe), či sa jedná o umiestnenie v miestach, kde sa daná osoba presunie iba z dôvodu autorizácie. Ďalej sa naskytuje otázka,

či je potrebné verifikovať každú osobu v obraze, prípadne je postačujúca verifikácia iba jednej osoby k autorizácií (verzia použitá v tejto práci).

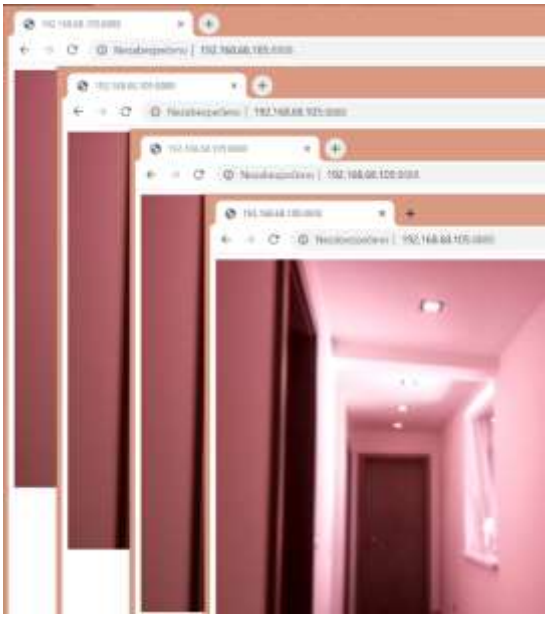


Obrázok 49: Porovnanie HAAR, HOG a CNN pri detekcií tváre v obraze. Ako vidíme na obrázku, metóda HAAR pri danom natočení tvár nedeteguje. Taktiež môžeme vidieť ako metóda CNN značne orezáva detegovanú tvár a vynecháva oblasť čela a brady.

Sprístupnenie videa online prebieha spoľahlivo a bez problémov dosahujúc pri súčasnej implementácií rýchlosť okolo ~10 FPS. Bezpečné narábanie s nahrávaným obrazom umožňuje sledovanie na viacerých zariadeniach súčasne (Obrázok 50).

Pre dosiahnutie čo najlepších výsledkov musí byť braný ohľad na dostatočné prispôsobenie zariadenia všetkým podmienkam, v ktorých bude používané, a ktoré majú vplyv na kvalitu výstupu. Počas niekoľkodňového testovania zariadenia a následnej kontroly neboli zistené problémy pri autentifikácii ani falošne pozitívne detekcie. Jednalo sa však o testovanie vo vnútorných priestoroch s dostatočným osvetlením a v domácnosti s menším počtom osôb.

V snahe o vylepšenie projektu, je možné použitie kamery s lepšími sledovacími uhlami, ktorá dokáže pracovať aj v zhoršených svetelných podmienkach, a tým pádom poskytuje kvalitnejší vstup pre rozpoznávanie. Možným pokračovaním v projekte by bolo posunutie úrovne zabezpečenia na vyššiu úroveň, použitím napríklad projektoru siete infračervených bodov spolu so snímačom hĺbky, čím by sa úplne znemožnilo použitiu 2D obrazu pri autentifikácií.



Obrázok 50: Server bez problémov zvláda niekoľko klientov zároveň. Nevýhodou je však čiastočné oneskorenie, ktoré nastáva pri príliš veľkom počte klientov pristupujúcich k streamovanému videu, čo však nie je prípad tohto projektu.

Dosiahnuté ciele a možné pokračovanie

Výstupom tejto diplomovej práce je funkčný prototyp inteligentného priezoru. Cieľom bolo demonštrovať možnosť využitia Raspberry Pi na vykonávanie funkcií požadovaných v zadaní, ako je spracovávanie obrazu, detekcia tváre, záznam a vzdialená správa zariadenia. Je možné konštatovať, že výsledný prototyp spĺňa všetky ciele diplomovej práce.

Z hľadiska bežného použitia inteligentného priezoru, je však výsledný prototyp ukrátený o niektoré funkcie, ktoré bývajú súčasťou iných bežne používaných väčšinou neinteligentných video vrátnikov (priezorov). Jednou z prvých je absencia pripojenia na fyzické dvere, ktoré by zariadenie v bežnom prípade mohlo otvoriť. Zariadenie, ktorému je umožnená takáto funkcia musí poskytovať dostatočnú úroveň zabezpečenia, keďže sa jedná o povolenie prístupu do domu, miestnosti, budovy. Tento systém, vzhľadom na použitie rozpoznávania tváre, ako formy autorizácie, poskytuje iba minimálnu úroveň zabezpečenia, a preto nie je vhodné, aby bolo takémuto systému umožnené povoľovať prístup osôb do akéhokoľvek objektu. Detaily k bezpečnosti použitia rozpoznávania tváre sú poskytnuté nižšie.

Realizácia výstupu vo forme zaznamenaných dát, svetelného signálu, zvukového a obrazového výstupu, bola vhodnejšou alternatívou a taktiež umožňuje použitie zariadenia takpovediac „out of the box“ (voľný preklad priamo z krabice), bez nutnosti pripájať ho na fyzické dvere. Zariadenie môže plne slúžiť ako inteligentný dochádzkový systém, záznamník pohybov osôb, sledovanie priestorov, detekcia a nahrávanie pohybu, bezpečnostný kamerový systém, atď. Ak by aj napriek bezpečnostným rizikám finálna verzia bola realizovaná prostredníctvom formálneho pripojenie motorčeka na otvorenie zámku, prípadne elektronického zámku dverí, tak by bez koncového pripojenia na dvere použitie a využitie zariadenia nebolo možné. V domácom prostredí nebola možnosť prispôsobenia zámkov na elektronické použitie, z toho dôvodu takáto možnosť nebola testovaná. V prípade potreby je totiž realizáciu možné vykonať v relatívne krátkom čase, úpravou použitia výstupných kolíkov na Raspberry na ktoré sú v súčasnosti pripojené diódy, tlačidlo a bzučiak.

Úprava zariadenia na reálne používanie by si spravidla vyžadovalo použitie záložných batérii pre prípad výpadku elektrického prúdu a v neposlednom rade možnosť sofistikovaného vypnutia/zapnutia respektíve resetovania. Tieto dodatočné a pri reálnom zariadení často potrebné funkcie neboli realizované z dôvodu malej pridanej hodnoty k výslednému návrhu.

Rozpoznávanie tváre aj napriek širokej škále použitia, jednoznačne nepatrí k najbezpečnejším biometrickým metódam. Mnohé systémy rozpoznávania tvárí renomovaných svetových výrobcov inteligentných telefónov boli prelomené iba použitím obyčajnej fotografie (Navarro, 2019). Dokonca aj oveľa sofistikovanejšie systémy

rozpoznávania tváre, vyžívajúce hĺbkové dáta aj dômyselné sledovanie regiónu očí, je v súčasnosti možné oklamať s použitím väčšej či menšej námahy (Winder, 2019) (Kumar, 2017).

Všeobecne by sa dalo povedať, že jednou zo slabín, ktorá vzniká pri použití biometrickej metódy ako rozpoznávanie tváre (ale aj iných biometrických metód), pri autentifikácií je často neschopnosť zistiť či biometrické údaje prezentované v danom momente sú prezentované osobou, ktorej skutočne patria tzv. Liveness Detection (voľný preklad detekcia živosti), prípadne či je táto osoba pri vedomí (napr. spiaca osoba). Situáciu, keď neživý objekt vykazuje ľudské vlastnosti a tento je prezentovaný pred kamerou, alebo biometrickým senzorom nazývame tzv. spoofing (voľný preklad ťahanie za nos). Tento problém vzniká najmä technologickým pokrokom, ktorého použitie sa bohužiaľ nevyhýba ani nelegálnej sfére.

Algoritmy detekcie tváre využité v tejto diplomovej práci za normálnych okolností nie sú schopné rozoznávať, či sa jedná o skutočnú osobu alebo fotografiu. Za týmto účelom a ako snaha aspoň čiastočne znížiť možnosť tzv. spoofingu, bol popri bežnej detekcii implementovaný systém sledovania veľkosti otvorenia očí. Na fotkách je táto veľkosť nemenná a väčšinou sa tak javí aj v prípade mierneho pohybu s fotografiou. Samotná autentifikácie je teda podmienená, okrem zhody s databázou známych osôb, aj dostatočnou zmenou veľkosti otvorenia oka, ku ktorej môže dôjsť pri snímaní občas aj automaticky, v opačnom prípade je potrebné aby osoba žmurkla. Tento systém nepracuje v žiadnom prípade bezchybne (pri snahe je možné autorizovať osobu aj pomocou fotky), avšak vďaka tejto implementácií môžeme autorizáciu pomocou fotky, ktorá by inak prebehla bez problémov, prinajmenšom skomplikovať.

Výsledná Android aplikácia a jej funkcie sú podrobne popísané v predchádzajúcej kapitole. Aplikácia funguje s popísanými funkciami bez problémov v rámci dosahu siete Wifi, na ktorú sú pripojené aj ostatné využívané zariadenia. Žiaducou funkciou pri reálnom nasadení môže byť sprístupnenie živého prenosu z kamery, pri pripojení na internet aj mimo domácej siete. Takéto riešenie si vyžaduje využitie statickej verejnej IP adresy. Táto verzia nebola implementovaná z dôvodu, že iba minimum používateľov internetu disponuje takouto adresou. Táto služba je navyše dodatočne spoplatnená, v horšom prípade nie je poskytovateľom internetového pripojenia ani poskytovaná. Statická verejná IP adresa by umožnila pristupovať k obrazu z Raspberry Pi (priezoru), z ktorejkoľvek nie len domácej siete.

Alternatívne sa na tento účel dá využiť služba ako napríklad Dynamic DNS, ktorá udržuje na svojich serveroch aktuálne hodnoty dynamických IP adries na základe identifikácie zariadenia pomocou zvoleného používateľského názvu, bez potreby statickej verejnej IP adresy. Aj táto služba je však spoplatnená. V oboch prípadoch je navyše následne potrebné nastaviť presmerovania portov na našom domácom routery tak, aby požiadavka aplikácie na prístup k živému videu z Raspberry Pi (inteligentného priezoru) bola správne preložená a prístup bol umožnený.

Dôvodov pre nevyužitie týchto možností je viacero. Jednou je už spomínané spoplatnenie daných služieb a taktiež potreba znalosti konfigurácie domácej siete, ktorá môže byť pre koncového užívateľa príliš náročná. V neposlednom rade je to aj nutnosť adekvátneho zabezpečenia takéhoto prístupu vzhľadom na prísne predpisy v oblasti ochrany dát, ktorým takéto obrazové dáta podliehajú. Zabezpečenie takéhoto prenosu a využitie vyššie uvedených možností môže byť predmetom budúceho skúmania.

Aplikácia, ako taká, ale umožňuje aj dosah mimo domácej siete, ktorý je zabezpečený využitím funkcií FireBase, vďaka čomu užívateľ obdrží informáciu o stlačení tlačidla na inteligentnom priezore (formou notifikácie pri akomkoľvek pripojení na internet). Získa tak príležitosť, v rámci možností reagovať na situáciu, aj keď sa nachádza mimo dosahu domácej siete.

Záver

Počiatočným cieľom tejto diplomovej práce bolo oboznámenie sa s platformou Raspberry Pi. Boli popísané základné technické parametre rôznych verzií, najpoužívanejšie softwarové vybavenie a alternatívy nachádzajúce sa na trhu.

V rámci softwarových možností, podpory zo strany výrobcu a komunity je Raspberry Pi vhodným kandidátom na veľké množstvo projektov. Práca je uľahčená množstvom dostupnej literatúry, podporou zo strany komunity, počtom dostupných open-source projektov a softwaru, pravidelných publikácií. Vďaka podpore širokého spektra bežne používaných programovacích jazykov a celkovo dobrým možnostiam kompatibility, je práca na tejto platforme umožnená širokému spektru možných používateľov.

Z analýzy vyplýva jednoznačné prvenstvo tejto platformy v segmente malých jednodoskových počítačov a Raspberry Pi v pomere cena/výkon na trhu iba ťažko hľadá premožiteľa.

Za hlavnú nevýhodu spojenú s touto platformou sa dá považovať obmedzený výpočtový výkon, ktorý významne zužuje možnosti výberu softwarových produktov a rozsahu ich použitia. Pri zvažovaní využitia Raspberry Pi pre tvorbu profesionálnych riešení (produktovexistuu aj ďalšie faktory, ktoré je potrebné brať do úvahy: beh operačného systému z SD karty, neprítomnosť Real-time Clock obvodu, napájanie pomocou micro USB, prehrievanie, škálovateľnosť, atď. Za jednu z nevýhod sa dá označiť aj rýchly vývoj nového softwaru, ktorý pri množstve neoficiálnych programov, knižníc a doplnkov, môže často spôsobiť nemalé problémy pri snahe o aktualizáciu, upgrade, úpravu existujúcich a tvorbu nových projektov.

Na základe vykonanej analýzy bola navrhnutá praktická realizácia predloženého konceptu. Po návrhu, ktorý v sebe zahrňuje aj rozpoznávanie tváre, nasleduje potrebné uvedenie do tejto problematiky s dôrazom na metódy použiteľné na zariadení so značne obmedzeným výkonom akým je Raspberry Pi.

Praktická časť predstavuje podrobný návod (manuál) práce na Raspberry Pi pre užívateľa, ktorý nemá žiadne alebo minimálne skúsenosti s touto platformou.

Ďalej práca poskytuje podrobný postup inštalácie jednotlivých knižníc použitých pri zázname a spracovaní obrazu, odosielaní videa na druhý počítač alebo sprístupnenia video online. Dôraz bol kladení na výber moderných, ľahko škálovateľných a perspektívnych riešení ktoré môžu byť výhodné pri ich budúcom použití a tento návod tak uľahčí prácu s týmto softwarom a knižnicami menej skúseným užívateľom.

V časti implementácia sú bližšie popísané dôležité časti zdrojového kódu, slúžiace k implementácií používaných knižníc a softwaru (vytvorenie databázy, detekcia tváre, detekcia pohybu, sledovanie zmeny otvorenia očí, streamovanie videa, ...). Snahou je poskytnúť čitateľovi ucelený pohľad na kroky, ktoré je potrebné vykonať pri konkrétnych

úkonoch. Opísané časti tak môžu slúžiť ako východiskový bod danej problematiky pre ďalšiu činnosť.

Vytvorený inteligentný priezor umožňuje autentifikáciu osôb ako na základe ich tváre, tak aj na základe hesla v grafickej forme. Využité je aj sledovanie zmeny otvorenia očí čo sťažuje „klamanie systému“ použitím 2D fotografií a detekcia pohybu. Kvalita rozpoznávania vykazuje za ideálnych podmienok vysokú presnosť vďaka pokročilosti použitých metód, implementovaných na počítači spracovávajúcom obraz z Raspberry Pi. Pri spracovávaní obrazu priamo na Raspberry Pi je rýchlosť spracovávania značne nižšia.

Jedným z cieľov diplomovej práce bolo zabezpečiť vzdialenú správu zariadenia. Tú umožňuje Android aplikácia. Ukladanie obrazových a textových dát je zabezpečené počítačom spracovávajúcim obraz, ktorý tieto dáta automaticky nahráva do cloudovej služby.

Výsledný prototyp zariadenia je možné použiť ako formu inteligentného systému kontroly dochádzky, záznamu vstupov, bezpečnostného kamerového systému so záznamom pohybu a tvárí osôb.

Na záver je možné konštatovať, že Raspberry Pi umožňuje vytvorenie funkčného systému na rozpoznávanie tváre, bez nutnosti použitia dodatočného hardwaru, avšak s obmedzeniami. Práca prezentuje aj prístup ako vytvoriť takýto systém využitím výkonnejšieho počítača v spojení s Raspberry Pi. Existujú aj ďalšie prístupy zvyšujúce výkon samotného Raspberry Pi, napríklad využitie USB akcelerátorov (Coral, 2020). Takýto prístup môže byť predmetom ďalšieho skúmania.

Bibliografická citácia

ABATE, A. F. et al. 2D and 3D face recognition: A survey. Pattern Recognition Letters, v. 28, n. 14, p. 1885-1906, Oct 2007. ISSN 0167-8655. Dostupné z: < <Go to ISI>://WOS:000249543900005 >.

AQUEEL, A. Introduction to Raspberry Pi 3 B+. 2018. Dostupné z: < https://www.theengineeringprojects.com/2018/07/introduction-to-raspberry-pi-3-b-plus.html >. Cit.: 28.03.2019.

BASS, J. imagezmq: Transporting OpenCV images. 2019. Dostupné z: < https://github.com/jeffbass/imagezmq >. Cit.: 04.05.2019.

BIRCHARD, T. The Art of Routing in Flask. 2019. Dostupné z: < https://hackersandslackers.com/flask-routes/ >. Cit.: 02.02.2020.

CELLAN-JONES, R. A 15 pound computer to inspire young programmers. 2011. Dostupné z: < http://www.bbc.co.uk/blogs/thereporters/rorycellanjones/2011/05/a_15_computer_to_inspire_young.html >.

CMAKE. 2019. Dostupné z: < https://cmake.org/ >. Cit.: 10.05.2019.

CORAL. Get started with the USB Accelerator. 2020. Dostupné z: < https://coral.ai/docs/accelerator/get-started/#requirements >. Cit.: 04.04.2020.

CORNEJO-BUENO, L. et al. Bayesian optimization of a hybrid system for robust ocean wave features prediction. Neurocomputing, v. 275, p. 818-828, Jan 2018. ISSN 0925-2312. Dostupné z: < <Go to ISI>://WOS:000418370200078 >.

CORRAL-GARCIA, J. et al. Analysis of Energy Consumption and Optimization Techniques for Writing Energy-Efficient Code. Electronics, v. 8, n. 10, p. 21, Oct 2019. Dostupné z: < <Go to ISI>://WOS:000498262700132 >.

CUDA. Build and Install dlib with CUDA support on Ubuntu. 2018. Dostupné z: < http://leadtosilverlining.blogspot.com/2018/08/build-and-install-dlib-with-cuda.html >. Cit.: 20.05.2019.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. Conference on Computer Vision and Pattern Recognition, 2005, San Diego, CA. Ieee Computer Soc, Jun 20-25. p.886-893.

DING, L. Y.; MARTINEZ, A. M.; IEEE. Precise detailed detection of faces and facial features. IEEE Conference on Computer Vision and Pattern Recognition, 2008, Anchorage, AK. Ieee, Jun 23-28. p.3653-3659.

DOCS. Virtual Environments and Packages. 2019. Dostupné z: < https://docs.python.org/3/tutorial/venv.html >. Cit.: 23.1.2019.

DONG, J. et al. Learning to Align Images using Weak Geometric Supervision. 6th International Conference on 3D Vision (3DV), 2018, Verona, ITALY. Ieee, Sep 05-08. p.700-709.

DROPBOX. Getting started with the Dropbox API. 2020. Dostupné z: < https://www.dropbox.com/developers/reference/getting-started >. Cit.: 01.02.2020.

FERNANDEZ, M. C. D. et al. Simultaneous Face Detection and Recognition using Viola-Jones Algorithm and Artificial Neural Networks for Identity Verification. 2014 Ieee Region 10 Symposium, p. 672-676, 2014. Dostupné z: < <Go to ISI>://WOS:000393381300131 >.

FIREBASE. Firebase helpsmobile and webapp teams succeed. 2020. Dostupné z: < https://firebase.google.com/?hl=sk >. Cit.: 10.09.2020.

FITZER, A. The History of the Raspberry Pi. 2018. Dostupné z: < https://hitaltech.co.uk/the-history-of-the-raspberry-pi/ >.

Flask. 2019. Dostupné z: < https://palletsprojects.com/p/flask/ >. Cit.: 09.09.2019.

GEITGEY, A. Face Recognition. 2017. Dostupné z: < https://face-recognition.readthedocs.io/en/latest/face_recognition.html >. Cit.: 02.02.2019.

GIBS, S. Raspberry Pi becomes best selling British computer. 2015. Dostupné z: < https://www.theguardian.com/technology/2015/feb/18/raspberry-pi-becomes-best-selling-british-computer >.

GODBEHERE, A. B. et al. Visual Tracking of Human Visitors under Variable-Lighting Conditions for a Responsive Audio Art Installation. American Control Conference (ACC), 2012, Montreal, CANADA. Ieee Computer Soc, Jun 27-29. p.4305-4312.

GUENNOUNI, S.; AHAITOUF, A.; MANSOURI, A. Face Detection: Comparing Haar-like combined with Cascade Classifiers and Edge Orientation Matching. New York: Ieee, 2017. ISBN 978-1-5090-6681-0. Dostupné z: < <Go to ISI>://WOS:000403404900012 >.

GUO, B. S. et al. The Tabu_Genetic Algorithm: A Novel Method for Hyper-Parameter Optimization of Learning Algorithms. Electronics, v. 8, n. 5, p. 19, May 2019. ISSN 2079-9292. Dostupné z: < <Go to ISI>://WOS:000470999900110 >.

GUS. Raspberry Pi Camera vs NOIR Camera. 2019. Dostupné z: < https://pimylifeup.com/raspberry-pi-camera-vs-noir-camera/ >. Cit.: 30.03.2019.

GUTIERREZ, R. J. Arduino vs Raspberry Pi: How to Choose the Right Board. 2019. Dostupné z: < https://all3dp.com/1/arduino-vs-raspberry-pi/ >.

HAO, J. G.; HO, T. K. Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language. Journal of Educational and Behavioral Statistics, v. 44, n. 3, p. 348-361, Jun 2019. ISSN 1076-9986. Dostupné z: < <Go to ISI>://WOS:000468930900005 >.

HASSNER, T.; IEEE. Viewing Real-World Faces in 3D. IEEE International Conference on Computer Vision (ICCV), 2013, Sydney, AUSTRALIA. Ieee, Dec 01-08. p.3607-3614.

HATTERSLEY, L. Raspberry Pi 4 vs Raspberry Pi 3B+. 2019. Dostupné z: < https://magpi.raspberrypi.org/articles/raspberry-pi-4-vs-raspberry-pi-3b-plus >. Cit.: 10.01.2020.

HE, K. M. et al. Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, Seattle, WA. Ieee, Jun 27-30. p.770-778.

HEATH, N. Choosing a Raspberry Pi OS? Here's the definitive list. 2018. Dostupné z: < https://www.techrepublic.com/pictures/choosing-a-raspberry-pi-os-heres-the-definitive-list/ >. Cit.: 10.03.2019.

HINTJENS, P. ZeroMQ. 2020. Dostupné z: < https://zeromq.org/get-started/ >. Cit.: 06.06.2020.

HOUSSAINI, A. S. et al. Real-Time Driver's Hypovigilance Detection using Facial Landmarks. 2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (Wits), p. 4, 2019. Dostupné z: < <Go to ISI>://WOS:000475695900069 >.

HUANG, G. Labeled Faces in the Wild. 2019. Dostupné z: < http://vis-www.cs.umass.edu/lfw/ >.

HUANG, G. B. et al. Unsupervised joint alignment of complex images. 11th IEEE International Conference on Computer Vision, 2007, Rio de Janeiro, BRAZIL. Ieee, Oct 14-21. p.237-244.

JARVAN. Differences between Raspberry Pi B, B+, A+, 2B, 3B. 2017. Dostupné z: < https://www.elecrow.com/blog/differences-between-raspberry-pi-b-b-a-2b-3b/ >.

Kebidu. 2019. Dostupné z: < https://www.aliexpress.com/item/kebidu-Portable-Mini-Monaural-Speaker-Music-Stereo-Audio-Speakers-For-Smart-Moblie-Phone-MP4-With-3/32920365765.html?spm=a2g0s.9042311.0.0.27424c4dBYz4rh >. Cit.: 30.03.2019.

KHAMMARI, M. Robust face anti-spoofing using CNN with LBP and WLD. Iet Image Processing, v. 13, n. 11, p. 1880-1884, Sep 2019. ISSN 1751-9659. Dostupné z: < <Go to ISI>://WOS:000487789000008 >.

KHAZRI, A. Face Detection models and softwares. 2019. Dostupné z: < https://becominghuman.ai/face-detection-models-and-softwares-42b562a8e151 >. Cit.: 1.5.2019.

KIMARI, K. 5 of the Best Raspberry Pi Alternatives in 2019. 2019. Dostupné z: < https://www.maketecheasier.com/raspberry-pi-alternatives/ >.

KING, D. High Quality Face Recognition with Deep Metric Learning. 2017. Dostupné z: < http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html >. Cit.: 26.4.2019.

KING, D. Dlib C++ Library. 2019a. Dostupné z: < http://dlib.net/ >. Cit.: 09 .09.2019.

KING, D. Using dlib from Python. 2019b. Dostupné z: < http://dlib.net/compile.html >. Cit.: 15.05.2019.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. Communications of the Acm, v. 60, n. 6, p. 84-90, Jun 2017. ISSN 0001-0782. Dostupné z: < <Go to ISI>://WOS:000402555400026 >.

KUMAR, A.; KAUR, A.; KUMAR, M. Face detection techniques: a review. Artificial Intelligence Review, v. 52, n. 2, p. 927-948, Aug 2019. ISSN 0269-2821. Dostupné z: < <Go to ISI>://WOS:000472573300006 >.

KUMAR, M. Apple iPhone X's Face ID Hacked (Unlocked) Using 3D-Printed Mask. 2017. Dostupné z: < https://thehackernews.com/2017/11/iphone-face-id-unlock-hack.html >. Cit.: 09.09.2019.

LAM, L. K. FACE RECOGNITION AND EMOTION DETECTION USING PCA AND VIOLA-JONES ALGORITHMS. 11th International Conference on Technology, Education and Development (INTED), 2017, Valencia, SPAIN. Iated-Int Assoc Technology Education & Development, Mar 06-08. p.5225-5230.

LI, B.; HUO, G. Face recognition using locality sensitive histograms of oriented gradients. Optik, v. 127, n. 6, p. 3489-3494, 2016. ISSN 0030-4026. Dostupné z: < <Go to ISI>://WOS:000369456500059 >.

LI, S. Z. Encyclopedia of Biometrics. Springer, Boston, MA, 2009. 1466 ISBN 978-0-387-73002-8.

LOCKER, A. New Raspberry Pi 3 B+: Faster Chipset, Better Network Features, Same Price. 2018. Dostupné z: < https://all3dp.com/new-raspberry-pi-3-b-faster-chipset-better-network-features-price/ >. Cit.: 28.03.2019.

LOCKER, A. Best Single Board Computers 2019 (Raspberry Pi Alternatives). 2019. Dostupné z: < https://all3dp.com/1/single-board-computer-raspberry-pi-alternative/ >.

LONG, M. Best Raspberry Pi Alternatives 2019. 2019. Dostupné z: < https://www.electromaker.io/blog/article/best-raspberry-pi-alternatives-2019 >.

LUO, G. A review of automatic selection methods for machine learning algorithms and hyper-parameter values. Network Modeling and Analysis in Health Informatics and Bioinformatics, v. 5, n. 1, p. 16, Dec 2016. ISSN 2192-6670. Dostupné z: < <Go to ISI>://WOS:000447430700018 >.

LYONS, C. A History Of The Raspberry Pi. 2015. Dostupné z: < http://novadigitalmedia.com/history-raspberry-pi/ >.

MAGPI. SSH: REMOTE CONTROL YOUR RASPBERRY PI. 2017. Dostupné z: < https://www.raspberrypi.org/magpi/ssh-remote-control-raspberry-pi/ >. Cit.: 10.04.2019.

MAHESHWARI, M. Understanding SSH workflow. 2017. Dostupné z: < https://medium.com/@Magical_Mudit/understanding-ssh-workflow-66a0e8d4bf65 >. Cit.: 10.04.2019.

MANTORO, T.; AYU, M. A.; SUHENDI. Multi-Faces Recognition Process Using Haar Cascades and Eigenface Methods. 6th International Conference on Multimedia Computing and Systems (ICMCS), 2018, Rabat, MOROCCO. Ieee, May 10-12. p.24-28.

MARCEL, S. Lab: Face Verification / Face Authentication. 2006. Dostupné z: < https://www.idiap.ch/~marcel/labs/faceverif.php >. Cit.: 19.1.2019.

MEYERS, E.; WOLF, L. Using biologically inspired features for face processing. International Journal of Computer Vision, v. 76, n. 1, p. 93-104, Jan 2008. ISSN 0920-5691. Dostupné z: < <Go to ISI>://WOS:000252185800006 >.

MILLER, P. Raspberry Pi sold over 12.5 million boards in five years. 2017. Dostupné z: < https://www.theverge.com/circuitbreaker/2017/3/17/14962170/raspberry-pi-sales-12-5-million-five-years-beats-commodore-64 >.

MOE, L. Best Raspberry Pi Alternatives. 2019. Dostupné z: < https://www.electromaker.io/blog/article/10-best-raspberry-pi-alternatives >.

NAVARRO, F. These phones can be unlocked with nothing more than a photograph. 2019. Dostupné z: < https://www.komando.com/security-privacy/these-phones-can-be-unlocked-with-nothing-more-than-a-photograph/529870/ >. Cit.: 09.09.2019.

NUTTALL, B. Top 10 Raspberry Pi add-on boards. 2016. Dostupné z: < https://opensource.com/life/16/7/top-10-Raspberry-Pi-boards >. Cit.: 30.03.2019.

OpenCV. 2019. Dostupné z: < https://opencv.org/about.html >. Cit.: 30.03.2019.

OSOWSKI, S.; SIWEK, K. CNN application in face recognition. Przeglad Elektrotechniczny, v. 96, n. 3, p. 142-145, 2020. ISSN 0033-2097. Dostupné z: < <Go to ISI>://WOS:000519193200031 >.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, v. 12, p. 2825-2830, Oct 2011. ISSN 1532-4435. Dostupné z: < <Go to ISI>://WOS:000298103200003 >.

PETERSON, T. Raspberry Pi just turned 5. Here's a brief history of the world's tiniest hobbyist computer., 2017. Dostupné z: < https://medium.freecodecamp.org/raspberry-pi-just-turned-5-d4210cc29230 >.

PETLOVANA, Y. Top 13 Python Web Frameworks to Learn in 2020. 2020. Dostupné z: < https://steelkiwi.com/blog/top-10-python-web-frameworks-to-learn/ >. Cit.: 10.1.2020.

PFAFFLY, S. How to change Raspberry Pi's Swapfile Size on Raspbian. 2015. Dostupné z: < https://www.bitpi.co/2015/02/11/how-to-change-raspberry-pis-swapfile-size-on-rasbian/ >. Cit.: 23.1.2019.

Python. 2019. Dostupné z: < https://www.python.org/about/ >. Cit.: 30.03.2019.

PYZMQ. PyZMQ Documentation. 2020. Dostupné z: < https://pyzmq.readthedocs.io/en/latest/index.html >. Cit.: 06.06.2019.

RASPBERRYPI. Camera Module. 2016. Dostupné z: < https://www.raspberrypi.org/documentation/hardware/camera/ >. Cit.: 30.03.2019.

ROUSE, M. virtual network computing (VNC). 2006. Dostupné z: < https://searchnetworking.techtarget.com/definition/virtual-network-computing >. Cit.: 10.04.2019.

RUSH, C. Getting Started with RTk Traffic HAT. 2018. Dostupné z: < https://learn.pi-supply.com/make/getting-started-with-rtk-traffic-hat/#getting-started-with-rtk-traffic-hat >. Cit.: 30.03.2019.

SCHROFF, F. et al. FaceNet: A Unified Embedding for Face Recognition and Clustering. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, Boston, MA. Ieee, Jun 07-12. p.815-823.

SHARIFARA, A. et al. A General Review of Human Face Detection Including a Study of Neural Networks and Haar Feature-based Cascade Classifier in Face Detection. New York: Ieee, 2014. 73-78 ISBN 978-1-4799-6444-4. Dostupné z: < <Go to ISI>://WOS:000380456400014 >.

SINGH, A.; THAKUR, N.; SHARMA, A. A Review of Supervised Machine Learning Algorithms. Proceedings of the 10th Indiacom - 2016 3rd International Conference on Computing for Sustainable Global Development, p. 1310-1315, 2016. Dostupné z: < <Go to ISI>://WOS:000388117501079 >.

TAIGMAN, Y. et al. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. 27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, Columbus, OH. Ieee, Jun 23-28. p.1701-1708.

TATHAM, S. PuTTY: a free SSH and Telnet client. 2019. Dostupné z: < https://www.chiark.greenend.org.uk/~sgtatham/putty/ >. Cit.: 10.04.2019.

THIEDE, L. A.; PARLITZ, U. Gradient based hyperparameter optimization in Echo State Networks. Neural Networks, v. 115, p. 23-29, Jul 2019. ISSN 0893-6080. Dostupné z: < <Go to ISI>://WOS:000468877100003 >.

TUNG, L. Raspberry Pi has now sold 30 million tiny single-board computers. 2019. Dostupné z: < https://www.zdnet.com/article/raspberry-pi-now-weve-sold-30-million/ >. Cit.: 10.01.2020.

UBUNTU. What is Ubuntu? , 2020. Dostupné z: < https://help.ubuntu.com/lts/installation-guide/s390x/ch01s01.html >. Cit.: 02.02.2020.

UPTON, E. Raspberry Pi 3 Model B+. 2018. Dostupné z: < https://www.raspberrypi.org/blog/raspberry-pi-3-model-bplus-sale-now-35/ >.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. Conference on Computer Vision and Pattern Recognition, 2001, Kauai, Hi. Ieee Computer Soc, Dec 08-14. p.511-518.

WANG, Z. et al. Automatic Hyperparameter Tuning of Machine Learning Models under Time Constraints. IEEE International Conference on Big Data (Big Data), 2018, Seattle, WA. Ieee, Dec 10-13. p.4967-4973.

WIKIPEDIA. Raspberry Pi 3b+. 2019. Dostupné z: < https://en.wikipedia.org/wiki/Raspberry_Pi >. Cit.: 28.2.2019.

WINDER, D. Apple's iPhone FaceID Hacked In Less Than 120 Seconds. 2019. Dostupné z: < https://www.forbes.com/sites/daveywinder/2019/08/10/apples-iphone-faceid-hacked-in-less-than-120-seconds/#58d8f8a521bc >. Cit.: 09.09.2019.

XU, P. Y.; IOP. Review on Studies of Machine Learning Algorithms. International Symposium on Power Electronics and Control Engineering (ISPECE), 2018, Xian Univ Technol, Xian, PEOPLES R CHINA. Iop Publishing Ltd, Dec 28-30.

YANG, H. C.; WANG, X. A. Cascade face detection based on Histograms of Oriented Gradients and Support vector machine. New York: Ieee, 2015. 766-770 ISBN 978-1-4673-9473-4. Dostupné z: < <Go to ISI>://WOS:000380398500131 >.

YAVUZ, H. S.; CEVIKALP, H.; EDIZKAN, R. A comprehensive comparison of features and embedding methods for face recognition. Turkish Journal of Electrical Engineering and Computer Sciences, v. 24, n. 1, p. 313-340, 2016. ISSN 1300-0632. Dostupné z: < <Go to ISI>://WOS:000368559000019 >.

YI, D. et al. Towards Pose Robust Face Recognition. 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, Portland, OR. Ieee, Jun 23-28. p.3539-3545.

ZHAO, W. et al. Face recognition: A literature survey. Acm Computing Surveys, v. 35, n. 4, p. 399-459, Dec 2003. ISSN 0360-0300. Dostupné z: < <Go to ISI>://WOS:000187181700003 >.

ZHU, Z. The Triplet loss in cosine similarity., 2017. Dostupné z: < https://www.researchgate.net/figure/The-Triplet-loss-in-cosine-similarity_fig3_316736728 >. Cit.: 5.5.2019.

Raspbian. 2018. Dostupné z: < https://www.raspbian.org/RaspbianAbout >. Cit.: 5.3.2020.

Zoznam skratiek, symbolov, obrázkov a tabuliek

Zoznam skratiek

ABA	Appearance Based Algorithm
BGR	Blue Green Red
BLE	Bluetooth Low Energy
CPU	Central Processing Unit
CSI	Camera Serial Interface
EEPROM	Electrically Erasable Programmable Read-Only Memory
GPIO	General Purpose Input/output
GPU	Graphics Processing Unit
HAT	Hardware Attached on Top
IP	Internet Protocol
PoE	Power over Ethernet
QR	Quick Response
RGB	Red Green Blue
SSH	Secure Shell

Zoznam obrázkov

Obrázok 1: Logo Raspberry Pi (Wikipedia, 2019). 17
Obrázok 2: Model Raspberry Pi 3 B+ (Wikipedia, 2019). 17
Obrázok 3: Raspberry Pi 1 A+ (vľavo) a model B+ (vpravo)(Jarvan, 2017). 18
Obrázok 4: Raspberry Pi Zero (Peterson, 2017). 19
Obrázok 5: Raspberry Pi 4 (Hattersley, 2019). 19
Obrázok 6: Arduino Mega 2560 (Locker, 2019). 22
Obrázok 7: Rock Pi Model B (Locker, 2019). 23
Obrázok 8: Banana Pi M64 (Long, 2019). 24
Obrázok 9: Asus Tinker Board S (Locker, 2019). 25
Obrázok 10: Odroid-XU4 (Long, 2019). 26
Obrázok 11: Popis modelu Raspberry Pi 3 B+(Aqueel, 2018). 28
Obrázok 12: „Pin-out“ pre model 3B+(Aqueel, 2018). 29
Obrázok 13: Kamerový modul Pi V2 NoIR (Raspberrypi, 2016). 30
Obrázok 14: Prenosný reproduktor (Kebidu, 2019). 30
Obrázok 15: Traffic HAT (Nuttall, 2016). 31
Obrázok 16: Logo Python (Python, 2019). 32
Obrázok 17: Logo OpenCV (OpenCV, 2019). 32
Obrázok 18: Logo Dlib (King, 2019a). 33
Obrázok 19: Logo Flask (Flask, 2019). 33
Obrázok 20: Logo ZeroMQ (Hintjens, 2020). 34
Obrázok 21: Logo Firebase (Firebase, 2020). 35
Obrázok 22: Ukážka výstupu systému na detekciu tváre (Khazri, 2019). 38
Obrázok 23: Proces zrovnávania tváre. 39
Obrázok 24: Triplet Loss funkcia (Zhu, 2017). 41
Obrázok 25: Ukážka klasifikácie 42
Obrázok 26: Ukážka NOOBS na oficiálnej stránke www.raspberrypi.org. 45
Obrázok 27: Zdieľania internetového pripojenia. 46
Obrázok 28: Zvýraznený riadok s modrou ikonou odkazuje na IP adresu Raspberry Pi. 47
Obrázok 29: Užívateľské rozhranie PuTTY. 47
Obrázok 30: Prihlásenie. 48
Obrázok 31: Pracovná plocha Raspberry Pi v prostredí VNC Viewer. 49
Obrázok 32: Ukážka rezervovanej adresy pre Raspberry Pi. 55
Obrázok 33: Osobné zložky jednotlivých osôb.. 57
Obrázok 34: Reprezentácia tváre vo forme 128-bytového vektoru. 58
Obrázok 35: Výstup detekcie tváre metódou HOG.. 61
Obrázok 36: Zobrazenie kľúčových bodov tváre (Houssaini et al., 2019). 61
Obrázok 37: Umiestnenie kľúčových bodov na oblasť tváre v reálnom čase. 62
Obrázok 38: Proces sledovania oblasti očí. 64
Obrázok 39: Obrázok z detekcie QR kódu. 65
Obrázok 40: Ukážka z detekcie pohybu v reálnom čase. 67
Obrázok 41: Súbory nahrané do cloudovej služby 69
Obrázok 42 : Zoznam povolených prístupov LOG.txt a fotografie z detekcie kódu. 69
Obrázok 43: Zobrazenie streamovaného videa v prehliadači a Android aplikácií. 71
Obrázok 44: Prvý krok vytvorenia projektu. 76
Obrázok 45: Druhý krok vytvorenia projektu. 77
Obrázok 46: Raspberry Pi umiestnené v kryte, upevnené pri vstupe do miestnosti. 79
Obrázok 47: Mobilná aplikácia 80
Obrázok 48: Výstup pri detekcií pohybu v reálnom čase.. 81

Obrázok 49: Porovnanie HAAR, HOG a CNN detekcie. 84
Obrázok 50: Práca so serverom. 85

Zoznam tabuliek

Tabuľka 1: Špecifikácie vybraných modelov Raspberry Pi (Corral-Garcia et al., 2019).. 20
Tabuľka 2: Porovnanie rýchlosti spracovávania obrazu na Raspberry Pi. 82

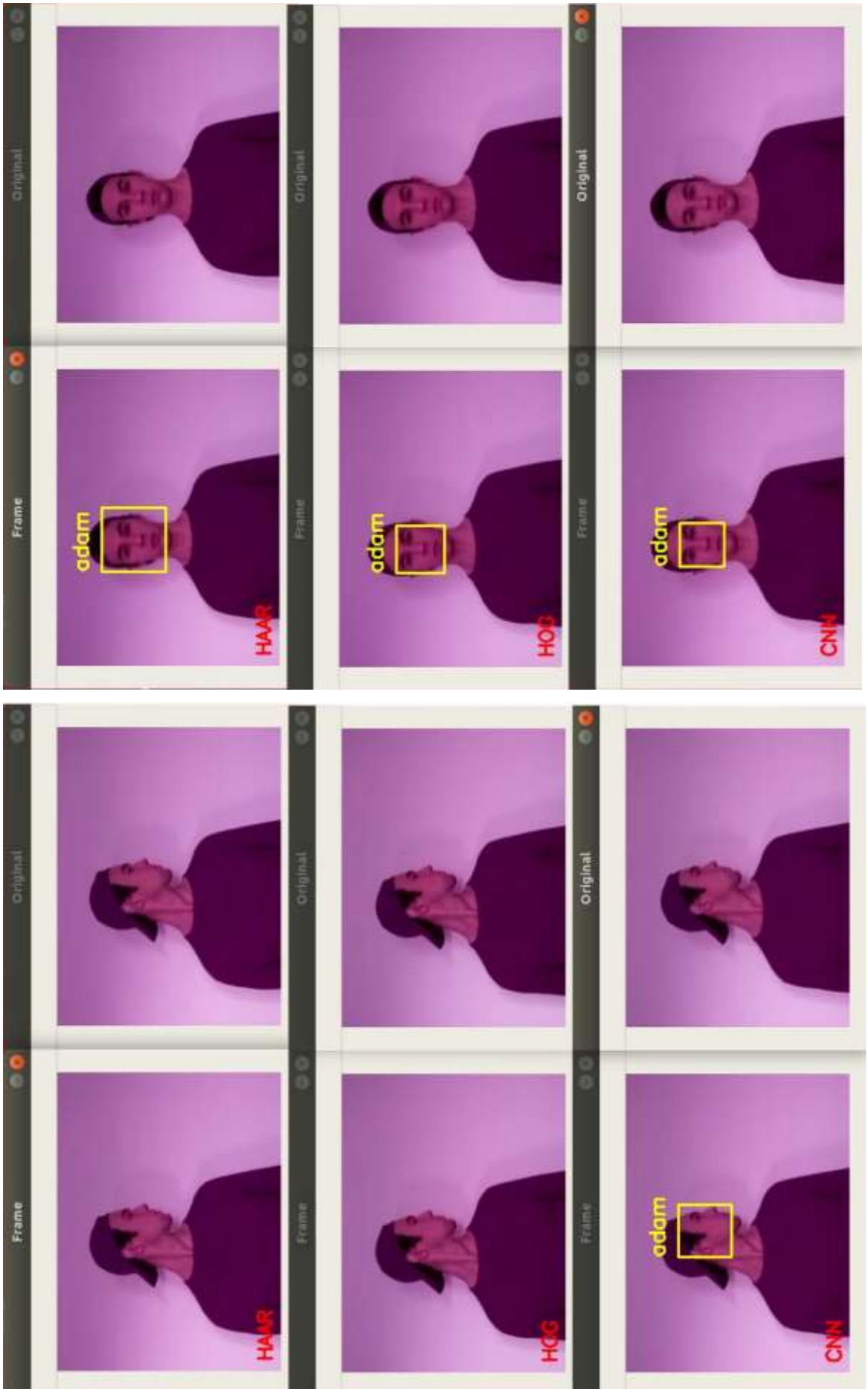
Zoznam príloh

Príloha A: Porovnanie detekcie metód HAAR, HOG, CNN pri rôznych natočeniach tváre

Príloha B: Ukážka detekcie pri prekrytej tvári; posledný detegovaný a prvý nedetegovaný obraz

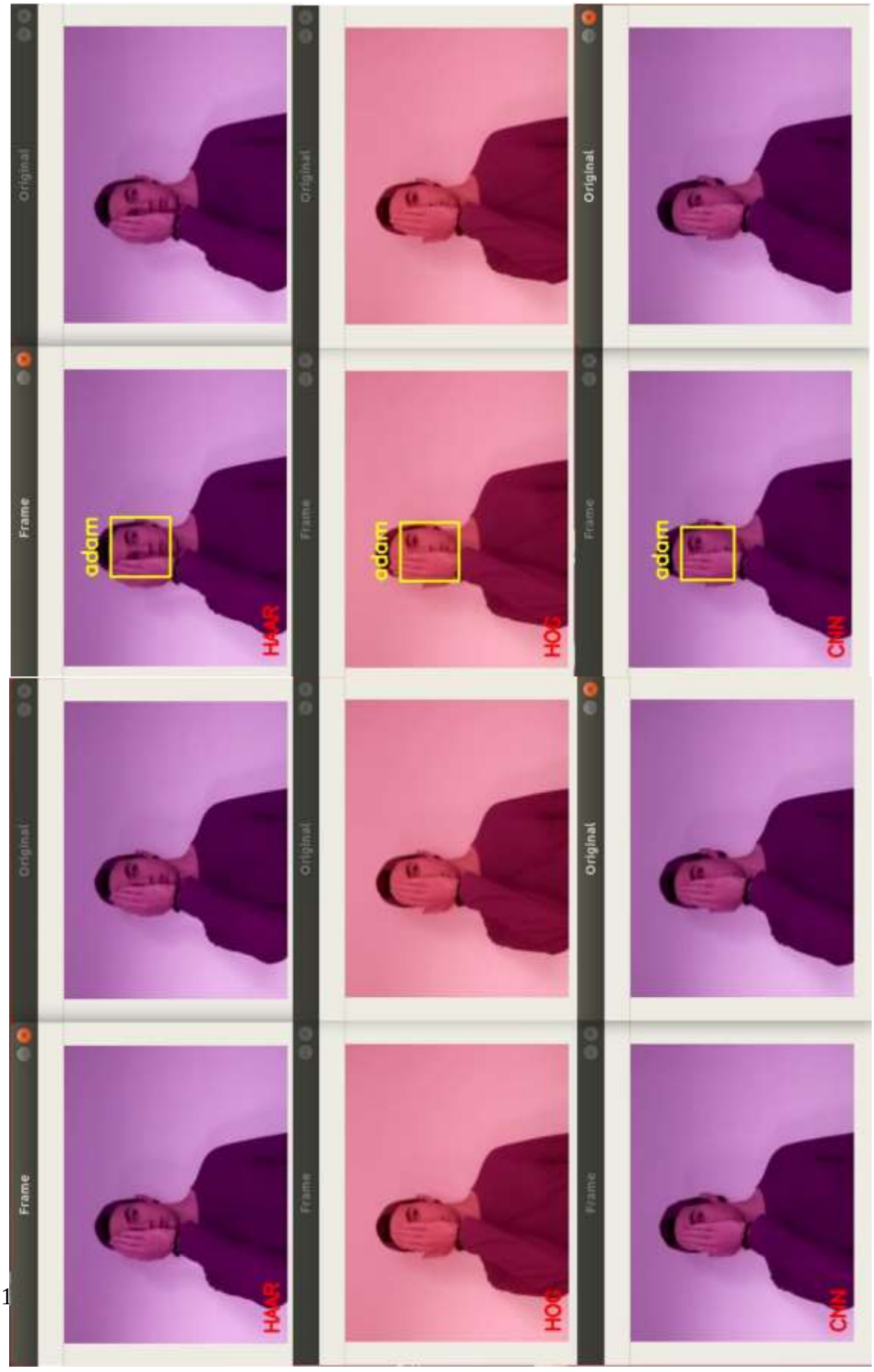
Príloha A :

Porovnanie detekcie metód HAAR, HOG, CNN pri rôznych natočeniach tváre



Frame
Original
HAAR
Frame
Original
HOG
Frame
Original
Unknown
CNN

Frame
Original
adam
HAAR
Frame
Original
adam
HOG
Frame
Original
adam
CNN

Príloha B:

Ukážka detekcie pri prekrytej tvári; posledný detegovaný a prvý nedetegovaný obraz