Stredná odborná škola
Športová 675, 916 01 Stará Turá

STREDOŠKOLSKÁ ODBORNÁ ČINNOSŤ

Číslo odboru: 11

Názov odboru: Informatika

SMART GATE - inteligentný vstupný systém

2019

Stará Turá

Riešitelia

Roman Alexander Mariančík

Ročník štúdia: štvrtý

Stredná odborná škola
Športová 675, 916 01 Stará Turá
Trenčiansky samosprávny kraj

STREDOŠKOLSKÁ ODBORNÁ ČINNOSŤ

Číslo odboru: 11

Názov odboru: Informatika

SMART GATE - inteligentný vstupný systém

2019

Stará Turá

Riešitelia

Roman Alexander Mariančík

Konzultant: Ing. Peter Petkov

Ročník štúdia: štvrtý

Čestné prehlásenie

Čestne prehlasujem, že svoju prácu na tému *SMART GATE - inteligentný vstupný systém* som vypracoval/a samostatne pod vedením Ing. Petra Petkova a s použitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ďalej prehlasujem, že tlačená a elektronická verzia práce SOČ je zhodná a nemám závažný dôvod proti sprístupňovaniu tejto práce v súlade so zákonom č. 618/2003 Z.z., o autorskom práve a právach súvisiacich s autorským právom (autorský zákon) v platnom znení.

V Starej Turej dňa: _____

_____

Roman Alexander Mariančík

Poďakovanie

Týmto sa chcem poďakovať mojim spolužiakom Jaroslavovi Šopíkovi, Jurajovi Kulichovi a Timotejovi Masárovi za všestrannú pomoc a podporu počas procesu tvorenia tejto práce. Špeciálne poďakovanie patrí Jaroslavovi Šopíkovi za zoznámenie ma s kvalitným typografickým systémom LaTex, v ktorom bola napísaná táto dokumentácia. Písanie v tomto systéme nebolo vždy jednoduché ale vďaka ochotnej pomoci Jaroslava Šopíka a Juraja Kulicha sa táto práca stala realitou a s výsledkom som nadmieru spokojný.

Obsah

Úvod

Predmetom tejto práce je komplexný vstupný systém s jednoduchou obsluhou, bez nutnosti používania kľúčov na vstup do objektov s nižšou úrovňou zabezpečenia. Zároveň demonštruje ohromujúce schopnosti neurónových sietí na poli spracovania obrazu.

Originálna myšlienka na vytvorenie tohto systému mi napadla, keď som jedného dňa prichádzal domov, mal som plné ruky a tým pádom som si nemohol odomknúť bránku. Dávnejšie som sa zaujímal o spracovanie obrazu použitím neurónových sietí. Tak mi napadlo, že by bolo zaujímavé vyriešiť taký banálny „problém“ ako otvorenie si bránky použitím komplexného programu počítačového videnia. Až neskôr sme si uvedomili, že tento program funguje ozaj dobre a má širšie uplatnenie. Nemusí nutne otvárať nejaký objekt kvôli obávam o spoľahlivé zabezpečenie ale s minimálnymi úpravami môže napríklad monitorovať príchody a odchody zamestnancov, počítať vstupy ľudí a pod.

Programovanie ma odjakživa bavilo a keď som objavil svet umelej inteligencie a strojového učenia bol som fascinovaný čo dokážu stroje dosiahnuť. Táto fascinácia jednoznačne pokračuje do dnešného dňa a určite bude pokračovať aj na vysokej škole tohto zamerania. Práve preto som bol nadšený, keď sa mi podarilo spojazdniť a aspoň teoreticky pochopiť použitú knižnicu hĺbkového strojového učenia Dlib. Samozrejme samotné jadro rozpoznávania tvárí som neprogramoval ja. Jedná sa o mimoriadne náročnú úlohu hodnú dizertačnej práce, ale vďaka zverejneniu tejto fantastickej knižnice, som mohol aspoň implementovať tento kód a vytvoriť z neho vlastnú aplikáciu.

1 Problematika, prehľad literatúry

1.1 Umelá inteligencia

Kaplan a Haenlein definovali umelú inteligenciu ako: „Schopnosť systému správne interpretovať externé dáta, učiť sa z týchto dát a použiť tieto znalosti na dosiahnutie špecifického cieľa a úloh pomocou flexibilnej adaptácie“ [1].

Umelá inteligencia (ďalej len AI z anj. artificial inteligence) je využívaná v oblastiach, ktoré sú príliš komplexné na to aby bol vyvinutý konkrétny algoritmus podľa ktorého by sa daný problém vyriešil. Napríklad na spracovanie prirodzeného jazyka, spracovanie veľkého množstva komplexných dát, v medicíne na analýzu tkaniva podrobeného vyšetreniam MRI/CT a analýzu liekov, v autonómnych vozidlách, práve v oblasti spracovania obrazu a zvuku a mnohých ďalších oblastiach.

V súčasnosti je umelá inteligencia schopná riešiť špecifické úlohy, ale stále sa nejedná o všeobecnú tzv. silnú umelú inteligenciu schopnú riešiť akúkoľvek úlohu, práve týmto smerom sa uberá vývoj [2].

Strojové učenie - machine learning

Strojové učenie sa považuje za podoblasť umelej inteligencie. AI sa počas behu zlepšuje v presnosti svojich predikcií a vytvára všeobecný model z tréningových dát, ktorý AI umožňuje robiť samostatné predikcie a rozhodnutia bez explicitného naprogramovania na danú činnosť. Učenie si vyžaduje čo najväčšie možné množstvo dát. Pri učení bez učiteľa (unsupervised learning) AI dokáže nájsť vzory vo vstupných dátach, bez nutnosti pridelenia štítkov (labels) ku každej hodnote v dátach človekom. Pri učenie s učiteľom (supervised learning) človek najskôr pridelí štítky k dátam. Následne je AI trénovaná na označkovanom datasete (t.j. súbore dát).

Učenie s učiteľom sa delí na dva základné typy:

- Klasifikácia (classification) pri tomto type sa AI naučí rozhodnúť do ktorej z predom definovaných kategórií testované dáta patria. Takže výstupné dáta naberajú diskrétne hodnoty. Typickým príkladom je SVM (support vector ma-

chine – podporný vektorový stroj). Jedná sa o model, ktorého cieľom je odhadnúť lineárnu hranicu predeľujúcu tréningové dáta do dvoch kategórií [3][4].

- Regresia (regression) AI sa snaží pomocou spätnej väzby čo najpresnejšie odhadnúť funkciu, ktorá reprezentuje vzťah medzi vstupnými a výstupnými dátami. Tým pádom výstupné dáta môžu naberať akúkoľvek hodnotu z určeného rozsahu [3][4].

Neurónové siete

Neurónové siete sú výpočtové systémy čiastočne inšpirované štruktúrou biologického mozgu. Samotné neurónové siete nie sú algoritmus, ale skôr štruktúra pre viaceré algoritmy strojového učenia, ktoré spoločne spracovávajú komplexné dátové vstupy. Neurónová sieť je založená na veľkom množstve uzlov, nazývaných umelé neuróny, ktoré vzdialene pripomínajú neuróny v biologickom mozgu. Každý spoj prenáša signály od jedného neurónu k druhému, tak ako v biologickom mozgu [4] (viď. príloha A, Obr. 1).

Hĺbkové učenie - deep learning

Deep learning je skupina algoritmov strojového učenia, ktorá využíva kaskádu viacerých vrstiev jednotiek spracovania pre extrahovanie vlastností. Každá nasledujúca vrstva používa výstupy z predchádzajúcej ako vstupy. Najpoužívanejším typom sú práve hĺbkové neurónové siete (deep neural networks). Rozdiel oproti jednoduchším algoritmom strojového učenia spočíva v princípe extrahovania dát. V klasických algoritmoch strojového učenia je extrahovanie dát vykonané človekom, zatiaľ čo pri hĺbkovom učení sa o toto postará samotná neurónová sieť. Učenie je rozdelené na učenie s učiteľom pri klasifikačných a regresných aplikáciách a na učenie bez učiteľa pre analýzu vzorov vo veľkom množstve rozličných neoznačených dát [5].

V hĺbkovom učení sa každá vrstva naučí transformovať vstupné dáta do zakaždým o niečo viac abstraktnej podoby. Napríklad pri probléme rozpoznávania tvárí, ktorým sa zaoberá aj táto práca je približný postup nasledovný: na úplne prvej úrovni môžu byť surové dáta (fotografia) v podobe matice pixelov, táto vrstva môže abstrahovať pixely a zakódovať hrany pre lepšie spracovanie, na ďalšej vrstve

sa skomponuje a zakóduje rozloženie hrán. Tretia vrstva môže zakódovať nos a oči a štvrtá rozpoznať že fotografia obsahuje tvár. Dôležité je, že počas hĺbkového učenia sa neurónová sieť naučí rozdeliť tieto činnosti do vrstiev sama, bez zásahu človeka. Hĺbkové učenie (deep learning) dostalo svoj názov práve kvôli týmto vrstvám [6].

ResNet34

Problémom s hĺbkovými neurónovými sieťami je, že je ich náročné optimalizovať. Často sa stáva, že hlbší model (viacero vrstiev) nie je o nič presnejší, či dokonca horší ako plytšie modely (menej vrstiev). Prekvapivo toto nie je spôsobené preučením (overfitting) [7].
Tento problém rieši napríklad aj zvyšková hĺbková neurónová sieť ResNet (Residual Network) vyvíjaná viacerými veľkými organizáciami ako Microsoft Research a Facebook Research [8].
Táto neurónová sieť bola mimoriadne úspešná a taktiež je základom pre neurónovú sieť implementovanú v knižnici Dlib, ktorú využívame [9].

1.2 Počítačové videnie

Počítačové videnie je odvetvie informatiky zaoberajúce sa vytváraním softvéru a zariadení schopných získať definujúce informácie zo zachyteného obrazu. Umožňuje spracovanie fotografií a videa. Uplatňuje sa v mnohých odvetviach ako medicína, automatizácia, automobilový priemysel, monitorovanie objektov a pod.

Počítačové videnie je väčšinou implementované v spojení s umelou inteligenciou. Počítačové videnie poskytuje dáta z okolia (vstupy), ktoré sú privedené do umelej inteligencie na spracovanie, ktorá následne reaguje na tieto podnety [10].

V našej práci využívame OpenCV (knižnica počítačového videnia) na načítanie snímok z priemyselnej IP kamery do programu.

1.3 Rozpoznávanie tvárí

Rozpoznávanie tvárí je technológia určená na identifikáciu a overenie identity osoby z digitálnej fotografie alebo snímky videa. Fungujú na rôznych princípoch ale všeobecne porovnávajú extrahované črty tváre podrobenej testovaniu s črtami

tvárí uložených v databáze. Presnosť tejto metódy je o niečo nižšia ako pri iných druhoch biometrie ako sú identifikácia pomocou odtlačkov prstov alebo analýzou očnej zornice.
Všeobecne sa delia na:

- Tradičné systémy – pracujú na podobnom princípe ako je popísané vyššie. Extrahuje určité geometrické črty tváre z 2D snímok, napríklad relatívnu polohu a veľkosť očí, nosa, sánky atď. a porovnávajú ich s tvárami v databáze. Táto práca využíva práve túto metódu [11].
- 3D systémy – ako názov napovedá, systém vytvára z tváre trojrozmerný model pomocou viacerých senzorov. Táto metóda rapídne zvyšuje presnosť identifikácie. Systém nie je citlivý na zmeny v osvetlení alebo natočenie tváre [12].
- Analýza textúry pokožky tváre – systém extrahuje špecifické čiary, línie a body na pokožke tváre a prenáša ich do matematického modelu s ktorým môže robiť ďalšie výpočty [11].

1.4 Django a web

Webový server je program bežiaci na dedikovanom počítači tiež nazývanom server. Jeho úlohou je prijímať požiadavky od klientov (webových prehliadačov), spracovať požiadavky (napr. interakciou s databázou) a odoslať odpoveď (napr. vyplnenú údajmi z databázy). Prehliadač následne interpretuje obdržanú webovú stránku v podobe HTML, s použitím CSS, ktoré reprezentuje formátovanie stránky a Java Scriptom, ktorý poskytuje určitú dynamiku web stránky na strane používateľa. Django značne zjednodušuje tieto činnosti, umožňuje tvorbu aplikácií ktoré sú:

- Kompletné - Django obsahuje takmer všetko, čo môže developer požadovať, všetko funguje medzi sebou spoľahlivo a obsahuje rozsiahlu dokumentáciu.
- Všestranné - Django môže byť použité na takmer akýkoľvek druh aplikácie. Dobre spolupracuje s akýmkoľvek typom front endu a umožňuje dodať obsah v akomkoľvek formáte (napr. HTML, RSS, JSON, XML atď.)
- Bezpečné - Django pomáha developerom vyvarovať sa mnohým bežným chybám. Napríklad Django poskytuje bezpečný spôsob spravovania používateľských účtov a hesiel. Heslá nie sú vymieňané cez cookies a heslá nie sú ukladané

ako jednoduchý text. Miesto toho je heslo zahashované a uložené v tejto podobe. Aj keby sa útočník zmocnil zahashovanej verzie hesla stále je veľmi zložité zistiť pôvodné heslo. Django ďalej poskytuje ochranu pred mnohými náchylnosťami ako je SQL Injection, cross-site scripting, cross-site request forgery a clickjacking.

- Škálovateľné - Django je založené na komponentnej architektúre kde nič nie je zdieľané, takže pri zvýšení prevádzky je jednoduché pridať databázové alebo aplikačné servery. Niektoré veľmi vyťažené stránky ako Instagram alebo Disqus úspešne rozšírili Django aby spĺňal ich požiadavky.
- Udržiavateľné - Django povzbudzuje princípy DRY (Don't repeat yourself - neopakuj sa). Maximalizuje znovu použiteľný kód vďaka čomu je jednoduchšie tento kód udržovať.
- Prenositeľné - Django je zapísané v Pythone, takže beží na väčšine bežne používaných platforiem a navyše veľa poskytovateľov umožňuje nasadenie webového servera priamo v Djangu [13].

Model-View-Template architektúra

Django využíva architektúru Model-View-Template (viď. príloha A, Obr. 2) Model predstavuje dáta, modely vytvárajú a pracujú s databázou. View (pohľad) prijíma požiadavku, vykonáva ľubovoľné činnosti a výpočty s dátami a následne pošle odpoveď webovému prehliadaču. Väčšinou vyplnením šablóny (template).
Kód nutný na činnosť Djanga je väčšinou rozdelený do viacerých súborov:

- URL adresy (*urls.py*) – používa sa na presmerovanie http žiadosti k správnej funkcií v rámci súboru *views.py*.
- View (*views.py*) – súbor s funkciami, ktoré spracovávajú žiadosti. Prijímajú http žiadosti a vracajú http odpovede. View pristupuje k dátam pomocou modelom a prenecháva formátovanie žiadosti a vyplnenie dátami na šablóny (templates).
- Modely (*models.py*) – objekty Pythonu, ktoré definujú štruktúru dát danej aplikácie a poskytujú mechanizmus na spracovanie a získavanie záznamov z databázy.
- Šablóny – HTML súbor definujúci rozloženie stránky. Obsahuje premenné, ktoré sú obsadené dátami z modelov pomocou funkcií vo *views.py* [13].

2 Ciele práce

Cieľom práce je vytvorenie spoľahlivého a presného vstupného systému s rozpoznávaním tvárí využitím vyššie popísaných konceptov. Cieľom práce nie je priblížiť sa k zaužívaným vysokovýkonným systémom rozpoznávania tvárí nasadených v rôznych veľkých korporáciách. Uplatnenie vidíme skôr v súkromnej sfére. Malí podnikatelia by mohli z podobného systému značne benefitovať, na vysvetlenie tohto tvrdenia je najskôr nutné pochopiť základnú činnosť tohto zariadenia.

Systém bude pozostávať z IP kamery umiestnenej pri hlavnom vstupe, Arduina, ktoré bude podľa potreby otvárať vstup a Python programu bežiaceho vo virtuálnom počítači na serveri. Dve hlavné úlohy programu budú rozpoznávanie tvárí a poskytovanie webového servera. Rozpoznávanie tvárí umožňuje knižnica strojového učenia Dlib a webový server je zabezpečený pomocou webového frameworku Django.

Ak sa bude pred kamerou nachádzať známa tvár a s prístupom do objektu, systém otvorí vstup. Ak tvár bude známa ale bez prístupu, systém zapíše informáciu o osobe do databázy do sekcie záznamov (logs). Ak tvár bude neznáma, taktiež sa zapíše informácia o osobe spolu s fotografiou osoby. Celý systém bude spravovateľný pohodlne cez webový server s bohatou funkcionalitou. S cieľom obmedziť nutnosť servisných zásahov a prístupov na server (napríklad na reštartovanie systému).

Podobná činnosť zariadenia by mohla nájsť uplatnenie v rôznych firmách, v situáciách kde je nutné monitorovanie osôb. Napríklad monitorovanie dochádzky zamestnancov, dochádzky študentov, zabránenie zneužívaniu skipassov neskorším predajom iným osobám, zabezpečenie objektov sledovaním osôb - v prípade, že by sa v objekte nachádzala nepovolaná osoba systém by ohlásil situáciu zodpovednej osobe, generovanie štatistík návštevnosti obchodov - systém by počítal neznáme osoby. Všetky tieto a ďalšie variácie použitia systému sú reálne a relatívne jednoducho uskutočniteľné minimálnymi modifikáciami základného programu.

Na dosiahnutie týchto cieľov je potrebné najskôr spracovať mechanizmus rozpoznávania tvárí, následne vyriešiť komunikáciu s Arduinom po sieti LAN pomocou ethernetového modulu a nakoniec integrovať tieto systémy s Djangom aby vznikol ucelený systém a aby jeho jednotlivé komponenty medzi sebou dobre spolupracovali.

3 Metodika práce

3.1 Použité technológie

Program bol vyvíjaný v programovacom jazyku Python 3.6, hlavne pre jeho jednoduchú a prehľadnú syntax a obrovské množstvo vysokokvalitných knižníc. Python je veľmi silný jazyk na všeobecné použitie, často využívaný vo vedeckej sfére.

Dlib

Dlib je knižnica zapísaná v C++ so všeobecným zameraním. Knižnicu vytvoril Davis King a je zverejnená pod softwarovou licenciou Boost, ktorá umožňuje jej voľné použitie. Táto knižnica je jadro programu, stará sa o detekciu a rozpoznávanie tvárí.

Numpy

Matematická knižnica, ktorú využívame na porovnanie tvárí výpočtom euklidovských vzdialeností medzi bodmi v 128 dimenzionálnom priestore. Podobné tváre sú v tomto priestore blízko pri sebe, odlišné ďalej.

OpenCV

Knižnica počítačového videnia. Používame ju na čítanie videa zo streamu IP kamery a úpravu obrazu pred spracovaním knižnicou Dlib (zmenšenie, prevedenie na čiernobiele snímky).

Django

Vysokoúrovňový Python webový framework, ktorý je zameraný na rapídny a bezpečný vývoj komplexných webových stránok. Poskytuje širokú funkcionalitu, takže vývojár sa môže zamerať na programovanie webovej aplikácie bez nutnosti „znovu objavovať koleso“ [13].

MySQL

MySQL je výkonná relačná databáza s architektúrou klient-server, prispôsobená pre firemné prostredie. Django poskytuje bezchybnú integráciu s týmto typom da-

tabázy pomocou modelov.

PyCharm

PyCharm je developerské prostredie pre Python od JetBrains s veľmi rozsiahlymi možnosťami, ktoré značne uľahčovalo a spríjemňovalo vývoj tohto programu.

Ubuntu

Tento program beží na linuxovej distribúcií Ubuntu 18.04LTS. Jedná sa o jednu z najpoužívanejších distribúcií, s bohatou škálou základných balíčkov. Táto distribúcia bola zvolená pre najlepší výkon spracovania a dobrú dostupnosť potrebných balíčkov. Napríklad pri našom testovaní bol čas spracovania videa v Ubuntu približne polovičný oproti Windowsu.

VMware

Operačný systém je spúšťaný ako virtuálny počítač, pre jednoduchší vývoj, prenos a zálohovanie. Rozdiel v rýchlosti spracovania medzi fyzickým a virtuálnym HW nebol spozorovateľný. VMware Worksation je virtualizačný software tzv. hypervisor umožňujúci emuláciu hardwaru. Operačný systém nainštalovaný na tento virtualizovaný hardware sa správa ako keby bežal na fyzickom hardwari. Moderné systémy umožňujú využívať hardwarovú virtualizáciu, čím značne zvyšujú výkon. Guest a host operačné systémy sú kompletne oddelené ale je možné povoliť zdieľanú schránku umožňujúcu kopírovanie medzi nimi. Použitá verzia programu bola Workstation Player 15.

Arduino

Arduino je open-source platforma s kontrolérom Atmel Atmega. Program je zapísaný v jazyku Wiring, ktorý vychádza z jazyka C++. V tomto projekte bolo použité Arduino Mega 2560 s ethernetovým modulom, ktoré po sieti LAN komunikuje s hlavným Python programom a podľa potreby otvára bránku a posiela signál v prípade stlačenia zvončeka.

3.2 Prevedenie rozpoznávania tvárí

Proces rozpoznávania tvárí si vyžaduje určitú sekvenciu separátnych činností aby systém docielil rozpoznanie tváre. Najskôr je nutné nájsť tvár na snímke, následne vykonať určité transformácie aby aj tvár zachytená pod uhlom alebo v zlom svetle mohla byť spracovaná, následne extrahovať špecifické črty tváre a nakoniec porovnať tieto črty s črtami tvárí v databáze.

3.2.1 Detekcia tvárí

Pred začatím rozpoznávania tvárí, musíme najskôr tváre nájsť v jednotlivých snímkach. Na to slúži práve detekcia tvárí. Existuje viacero algoritmov. Jeden z prvých komerčne dostupných bol Viola&Jones v súčasnosti sa však využívajú skôr komplexnejšie a podstatne presnejšie algoritmy pracujúce na princípe HOG. Tieto si ďalej vysvetlíme, keďže aj detektor tvárí implementovaný v knižnici Dlib, ktorú využívame je práve HOG. HOG je skratka pre Histogram of Oriented Gradients, čo by sa dalo preložiť ako histogram orientovaných gradientov.

Snímka sa najskôr konvertuje do čiernobieleho farebného priestoru. Následne algoritmus prejde cez blok pixelov. Algoritmus analyzuje intenzitu bloku pixelov obklopujúcich daný blok pixelov a porovnáva ju s blokom pixelov na ktorom sa práve nachádza. Následne blok pixelov nahradí šípkou, ktorá naznačuje ktorým smerom sa obrázok stáva tmavším (viď. príloha A, Obr. 3). Vďaka tejto analýze detekcia tvárí nepodlieha zmenám osvetlenia ako by sa dialo pri priamej analýze intenzity pixelov.

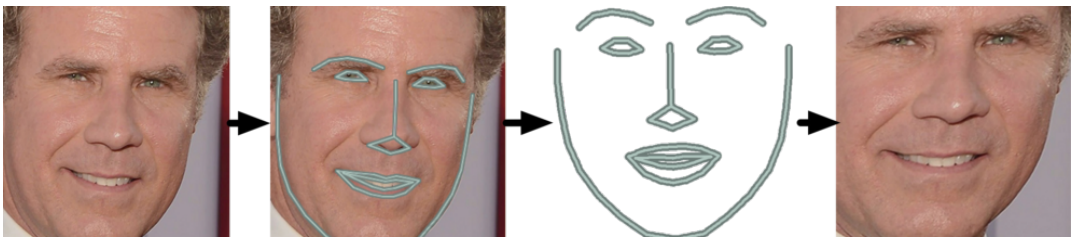
Na nájdenie tváre v snímke už len stačí nájsť časť snímky, ktorá sa najviac podobá na vzor extrahovaný z veľkého množstva tréningových tvárí [14].

3.2.2 Nájdenie špecifických bodov na tvári a premietanie tvárí

Po nájdení tváre na snímke je nutné nájsť určité predom definované body na tvári. V prípade knižnice Dlib konkrétne 68 bodov (viď. príloha A, Obr. 4), ktoré sa nazývajú landmarks. O to sa stará predom natrénovaný algoritmus strojového učenia, ktorý potom automaticky nájde týchto 68 bodov.

Človek si dokáže ľahko poradiť s rozpoznaním objektov (tvárí) aj keď sú natočené

v rôznych smeroch, ale pre počítač je toto veľký problém. Preto je nutné vykonať tzv. afiné transformácie týchto 68 bodov, aby boli tváre vždy natočené presne dopredu a vycentrované [14]. Afinné transformácie sú také transformácie, ktoré zachovávajú body, priame čiary a roviny nezmenené. Jednoduché príklady sú zmena veľkosti, rotácia a podobne [15].



Obr. 3.1: Z ľava: tvár nájdená detektorom tvárí, tvár po spracovaní čŕt tváre, príklad vycentrovanej tváre, tvár zrovnaná vykonaním afinných transformácií, autor: Adam Geitgey [14], (upravené).

3.2.3 Princíp rozpoznávania tvárí s Dlib

Táto knižnica na úlohy rozpoznávania tvárí implementuje neurónovú sieť ResNet34 s 29 konvolučnými vrstvami a niekoľkými odstránenými vrstvami a filtrami. Sieť bola od základu natrénovaná na dátovom sete 3 miliónov tvárí [9]. Knižnica dosahuje vysokú úroveň úspešnosti 99,38% na štandardnom dátovom sete *Faces in the Wild* to znamená, že správne predpovedá, či dva odlišné obrázky tváre patria jednej osobe v 99,38% prípadoch [9].

Detektor tvárí je HOG pracujúci na vyššie popísanom princípe. Detekuje tváre do rozmeru cca. 80x80 pixelov. Detektor vracia koordináty tváre, ktoré sú následne použité pre ďalšie rozpoznávanie a nakreslenie ohraničujúceho rámčeka okolo tváre v live stream videu.

Na extrahovanie čŕt tváre sa používa tzv. *shape predictor*. Na svoju činnosť potrebuje načítať predom natrénovaný model. Funkcia tohto objektu má ako argumenty snímku a koordináty tváre. Ďalej spracuje snímku a vráti 68 bodov, ktoré definujú tvár na obrázku a sú pevne dané. Tieto body sú ďalej využívané samotnou ResNet34 neurónovou sieťou a nami naprogramovaným detektorom žmurkania.

Na samotné rozpoznanie tvárí (odlíšenie 2 tvárí) využívame model rozpoznávania tvárí, ktorý je predom naučený s dátovým setom obsahujúcim 3 milióny tvárí. Fun-

kcia tohto objektu vypočíta deskriptory danej tváre. Argumenty tejto funkcie sú snímka a črty tváre (landmarks). Deskriptor je 128 meraní získaných zo 68 bodov, ktoré predstavujú črty tváre. Nie je možné poznať o aké merania sa jedná, samotná neurónová sieť si ich určuje počas trénovania aby dosiahla čo najvyššiu úspešnosť. Môže to byť napríklad vzdialenosť medzi nosom a horným viečkom oka a pod. Následne je týchto 128 meraní prenesených do 128 dimenzionálneho priestoru. Podobné tváre sú v tomto priestore blízko pri sebe a odlišné ďalej [14]. Na zistenie vzdialenosti medzi prislúchajúcimi bodmi stačí získať euklidovskú vzdialenosť pomocou vzorca (3.1):

$$\sqrt{\sum_{1}^{128}(a-b)^2} \tag{3.1}$$

kde:

- a je deskriptor (jedno meranie neurónovej siete) známej tváre
- b je deskriptor (jedno meranie neurónovej siete) neznámej tváre

Na tento účel využívame funkciu *numpy.linalg.norm()* z knižnice Numpy, ktorá poskytuje lepšiu optimalizáciu a rýchlosť pretože je zapísaná v C++.
Počet iterácií je 128 lebo pole deskriptorov má dĺžku 128. Takto sa neznáma tvár porovná voči každej známej v databáze. Toto môže mierne spomaľovať rozpoznávanie pri veľkom množstve tvárí v databáze. Systém sme testovali s databázou s 200 tvárami a spomalenie nebolo citeľné ale určite by sa pri ďalšom pridávaní tvárí skôr či neskôr prejavilo. Preto do budúcna plánujeme implementovať SVM – support vector machine (podporný vektorový stroj) na klasifikáciu tvárí do skupín, čo by celý proces značne urýchlilo a čas nutný na klasifikáciu by nestúpal s počtom tvárí v databáze.

Detektor žmurkania

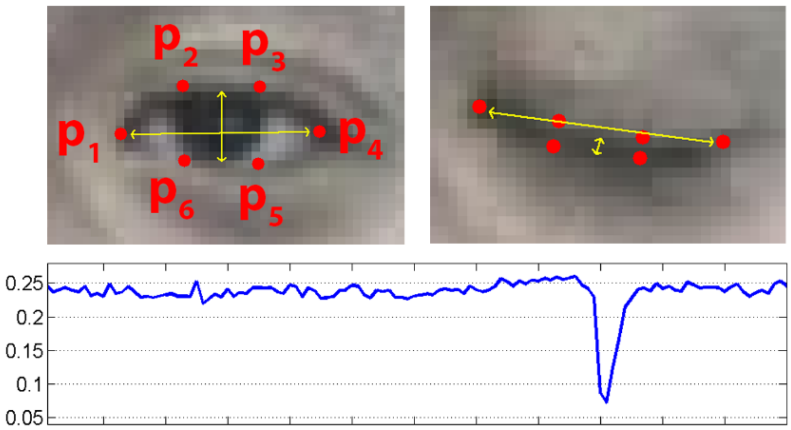
Detektor žmurkania bol implementovaný ako základná ochrana pred oklamaním systému vystavením fotografie známej osoby pred kameru.

Detektor žmurkania je založený na metóde použitej v práci *Eye-Blink Detection Using Facial Landmarks od Terezy Soukupovej* [16]. Jedná sa o funkciu, ktorá má argumenty črty tváre, z ktorých izoluje predom definované body na oku. Rozloženie všetkých bodov čŕt tváre je možné vidieť na obrázku 4 v prílohe A. Následne z týchto

bodov vypočíta pomer medzi výškou a šírkou oka (v citovanej práci nazvané EAR - eye aspect ratio) podľa vzorca (3.2):

$$EAR = \frac{\sqrt{(p2_x - p6_x)^2 + (p2_y - p6_y)^2} + \sqrt{(p3_x - p5_x)^2 + (p3_y - p5_y)^2}}{2 * \sqrt{(p1_x - p4_x)^2 + (p1_y - p4_y)^2}} \tag{3.2}$$

Kde *p1* až *p6* sú body v rovine medzi, ktorými sa počítajú euklidovské vzdialenosti a ich indexy x a y predstavujú koordináty týchto bodov.



Obr. 3.2: Umiestnenie bodov na oku, graf zobrazuje hodnotu EAR, výkyv predstavuje žmurknutie, autor: Tereza Soukupová [16].

Detektor žmurkania funguje tak, že na dvoch snímkach musí mať osoba zatvorené oči a na troch otvorené aby detektor udelil prístup. V praxi to funguje rýchlo a relatívne spoľahlivo ak je hodnota EAR a počty snímok dobre skalibrované.

3.3 Štruktúra programu

Rozpoznávanie tvárí

Spracovanie a rozpoznávanie tvárí zabezpečuje trieda *FaceRecognition()* zo súboru *FaceRecAPI.py*. Táto trieda sa stará o všetky činnosti súvisiace so spracovaním obrazu a tvárí. Napríklad načítavanie videa, zmenšenie veľkosti videa, funkcionalitu knižnice Dlib, implementáciu detektora žmurkania, preverenie prístupu osoby do objektu, zapisovanie do logov, komunikáciu s Arduinom a na základe vyhodnotenia prístupu aj otvorenie bránky.

Projekt *facerecnet*

Základnú štruktúru programu vytvára Django pri zadaní príkazu *django-admin startproject mysite* do terminálu.

Súbory nutné na činnosť a správu servera

Skript slúžiaci na správu servera je *manage.py*. Napr. spustenie servera sa vykoná príkazom *python manage.py runserver [serverIP:PORT]*. Ďalším dôležitým súborom je *settings.py*, ktorý obsahuje nastavenia webového servera, kľúče potrebné na šifrovanie atď. A *wsgi.py* je rozhranie na komunikáciu s projektom pri prípadnom nasadení serveru do produkcie.

urls.py

Na základe tohoto súboru Django presmerováva HTTP žiadosti na korešpondujúce funkcie, pomocou funkcie *path()*, ktorá vráti element vhodný na uloženie do premennej *urlpatterns*.
Cesta môže vyzerať napríklad *http://192.168.1.9:8000/subscribe*

views.py

Obsahuje funkciu renderujúcu domovskú stránku, funkciu na prihlásenie sa k prijímaniu notifikácií a na odhlásenie sa z prijimánia notifikácií.

Aplikácia *LiveView*

Ďalej zadaním príkazu *python manage.py startapp myapp* sa vytvorí aplikácia. V našom prípade aplikácia obsahuje väčšinu funkcionality programu. Aplikáciu sme nazvali trochu nenáležite *LiveView* a to z toho dôvodu, že pôvodne sa mala starať len o streamovanie live videa na webovú stránku, ale neskôr bolo zistené, že je jednoduchšie zakomponovať väčšinu funkcionality do jednej aplikácie.

admin.py

V tomto súbore sa definuje obsah administračného rozhrania Django Admin. Definuje sa, ktoré polia modelov (databázy) budú zobrazené a ako. Taktiež sú tu

definované funkcie, ktoré vykonávajú akcie po stlačení nadštandardných tlačidiel administračného rozhrania.

models.py

V tomto súbore sú deklarované modely, na základe ktorých Django vytvorí korešpondujúce tabuľky a polia v MySQL (alebo akejkoľvek inej) databáze. Každá trieda (ktorá dedí z *models.Model*) tohto súboru predstavuje tabuľku v databáze. Polia tabuliek sa definujú vytvorením objektu zo zdedenej triedy *models*.

urls.py

Princíp rovnaký ako pri *urls.py* celého projektu. Ale v tomto prípade sa cesty definované v tomto súbore vzťahujú k aplikácií *LiveView*.
Cesta môže vyzerať napríklad *http://192.168.1.9:8000/LiveView/start*.

views.py

Obsahuje najviac funkcií súvisiacich s integráciou rozpoznávania tvárí a Django webu. Stará sa o spúšťanie vlákien potrebných na rozpoznávanie tvárí. Triedy a funkcie by sa dali rozdeliť do niekoľkých skupín:

- Triedy deklarujúce zastaviteľné vlákna - Medzi ne patria triedy *FaceRecognitionThread()*, *ArduinoThread()*, *StreamThread()*. Funkcie patriace pod tieto triedy umožňujú zastavenie týchto vlákien pomocou udalostí (tzv. Event). Štandardné vlákno v Pythone nie je možné ľubovoľne zastaviť.
- Trieda *RecognitionThreads()* - Táto trieda vytvára objekty všetkých vlákien a ThreadPoolov potrebných pre rozpoznávanie tvárí. A taktiež obsahuje funkciu *startrecognition()*, ktorá spúšťa rozpoznávanie tvárí.
- Funkcia *facerecognition()* - Jadro celého rozpoznávania tvárí. Obsahuje cyklus, v ktorom sa spracováva obraz načítaný z kamery. Funkcia spája všetky časti dohromady.
- Funkcia *stream_server()* - načítava snímky z fronty (*Queue*) a konvertuje ich do bytestreamu. Toto je nutné pre streamovanie videa do web stránky.
- Funkcie pracujúce s webovými žiadosťami - Sú volané z webovej stránky po presmerovaní cez *urls.py*. Spracovávajú žiadosti a odpovedajú na ne.

HTML

Šablóny v Djangu obsahujú syntax na jednoduché vytváranie hierarchií medzí šablónami, napĺňanie šablón premennými, vytváranie riadiacich podmienok atď. Štruktúra šablón v tejto práci je nasledovná:

- *base.html* - Základná šablóna v ktorej sú vložené odkazy na CSS súbory.
- *home.html* - Rozširuje predchádzajúcu šablónu. Dopĺňa prakticky všetko, čo je možné vidieť na hlavnej stránke.
- *LiveView.html* - Rozširuje predchádzajúcu šablónu o správy pre používateľa informujúce o úspešnosti akcií vykonaných z hlavnej stránky.
- Šablóny *change_list.html* a *change_list2.html* upravujú vzhľad administračného rozhrania. Konkrétne ho dopĺňajú o tlačidlá a odkazy.

CSS

Kaskádové štýly CSS vytvárajú vzhľad stránky. V tejto práci boli použité voľne dostupné štýly Bootstrap na spracovanie celkového vzhľadu stránky a Google Material Icons a Font Awesome icons na vloženie ikoniek. Vzhľad stránky je orientovaný na filozofiu material designu a minimalistického štýlu.

JavaScript

Súbory *sw.js*, *registerSw.js* a *site.js* poskytujú funkcionalitu odosielania push notifikácií. Kód bol prevzatý z internetu [17].

4 Výsledky práce

4.1 Funkcionalita

Činnosť programu začína spustením servera kliknutím na odkaz na pracovnej ploche s názvom Face Recognition Server. Tento odkaz odkazuje na skript, ktorý spustí server s aktuálnou IP adresou servera a portom 8000. Následne sa načítajú súbory nutné na činnosť rozpoznávania tvárí. Po pripojení sa na IP adresu servera je používateľ privítaný prihlasovacou stránkou. Prihlásenie je zabezpečené autentifikačným modelom od Djanga, ktorý aj v predvolených nastaveniach poskytuje dobrú bezpečnosť.

Spustenie a činnosť rozpoznávania tvárí

Po prihlásení je používateľ presmerovaný na domovskú stránku. Odtiaľ je možné spustiť rozpoznávanie tvárí. Tento úkon zavolá funkciu *startrecognition()*. Vlákno *FaceRecognitionThread* sa dostane do cyklu v ktorom sa volajú funkcie *process()* a *access()* z triedy *FaceRecognition()*. Funkcia *process()* načíta snímku z kamery, zmenší ju, prevedie do čiernobielej, nájde na nej tváre, spracuje črty tvárí a deskriptory každej tváre, porovná ich s tvárami v databáze, zavolá funkciu *blink_detector()*, ktorá rozhodne či tvár žmurkla a do snímky nakreslí rámčeky ohraničujúce tváre na snímke, následne vráti snímku (*frame*) a list štítkov (*labels*) patriaci rozpoznaným tváram. Funkcia *access()* počíta na koľkých snímkach bola tvár vyhodnotená ako neznáma (unknown), známa ale bez prístupu alebo známa a s prístupom. Toto sme zaviedli na bezpečné identifikovanie tváre a zabránenie falošným pozitívnym alebo negatívnym identifikáciám. Ak je tvár bezpečne zaradená do jednej z kategórií, vykoná sa jedna z nasledovných činností.
Ak je tvár:

- neznáma a zároveň zazvonila pošle sa push notifikácia a uloží sa čas pokusu o prístup a snímka tváre do databázy. Ak nezvonila, nestane sa nič.
- známa ale nemá prístup uloží sa do databáze meno a čas pokusu o prístup a pošle sa push notifikácia s menom danej osoby

- známa a má prístup a žmurkla pošle sa signál po sieťových Unix socketoch pre Arduino, ktoré následne otvorí bránku.

Uložená snímka neznámej tváre môže byť stiahnutá, môže sa jej priradiť meno a uložiť do databázy známych tvárí.

Zastavenie rozpoznávania tvárí

Počas behu rozpoznávania tvárí tlačidlo na spustenie procesu zmení farbu a stane sa z neho tlačidlo na ukončenie rozpoznávania tvárí. Toto tlačidlo nastaví tzv. Event. Funkcia *process()* pri každej iterácií kontroluje stav premennej tohto objektu. Ak zistí že je nastavená pokúsi sa bezpečne ukončiť všetky vlákna podieľajúce sa na rozpoznávaní tvárí. Ak sa presiahne nastavený maximálny čas 15 sekúnd pošle správu na stránku, že nie je možné ukončiť rozpoznávanie tvárí. Toto sa stane ak nie je pripojená kamera alebo Arduino. Za normálnych okolností je vždy možné ukončiť rozpoznávanie tvárí.

Otvorenie streamu a otvorenie bránky

Stlačením tlačidla otvorenie streamu sa otvorí stream videa spracovaného algoritmami rozpoznávania tvárí.
Tlačidlo otvorenia bránky pošle po sieti LAN signál Arduinu na otvorenie bránky.

Push notifikácie

Stlačením tlačidla push notifikácií sa zapisuje do databázy či si daný používateľ želá dostávať notifikácie. Táto zmena je viazaná na používateľa takže sa prejaví na všetkých prihlásených zariadeniach daného používateľa. Push notifikácie je možné prijímať len cez bezpečné pripojenie HTTPS, keďže náš server je určený len na beh v sieti LAN, HTTPS nie je nutné. Z toho dôvodu sme tento problém vyriešili využitím bezpečného tunela Ngrok. Na využívanie push notifikácií je nutné najskôr sa prihlásiť na server cez bezpečnú HTTPS adresu Ngrok tunela a povoliť notifikácie v prehliadači. Pri ďalšom prístupe na server sa už stačí pripájať pomocou jeho IP adresi v sieti a program Ngrok už nemusí na serveri bežať.

4.1.1 Django Admin

Administračné rozhranie Django Admin poskytuje veľmi širokú funkcionalitu „out of the box“ (bez nutnosti konfigurácie). Avšak stránka bola doplnená o určitú funkcionalitu. Horná lišta stránky okrem zobrazenia stránky, zmeny hesla používateľa a odhlásenia umožňuje aj spustenie/zastavenie rozpoznávania, otvorenie streamu a otvorenie bránky. V sekcií *Authentication and Authorization* sa spravujú používatelia pracujúci so systémom a ich prístupové práva. V sekcií aplikácie t.j. *LiveView* sa spravujú nami vytvorené modely. Django ďalej umožňuje jednoduché vyhľadávanie a filtrovanie vo všetkých modeloch.

Logs

Do tohoto modelu (tabuľky) sa ukladajú logy o prístupe. Keďže sa jedná o logy všetky polia sú nastavené na možnosť *read-only* aby ich nebolo možné upravovať. Avšak je ich možné vymazať. Polia tohoto modelu sú meno osoby (prázdne ak je osoba neznáma), dátum pokusu o prístup, informácia či danej osobe bol pridelený prístup a prípadná snímka ak daná osoba bola neznáma.

Persons

Tento model obsahuje všetky „známe“ osoby. Polia tohoto modelu sú meno osoby, zaškrtávacie políčko stanovujúce prístup osoby do objektu a fotografia osoby nutná na spracovanie jej tváre. V hornej časti stránky sú okrem predvoleného tlačidla *Add person* aj doplnkové tlačidlá, ktorých činnosť je definovaná v súbore *admin.py*. Tlačidlo *Load files* načíta súbory (všetky relevantné údaje z databázy a deskriptory uložené na disku) toto tlačidlo je potrebné stlačiť pri zmenách mena alebo prístupu osôb. Ak predmetom zmeny bola aj fotografia je potrebné stlačiť tlačidlo *Run encodings*, ktoré prepočíta deskriptory známych osôb a následne načíta súbory. Tlačidlo *Run encodings* treba stlačiť aj pri pridaní/odobraní osôb.

Settings

Nastavenia je tabuľka databázy obsahujúca len jediný riadok s dvoma stĺpcami. V jednom je Device (zariadenie), v druhom tzv. Crop (orezanie). Zariadenie je rozbaliteľné pole s adresami streamov kamier. Adresa streamu sa definuje v kóde a v poli sa ukazuje len určené meno zariadenia. Stĺpec Crop určuje faktor, ktorým sa vynásobia

oba rozmery videa. Takže HD video 1280x720 sa pri crop faktore 0.25 zmenší na video s veľkosťou 320x180.

4.2 Testovanie

Testovanie rozpoznávania tvárí prebiehalo počas celej doby vývoja programu. Testovanie nebolo systematické a štatistické pre premenlivú povahu problematiky, ale dostatočne podložilo funkcie schopnosť systému. Na začiatku vývoja programu sme testovali aj rozpoznávanie tvárí založené čisto na knižnici OpenCV, avšak toto sa prejavilo extrémne nepraktické a nespoľahlivé. Keďže model rozpoznávania tvárí v OpenCV je jednoduchý algoritmus strojového učenia na svoju činnosť potrebuje veľa dát, konkrétne okolo 50 fotografií danej osoby bolo nutných na relatívne spoľahlivé rozpoznanie. Toto je V reálnych aplikáciách neprípustné. Knižnici Dlib stačí na činnosť jediná fotografia osoby, kedže využíva hĺbkové učenie.

Funkciu kamery plnilo viacero zariadení vrátane mobilov s nainštalovanou aplikáciou IP Webcam, lacná IP kamera ale hlavne profesionálna IP kamera HikVision.

Rozpoznávanie tvárí sa preukázalo ako veľmi spoľahlivé, presné a rýchle, a to aj za zlých svetelných podmienok. Rýchlosť spracovania bola samozrejme v reálnom čase. Približný čas udelenia prístupu známej osobe je 0,5 až 1 sekunda. Rozpoznávanie tvárí dokáže za dobrých svetelných podmienok rozpoznať známu tvár do vzdialenosti cca. 1,6 m pri spracovávaní HD videa 1280x720 s crop faktorom 0.25 t.j. rozpoznávaní tvárí vo videu o veľkosti 320x180. Táto vzdialenosť by narastala priamo úmerne s veľkosťou spracovávaného obrazu. Pri testovaní za rovnakých podmienok, ale s crop faktorom 1, čiže rozpoznávaní tvárí vo videu v plnom HD rozlíšení bola táto vzdialenosť približne 5,2 m. Avšak spracovávanie bolo extrémne pomalé a nedalo sa prakticky využívať. Skúmaním tohto problému, bolo zistené, že detektor tvárí implementovaný v knižnici Dlib nie je v základe zapísaný viacvláknovo, na rozdiel od funkií starajúcich sa o extrahovanie čŕt tváre a samotného rozpoznania.

Komunikácia s Arduinom cez sieťové Unix sockety je relatívne spoľahlivá. V prípade výpadku spojenia je nutné Arduino znovu pripojiť z administračného rozhrania. Arduino spína tranzistor NPN, ktorý spína relé, ktoré následne privedie napätie na elektromagnetický otvárač bránky.

5 Diskusia

Rozpoznávanie tvárí jednoznačne príjemne prekvapilo svojimi schopnosťami. Kvalita knižnice Dlib je na veľmi vysokej úrovni. Systém dokonca nemal problém s rozpoznaním ľudí v okuliaroch (vrátane slnečných). Jediné čo by bolo potrebné upraviť je prevedenie detektora tvárí aby bol schopný vyťažiť všetky jadrá procesora. To by umožnilo spracovávanie väčšieho videa. Podľa našich odhadov kľudne FullHD videa na výkonnejšom počítači. Vďaka tomu by systém mohol rozpoznávať tváre na relatívne veľké vzdialenosti. V momentálnom stave by toto bolo možné len na veľmi rýchlom procesore (nemusí mať nutne veľa jadier).

Jeden z problémov brániaci uplatneniu systému v určitých oblastiach by mohol byť, že známa tvár nasnímaná pod väčšími uhlami a náklonom je vyhodnotená ako neznáma tvár. Podobne pracuje väčšina systémov rozpoznávania tvárí, ale táto vlastnosť sťažuje uplatnenie systému na počítanie neznámych osôb t.j. napr. zákazníkov na tvorenie štatistík. Očakávame, že tento jav by spôsobil značné odchýlky.

Systém nedokáže perfektne určiť či osoba pred kamerou je naozaj fyzická osoba alebo fotografia. Detektor žmurkania zabraňuje určitému množstvu pokusov o oklamanie tohto systému fotografiou ale v niektorých prípadoch vykazoval zmiešané detekcie, navyše krátkym videom osoby s prístupom by útočník oklamal detektor žmurkania za každých okolností a získal by prístup do objektu. Preto do budúcna plánujeme využitie 3D snímača systému Microsoft Kinect, vďaka nemu by systém meral hĺbku tváre - zmenil by sa na oveľa sofistikovanejší 3D systém rozpoznávania tvárí. Ak by toto nebolo možné spojiť s knižnicou Dlib, systém by mohol len overiť či má tvár hĺbku t.j. či to nie je 2D plocha.

Navrhovaný spôsob distribúcie je rozširovaním virtuálneho počítača programu VMWare Workstation. A to z toho dôvodu, že na činnosť rozpoznávania tvárí je nutné veľké množstvo rôznych balíčkov a Python modulov, nehovoriac o integrácií s webovým serverom. Docieliť požadovanú funkčnosť by bolo náročné pri redistribúcií inštalačného média obsahujúceho len zdrojový kód a určité mechanizmy umožňujúce inštaláciu potrebných balíčkov. Počítame s tým, že taký spôsob by vyžadoval veľké množstvo zásahov osoby znalej v oblasti operačných systémov a sietí.

6 Záver

Webový server funguje absolútne bezchybne. Splnil všetky očakávania a v mnohých ohľadoch nás dokonca prekvapil svojou robustnosťou a ľahkosťou vývoja aplikácie. Vďaka integrovanému rozhraniu Django Admin nebolo nutné vytvárať mechanizmy na správu modelov a databáz čo značne zjednodušilo vývoj. Web server má bohatú funkcionalitu a jednoduchú obsluhu. Vďaka webovému serveru je možná správa systému z akéhokoľvek miesta v rámci siete LAN. Nakonfigurovaním smerovania portov (tzv. port forwarding) na routeri je možné systém ovládať z celého sveta, ak má daný používateľ verejnú IP adresu.

Rozpoznávanie tvárí funguje tak ako bolo zamýšľané. Rýchlo a spoľahlivo zistí identitu osoby pred kamerou. Dokonca vidíme aj širšie uplatnenie systému, ktoré však môže byť obmedzené rýchlosťou spracovania a samotným charakterom problematiky rozpoznávania tvárí. Túto funkcionalitu systému je možné s minimálnymi úpravami zdrojového kódu prispôsobiť požiadavkám zákazníka. Napríklad samotné otvorenie vstupu môže byť neželané kvôli obavám o bezpečnosť, v takom prípade by systém len monitoroval vstup, napríklad za účelom kontrolovania dochádzky zamestnancov alebo študentov. Avšak tento systém bol zamýšľaný hlavne ako vstupný systém so správou pomocou webového servera. Tento cieľ sa určite podarilo splniť nakoľko kamera umiestnená pri vstupe nepotrebuje rozpoznávať tváre zo vzdialenosti väčšej ako 1,6 m a neznáme tváre sú ignorované pokiaľ nezazvonia, takže bodom záujmu sú hlavne známe tváre s prístupom, ktorým systém udelí prístup prakticky okamžite. Tento systém máme nainštalovaný na rodinnom dome na otvorenie exteriérovej bránky, kde nemusíme mať obavy o prvotriedne zabezpečenie. Systém nám poskytuje bez kľúčový vstup na pozemok čo sa ukázalo ako veľmi praktické.

Treba dodať, že je nutné zvážiť ponechanie funkcionality zabezpečujúcej odfotenie neznámej osoby, nakoľko sa jedná o citlivé osobné informácie, ktoré môžu byť ďalej použité na biometrické spracovanie. Ak by prevádzkovateľ nemal dostatočne podloženú legálnosť takéhoto systému jednalo by sa o porušenie Všeobecného nariadenia o ochrane údajov GDPR. Túto funkcionalitu je možné veľmi jednoducho odstrániť zo zdrojového kódu.

7 Zhrnutie

Týmto môžeme skonštatovať, že hlavný cieľ stanovený v úvode bol naozaj splnený. Výsledkom práce je inteligentný vstupný systém pracujúci na základe biometrických informácií získaných z tváre spolu s webovým serverom umožňujúci jednoduchú správu systému. Podarilo sa nám vytvoriť spoľahlivý program, ktorý má aj značne širšie uplatnenie než bolo pôvodne zamýšľané. V práci je na rozpoznávanie tvárí implementovaná knižnica Dlib a webový server zabezpečuje framework Django. Počas tvorby tohto systému sme sa stretli s viacerými problémami, ktoré sa však v závere podarilo úspešne vyriešiť a vytvoriť plne funkčný celok umožňujúci pohodlný bez kľúčový prístup do objektov s nižšou ostrahou. Z týchto dôvodov sme sa rozhodli tento systém nasadiť na rodinnom dome, kde sa jeho činnosť ukázala byť značne nápomocná a praktická. Webový server nám umožňuje pohodlnú správu systému z akéhokoľvek miesta v rámci lokálnej siete LAN, poprípade aj z celého sveta náležitou konfiguráciou routera.

Resume

Hereby we can conclude that the main goal set in the beginning has been fulfilled. The result of our project is a smart entrance system giving access on the basis of biometrical informations gathered from a face, featuring a web server providing an easy and integrated management of the system. We succeeded in building a reliable system, which has profoundly broader field of application than was originally intended. We implemented Dlib machine learning library for facial recognition tasks and Django web framework for building a web server. During the period of making this software we encountered many problems which in turn we successfully solved and succeeded in building seamlessly working entity providing convenient access to objects with lower safety measures. For these reasons we decided to implement this system in a private property (family house), where its operation turned out to be helpful and practical. The web server provides us with means of managing the system from anywhere within our LAN or, optionally after appropriate configuration of the router from the whole world.

Zoznam použitej literatúry

1. KAPLAN, Andreas; HAENLEIN, Michael. Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence. *Business Horizons*. 2018, roč. 62. Dostupné z DOI: 10.1016/j.bushor.2018.08.004.

2. MARR, Bernard. *5 Important Artificial Intelligence Predictions (For 2019) Everyone Should Read* [online]. Forbes Magazine, 2018 [cit. 2019-02-13]. Dostupné z: https://www.forbes.com/sites/bernardmarr/2018/12/03/5-important-artificial-intelligence-predictions-for-2019-everyone-should-read/#3b5f376a319f.

3. VASEEKARAN, Gowthamy. *Machine Learning: Supervised Learning vs Unsupervised Learning* [online]. medium, 2018 [cit. 2019-02-11]. Dostupné z: https://medium.com/@gowthamy/machine-learning-supervised-learning-vs-unsupervised-learning-f1658e12a780.

4. *Build with AI* [online] [cit. 2019-02-11]. Dostupné z: https://deepai.org/machine-learning-glossary-and-terms/neural-network.

5. KARUNAKARAN, Dhanoop. *Deep learning series 1: Intro to deep learning – Intro to Artificial Intelligence – Medium* [online]. medium, 2018 [cit. 2019-02-11]. Dostupné z: https://medium.com/intro-to-artificial-intelligence/deep-learning-series-1-intro-to-deep-learning-abb1780ee20.

6. LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. *Nature*. Máj 2015, roč. 521, č. 7553, s. 436–444. ISSN 0028-0836. Dostupné z DOI: 10.1038/nature14539.

7. ZHANG; REN; SUN; JIAN. *Deep Residual Learning for Image Recognition* [online]. American Physical Society, 2015 [cit. 2019-02-11]. Dostupné z: https://arxiv.org/abs/1512.03385.

8. JAY, Prakash; JAY, Prakash. *Understanding and Implementing Architectures of ResNet and ResNeXt for state-of-the-art Image...* [online]. medium, 2018 [cit. 2019-02-11]. Dostupné z: https://medium.com/@14prakash/understanding-

and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624.

9. KING, Davis. *High Quality Face Recognition with Deep Metric Learning* [online]. 1970 [cit. 2019-02-11]. Dostupné z: http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html.

10. GOUR, Rinu. *AI-Python Computer Vision Tutorial with OpenCV* [online]. medium, 2018 [cit. 2019-02-11]. Dostupné z: https://medium.com/@rinu.gour123/ai-python-computer-vision-tutorial-with-opencv-b7f86c3c6a1a.

11. *How Facial Recognition Systems Work* [online]. HowStuffWorks, 2001 [cit. 2019-02-11]. Dostupné z: https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/facial-recognition.htm.

12. PONTIN, Mark Williams; PONTIN, Mark Williams. *Better Face-Recognition Software* [online]. MIT Technology Review, 2012 [cit. 2019-02-11]. Dostupné z: https://www.technologyreview.com/s/407976/better-face-recognition-software/.

13. *Django introduction* [online] [cit. 2019-02-11]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction.

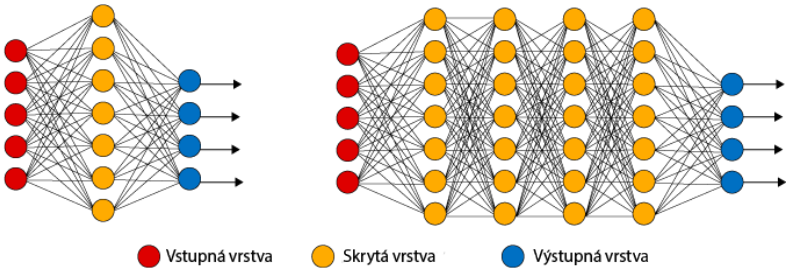
14. GEITGEY, Adam. *Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning* [online]. medium, 2016 [cit. 2019-02-11]. Dostupné z: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78.

15. *Affine transformation* [online]. Wikimedia Foundation, 2019 [cit. 2019-02-11]. Dostupné z: https://en.wikipedia.org/wiki/Affine_transformation.

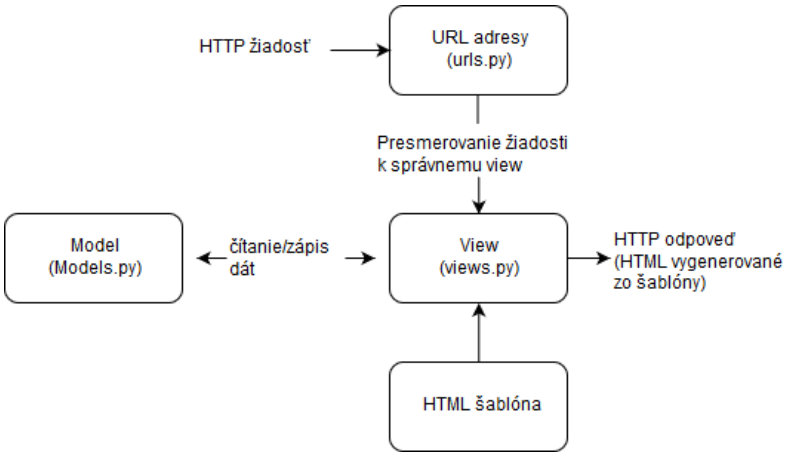
16. SOUKUPOVÁ, Tereza; CECH, Jan. Real-Time Eye Blink Detection using Facial Landmarks. In: [online]. 2016 [cit. 2019-02-11]. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/64839/F3-DP-2016-Soukupova-Tereza-SOUKUPOVA_DP_2016.pdf.

17. UMOFFIA, Richard. *How To Send Web Push Notifications from Django Applications* [online]. DigitalOcean, 2018 [cit. 2019-02-11]. Dostupné z: https://www.digitalocean.com/community/tutorials/how-to-send-web-push-notifications-from-django-applications.

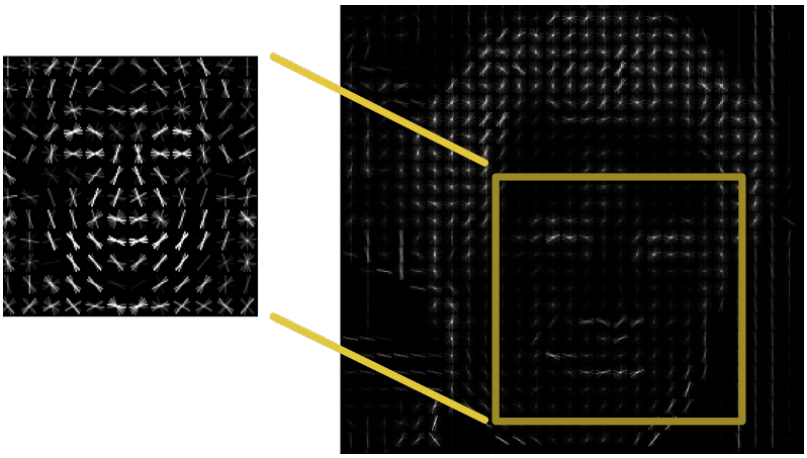
A Príloha - ilustrácie



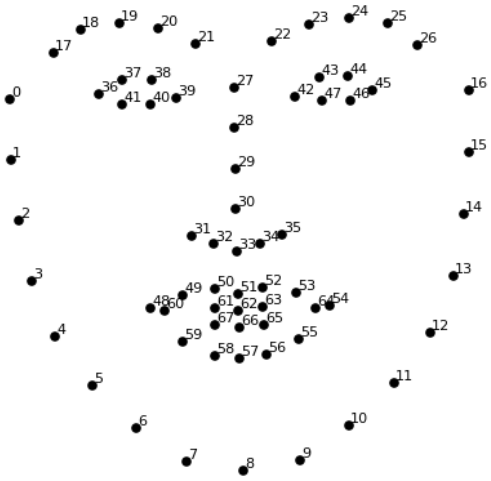
Obr. 1: Každý kruh predstavuje umelý neurón a šípky predstavujú jeho vzťah k ostatným neurónom. Na ľavej strane je jednoduchá neurónová sieť s jednou vrstvou, na pravej strane je hĺbková neurónová sieť s viacerými vrstvami, zdroj: https://towardsdatascience.com/mnist-vs-mnist-how-i-was-able-to-speed-up-my-deep-learning-11c0787e6935, (upravené).



Obr. 2: Znázornenie architektúry model-view-template, zdroj: developer.mozilla.org [13], (upravené).



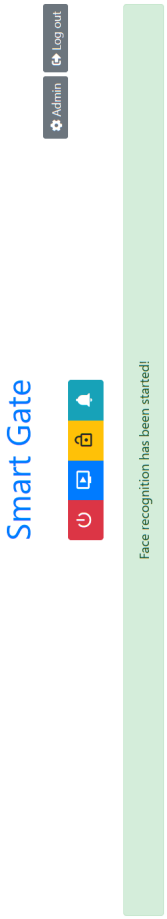
Obr. 3: Naľavo sa nachádza HOG vzor vytvorený z tréningových dát, napravo sa nachádza HOG vzor vytvorený z jedného obrázku podrobeného detekcií, autor: Adam Geitgey [14], (upravené).



Obr. 4: Umiestnenie 68 bodov na tvári, autor: Adam Geitgey [14], (upravené).

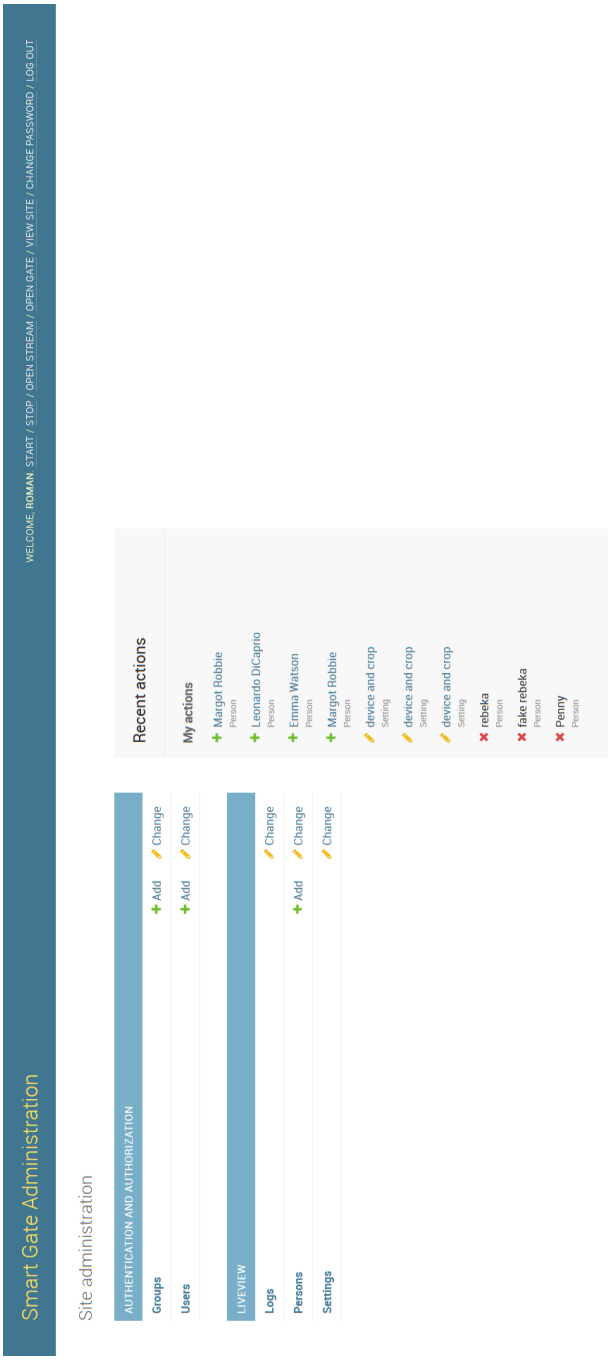
B Príloha - snímky obrazovky

Snímka obrazovky hlavnej stránky



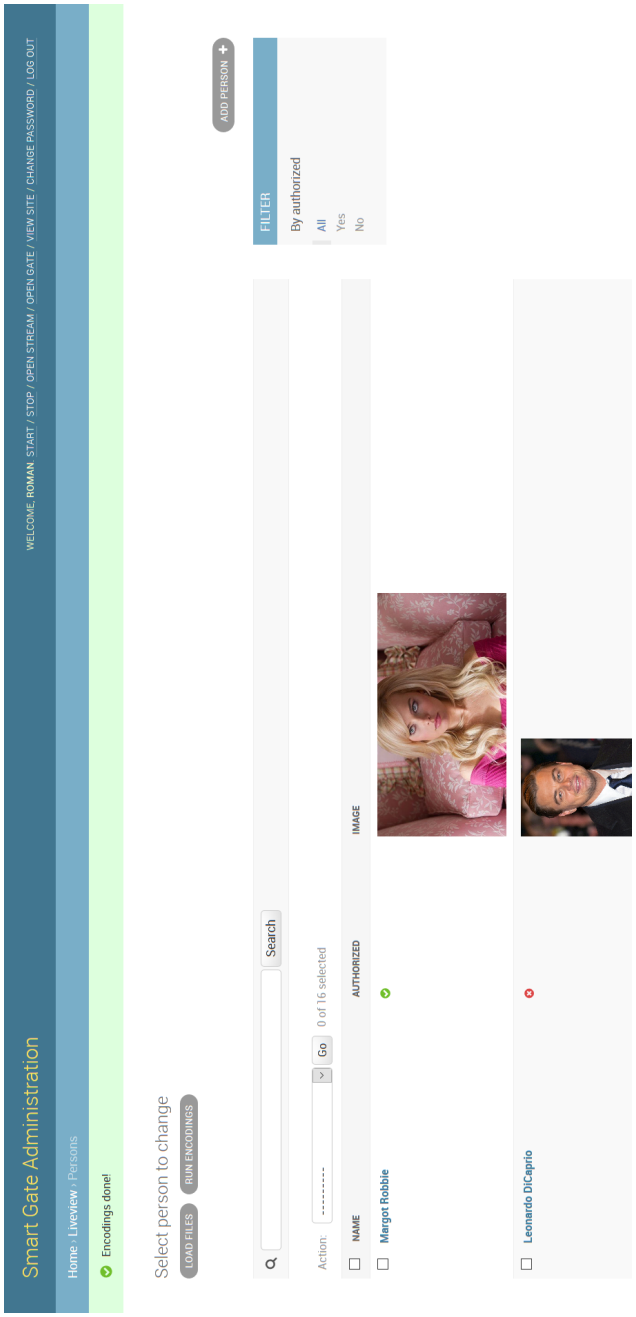
Obr. 1: Screenshot domovskej stránky. Tlačidlá v strede slúžia na spustenie/zastavenie rozpoznávania tvárí, otvorenie streamu, otvorenie bránky a povolovanie odosielania push notifikácií. V hornej pravej časti sú tlačidlá na prechod do administračného rozhrania a odhlásenie.

Snímka obrazovky administračného rozhrania



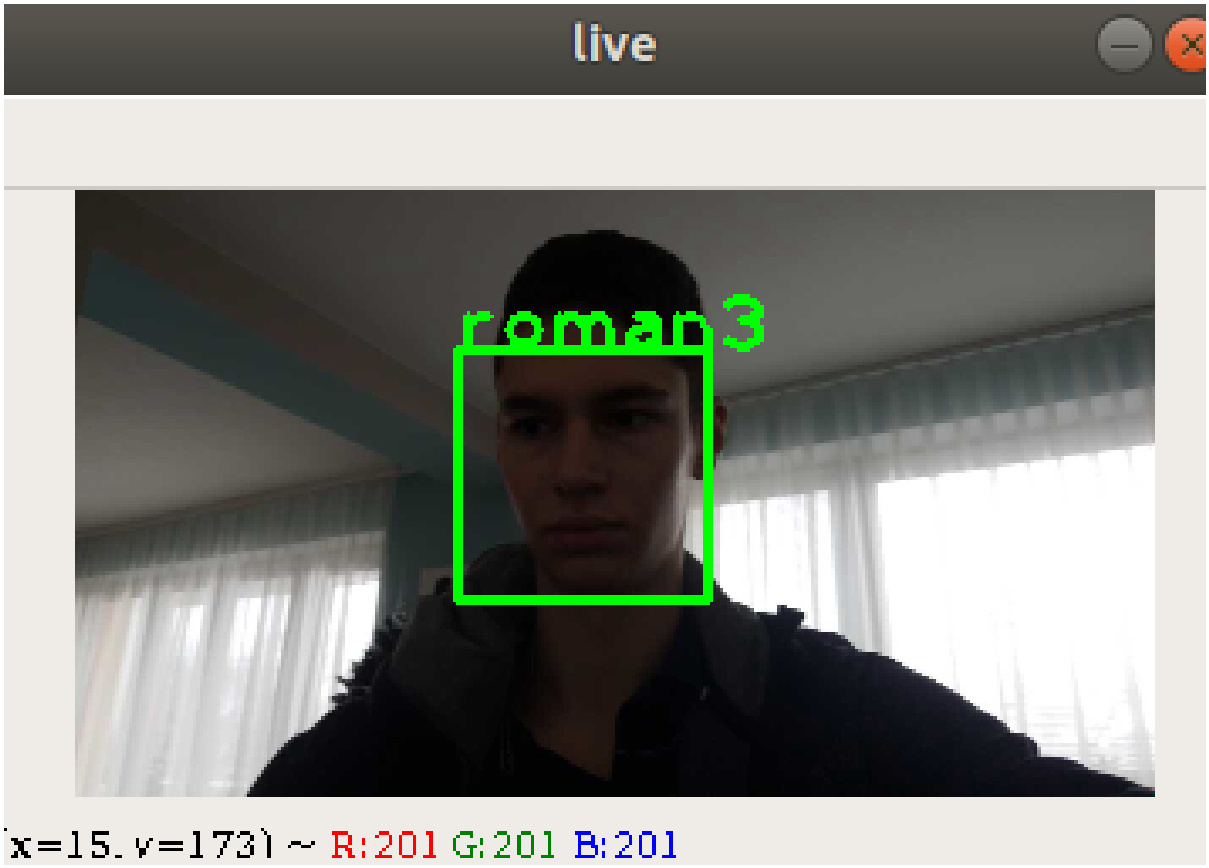
Obr. 2: Screenshot hlavnej stránky administračného rozhrania.

Snímka obrazovky administračného rozhrania



Obr. 3: Screenshot sekcie Persons administračného rozhrania po vykonaní spracovania známych tvárí.

Osoba v protisvetle



Obr. 4: Screenshot správne identifikovanej osoby v protisvetle, tvár je veľmi tmavá, no napriek tomu systém pridelil prístup.

C Príloha - ukážka kódu

Prevedenie funkcie *process()*

Zdrojový kód funkcie *process()* zo súboru *API/FaceRecAPI.py*[1]

```
#funkcia z triedy FaceRecognition(), stara sa o rozpoznavanie tvari.
 def process(self):
   #list so stitkami
   labels = []
   #premenna obsahujuca obrazok z fronty
   image = self.frameQ.get()
   #zmensenie velkosti snimky
   frame = self.resize_img(image, fx=self.resize_factor,
       fy=self.resize_factor)
   #zavolanie detektora tvari
   faces = self.detect(frame)
   #ak sa na snimke nachadzaju tvare:
   if faces is not None:
      #zavolanie funkcie, ktora extrahuje crty tvari
      landmarks = self.find_landmarks(frame, faces)
      #cyklus vykonajuci akcie na kazdej tvari
      for y in range(0, len(faces)):
         #nakreslenie ohranicujucich ramcekov
         rect = self.dlib2opencv(faces[y])
         self.draw(frame, rect)
         #porovnanie tvari zavolanim funkcie porovnvania, ktora vola
             funkciu na
         spracovanie deskriporov
         comparisons = (self.compare(self.descriptors,
             self.descriptor(frame, landmarks[y]))).tolist()
         #cyklus, pre kazde porovnanie tvare so vsetkymi tvarami v
             databaze
         for comparison in comparisons:
            if comparison <= 0.55:
               #zisti stitok patriaci tvari
               label = comparisons.index(comparison)
```

[1]Kompletný zdrojový kód systému sa nachádza na platforme GitHub: https://github.com/maromcik/FaceRecNet

```
                #skusi zistit meno podla stitku
                try:
                    self.PrintText(frame, self.names[label], rect[0],
                        rect[1])
                #ak sa take meno v databaze nenajde, vypise ze tvar uz
                    neexistuje
                except IndexError:
                    print("Person does not exist anymore, you have most
                        likely forgotten to load files")
            #do listu labels vlozi tuple skladajuci sa z rozhodnutia
                detektora zmurkania, ci osoba zmurkla a stitku danej
                tvare
            labels.append(self.blink_detector(landmarks[y], label))
        #ak su vsetky porovnania viac ako urcena hodnota
        if all(i >= 0.55 for i in comparisons):
            #vsetky tvare na snimke su vyhodnotene ako nezname
            self.PrintText(frame, "unknown", rect[0], rect[1])
            labels.append(None)
    #snimka sa naspat prevedie do farebnej
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    #vrati sa list stitkov a snimka
    return labels, frame
```

Funkcia *start()*

Zdrojový kód funkcie *start()* zo súboru *LiveView/views.py*[2]

```
#dekorator vynucujuci prihlasenie, ak pouzivatel nie je prihlaseny
    je presmerovany na prihlasovaciu stranku
@login_required(login_url='/accounts/login')
#funkcia spustajuca rozpoznavanie tvari
def start(request):
    #ziska aktualneho pouzivatela z objektu ziadosti request
    user = request.user
    #ak rozpoznavanie tvari nie je spustene, spusti sa
    #nastavi premennu message na hlasku informujucu o stave.
    #premenna status sluzi na zmenu farby hlasky.
    if rec_threads.startrecognition():
        message = "Face recognition is already running."
        status = 1
    else:
        message = "Face recognition has been started!"
        status = 0
    #premenna running sluzi na zmenu farby tlacidla spustenia.
    try:
        running = rec_threads.facerecognition_thread.isAlive()
    except AttributeError:
        running = False
    #premennej subscription je priradena informacia z databazy o
        stave prijimania push notifikacii pre daneho pouzivatela,
        sablona podla neho zobrazi bud ikonku notifikacie alebo
        preciarknutu ikonku notifikacie
    subscription = Subscriber.objects.get(user=user).subscription
    #Django celu sablonu naplni premennymi a posle ako http odpoved
    return HttpResponse(render(request, 'LiveView/LiveView.html',
        {'message': message, 'running': running, 'subscription':
        subscription, 'status': status}))
```

[2]Kompletný zdrojový kód systému sa nachádza na platforme GitHub: https://github.com/maromcik/FaceRecNet