

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analýza řidiče za pomocí sférických kamer

Driver Analysis Using Spherical Cameras

2020

Michal Falát

Zadání diplomové práce

Student: **Bc. Michal Falát**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Analýza řidiče za pomocí sférických kamer
Driver Analysis Using Spherical Cameras

Jazyk vypracování: slovenština

Zásady pro vypracování:

Analýza objektů s pomocí sférické kamery je v posledních letech hodně rozvíjené téma. Aplikace tohoto druhu může být použita například v oblasti samoříditelných vozidel.

1. Popište základní pojmy a metody v oblasti analýzy aktivit lidských postav v obrazech.
2. Seznamte se s volně dostupnými knihovnami a popište jaké možnosti nabízí v této oblasti (například OpenPose, wrnchAI, TensorFlow, OpenCV).
3. S pomocí knihoven vytvořte vybraný analyzátor řidiče za pomocí sférických kamer.
4. Experimentálně ověřte funkčnost, přesnost a rychlosť navrženého řešení na dostupných datasetech.
5. Své závěry rádně zdokumentujte v textu práce.

Seznam doporučené odborné literatury:

- [1] Cao, Zhe et al. "Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
- [2] V. Belagiannis and A. Zisserman, "Recurrent Human Pose Estimation," 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), Washington, DC, 2017, pp. 468-475.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Radovan Fusek, Ph.D.**

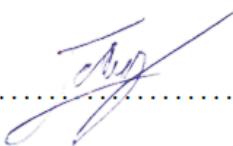
Datum zadání: 01.09.2019

Datum odevzdání: 30.04.2020



Prehlasujem, že som túto diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave 15. mája 2020

.....


Rád by som podakoval môjmu vedúcemu práce Ing. Radovanovi Fusekovi za pomoc a ochotu pri vypracovaní diplomovej práce.

Abstrakt

Hlavnou témou diplomovej práce je analýza správania vodiča v aute pomocou detekcie obrazu. Táto práca je rozdelená do viacerých samostatných častí, ktoré na seba postupne nadväzujú. Prvá časť spočíva v preskúmaní existujúcich riešení na detekciu postáv vo vozidlách. Druhá časť je zameraná na detekciu a spracovanie obrazu pomocou sférických kamier, ktoré sú potrebné k nasnímaniu celého tela vodiča. Posledná kapitola je venovaná samotnému programu na vyhodnotenie správania vodiča vo vozidle v rôznych jazdných situáciach a porovnanie jednotlivých metód na detekciu. V závere je rozpísaná problematika takéhoto spôsobu detekcie a možnosti použitia v reálnych systémoch vozidla.

Kľúčové slová: Detekcia obrazu, analýza vodiča, analýza ľudskej tváre, detekcia ľudí, vodič, detekcia vo vozidle, sférická kamera, Openpose, Tensorflow

Abstract

Main focus of this Diploma thesis is analysis of driver's behaviour in vehicle using image detection. This thesis is divided into few parts, which are connected to each other. The first part is about research of existing solutions for detecting driver's pose in vehicle. The second part is focused on an detection and processing video from spherical cameras, which are needed to capture entire driver's body. Next part is about final program for the analysing driver in vehicle and the comparison of the various methods for detection. The last chapter is overview about problematics of this type of detection and usage in real vehicle systems.

Keywords: Image detection, driver's analysis, analysis of human face, pedestrian detection, driver, image detection on vehicles, spherical camera, Openpose, Tensorflow

Obsah

Zoznam použitých skratiek a symbolov	7
Zoznam obrázkov	8
Zoznam tabuliek	10
Zoznam výpisov zdrojového kódu	11
1 Úvod	12
2 Detekcia vodiča vo vozidle	13
3 Detekcia a analýza ľudského tela v obrazoch	15
3.1 Príznakové rozpoznávanie	15
3.2 OpenPose	19
3.3 TF Pose Estimation	22
3.4 AlphaPose	24
3.5 Ostatné metódy	25
4 Využitie sférických kamier na detekciu obrazu	27
5 Vlastná implementácia	32
5.1 Požiadavky a návrh programu	33
5.2 Použité technológie	34
5.3 Vytvorenie a spracovanie videa	35
5.4 Detekcia vodiča	37
5.5 Analýza správania vodiča	45
5.6 Detekcia orientácie hlavy	54
5.7 Možnosti vylepšenia detektie	61
5.8 Zhrnutie výsledkov programu	61
6 Záver	63
Literatúra	64
Prílohy	67

Zoznam použitých skratiek a symbolov

2D	– 2-dimensional
3D	– 3-dimensional
CLI	– Command line interface
CNN	– Convolutional neural network
CNTK	– Microsoft cognitive toolkit
CPU	– Central processing unit
DLT	– Direct linear transformation
DOF	– Degree of freedom
FOV	– Field of view
FPS	– Frames per second
GB	– Gigabyte
GPU	– Graphical processing unit
HOG	– Histogram oriented gradients
IR	– Infra red
LED	– Light emitting diode
MP	– Megapixel
NMS	– Non maximum suppression
OpenCV	– Open source computer vision
PAF	– Part affinity fields
PX	– Pixel
RCNN	– Region convolutional neural network
SPPE	– Single-person pose estimator
SSTN	– Symmetric spatial transformer network
TF	– Tensorflow
VR	– Virtual reality

Zoznam obrázkov

1	BMW driving assistant - umiestnenie kamery na snímanie vodiča. [1]	14
2	Výpočet integrálneho obrazu - vstupný obraz (a), integrálny obraz (b)	15
3	Haar - dvoj-obdlžnikové príznaky (A, B), troj-obdlžnikové príznaky (C) a štvorobdlžnikové príznaky(D) [2]	16
4	Haar - detekcia tváre v slabých svetelných podmienkach	17
5	HOG - séria krokov [3]	18
6	HOG - vstupný obraz (a), normalizácia gradientu (b), orientácia gradientu (c), rozdelenie do buniek (d), vypočítané histogramy (e). [4]	18
7	OpenPose - odhad viacerých osôb v reálnom čase pomocou polí affinity filtra. [5] .	19
8	COCO - označenie častí tela v COCO datasete. [6]	20
9	Mapa pravdepodobnosti - vstupný obraz (a), mapa pravdepodobnosti výskytu končatiny postavy (b) [7]	23
10	Problém redundantnej detektie s použitím metódy bounding box [8]	24
11	WrnchAI - vstupný obraz (a), výsledok spracovania na platforme WrnchCloud(b)	26
12	Model sférickej projekcie. [9]	27
13	GoPro Omni - séria synchronizovaných kamier pre zachytenie sférickej fotografie. [10]	29
14	Znázornenie sférických projekcií - kubické zobrazenie (a), ekvidistantné zobrazenie (b)	30
15	Rovnaká sférická fotografia v rôznych zobrazeniach - kubické zobrazenie (a), ekvidistantné zobrazenie (b)	31
16	Porovnanie výstupu z kamier (ISO 1600) - GoPro Fusion (a), Ricoh Theta V (b)	36
17	Natívny výstup metódy OpenPose - normálne svetelné podmienky (a), zhoršené svetelné podmienky (b), nevyhovujúce svetelné podmienky (c).	38
18	Natívny výstup metódy TF pose - normálne svetelné podmienky (a), zhoršené svetelné podmienky (b), nevyhovujúce svetelné podmienky (c).	40
19	Minimálny rozdiel detektie postavy v rôznych situáciach - používanie riadiacej páky (a), obsluha klimatizácie alebo rádia (b).	46
20	Ukážka trénovacieho datasetu - držanie volantu (STEERING)	47
21	Ukážka trénovacieho datasetu - používanie riadiacej páky (SHIFTING)	47
22	Ukážka trénovacieho datasetu - nesprávna pozícia (WRONG)	47
23	Ukážka fungovania redukčnej vrstvy typu <i>MaxPooling2D</i>	49
24	Vizualizácia modelu siete pomocou nástroja Netron [11]	50
25	Výsledok detektie neurónovej siete	51
26	Ukážka hodnôt neurónovej siete - bez filtrácie(a), s použitou filtráciou (b)	52
27	Detekcia tváre pomocou landmarkov. Označenie bodov (a), detekcia na reálnej tvári vodiča (b)	55

28	Detekcia tváre pomocou landmarkov. Znázornenie referenčných bodov tváre v 3D [12] (a), detekcia na reálnej tvári vodiča (b), výber referenčných bodov v 2D (c)	56
29	Výsledok detektie orientácie tváre - pohľad vpravo (a), pohľad dolu (b), pohľad na vozovku (c), pohľad vľavo (d)	59
30	Orientácia hlavy vodiča s pomocnými čiarami	60
31	Výsledná detekcia pohyby hlavy a vyhodnotenie správania vodiča	62

Zoznam tabuliek

1	Technické parametre sférických kamier	35
2	Úspešnosť detekcie OpenPose - slabé svetelné podmienky, model COCO	42
3	Úspešnosť detekcie OpenPose, slabé svetelné podmienky, model MPI	42
4	Úspešnosť detekcie TF Pose Estimation, dataset <i>mobilenet_thin</i> , rozlíšenie trénovacieho vstupu 432×368 pixelov	43
5	Úspešnosť detekcie TF Pose Estimation, dataset <i>mobilenet_thin</i> , rozlíšenie trénovacieho vstupu 216×184 pixelov	43
6	Úspešnosť detekcie TF Pose Estimation, dataset <i>mobilenet_thin</i> , rozlíšenie trénovacieho vstupu 192×144 pixelov	44
7	Počet trénovacích obrázkov pre jednotlivé triedy	47
8	Úspešnosť vyhodnotenia správania pomocou neurónovej siete	53
9	Zoznam parametrov programu	67

Zoznam výpisov zdrojového kódu

1	Množina párov končatín v datasete COCO	21
2	Mapovanie častí ľudského tela OpenPose a TF Pose Estimation	41
3	Vrstvy neurónovej siete	49
4	Jednoduchý filter na odstránenie náhodných hodnôt s použitím prahovej hodnoty	52
5	Výpočet matice kamery	58
6	rotačného a pozičného vektora	59
7	Spustenie programu	67

1 Úvod

V dnešnom modernom svete sú autá takmer každodennou súčasťou života ľudí. Mnohokrát sa nikto nezamýšla nad ich bezpečnosťou, ktorá je v prípade zrážky klúčová. V súčasnosti pri jazde autom vodičovi asistuje veľké množstvo systémov, ktoré zvyšujú bezpečnosť posádky, ale aj ostatných účastníkov cestnej premávky. Aj keď tieto systémy ešte stále nedokážu vodiča úplne nahradíť, dokážu mu výrazným spôsobom pomôcť napríklad v krízových situáciach. Výhodou takýchto systémov je ich rýchlejší reakčný čas oproti človeku. Takéto systémy spočívajú v použití rôznych snímačov alebo kamier, ktoré aktívne sledujú okolie ale aj interiér vozidla. Vďaka takýmto moderným technickým riešeniam je možné predísť rôznym častokrát aj smrteľným doopravným nehodám. Výrobcovia áut sa čoraz častejšie snažia svoje systémy vylepšovať na čo najvyššiu možnú úroveň a poskytnúť tak vysokú mieru ochrany.

Táto diplomová práca je zameraná hlavne na problematiku analýzy vodiča pomocou detekcie obrazu zo sférickej (360-stupňovej) kamery. V diplomovej práci je postupne rozobratá problematika analýzy videa z kamery umiestnej v interiéri vozidla. Vhodným umiestnením kamery je možné získať obraz z prednej časti auta, ale aj obraz vodiča sediaceho za volantom. Táto práca sa primárne venuje najmä analýze a spracovaniu videa z interiéru vozidla na zachytenie ľudských aktivít vodiča. Aby bola dosiahnutá čo najväčšia časť tela vodiča, je potrebné mať dostatočne veľký uhol záberu. Bežné kamery majú uhol záberu veľmi nízky, aby dokázal z malej vzdialenosťi zachytiť celý snímaný objekt. Takýto problém sa naskytuje napríklad aj v interiéri vozidla, kde je vzdialosť kamery od snímaného objektu menej ako 1 meter, čo nemusí byť dostatočné na zosnímanie celého tela vodiča. Práve v takejto situácii je vhodné použiť širokouhlú prípadne sférickú kameru. Počas vypracovania práce boli k dispozícii viaceré kamery, s ktorými bolo zhodovených niekoľko desiatok videí v rôznych situáciach. Z takýchto videí sa dokázalo analyzovať a zistiť mnoho dôležitých informácií, ktoré sú spracované v tejto diplomovej práci. Tieto informácie boli zbierané nahrávaním videa sférickými kamerami za rôznych svetelných podmienok a pozícií vodiča. V tejto práci sú taktiež spomenuté problémy takejto detekcie, riešenia vzniknutých problémov, ale aj zhrnutie celkovej problematiky sledovania vodiča vo vozidle. V práci sú tiež zhrnuté ďalšie možnosti vylepšenia detekcie a porovnanie oproti klasickým kamerám.

V nasledujúcich kapitolách je postupne rozobratá problematika snímania ľudských postáv v obrazoch a skúmanie ich aktivít. Pre snímanie postáv existuje viacero metód, ktoré boli následne porovnané medzi sebou. Aby sa dokázala vyhodnotiť správna pozícia vodiča, v programe bola využitá neurónová sieť, ktorá bola trénovaná na vlastnom datasete postavenom na výstupe z kamier.

V súčasnosti taktiež neexistuje veľa podobných riešení, ktoré by sa venovali podobnej problematike či spracovaniu videa zo sférickej kamery a preto sa práca zameriava na túto oblasť. Pri analýze vodiča rovnako neboli nájdené žiadne vhodné datasety z interiéru vozidla snímané sférickou kamerou.

2 Detekcia vodiča vo vozidle

Hlavným zmyslom detektie vodiča je analyzovanie jeho správania za volantom. Pomocou takejto analýzy je možné napríklad odhadnúť, či je vodič unavený, alebo či sa dostatočne venuje okolitej premávke. Detekcia vodiča vo vozidle môže byť vykonaná viacerými spôsobmi. Zariadenie, ktoré sleduje vodičovo správanie môže byť založené na rôznych mechanických alebo digitálnych senzoroch, či kamerách. Aby sa zamedzilo zlyhaniu celého systému a nesprávnemu vyhodnoteniu situácie, je vhodné kombinovať viacero takýchto systémov vo vozidle súčasne. Existujú systémy, ktoré dokážu analyzovať správanie vodiča pomocou reakcie na točenie volantom, alebo rôzne trhavé reakcie vodičovho tela, ktoré sú spôsobené pokročilou fázou mikrospánku. Táto fáza je však už často sprevádzaná ďalšími príznakmi ako zatvorenie očí a krátkodobé nevnímanie okolia, čo môže byť veľmi nebezpečné pre všetkých účastníkov premávky. Okrem toho je možné snímať napríklad vodičove oči pomocou digitálnej kamery. Systém by mohol vyhodnocovať pravidelnosť mrknutia očí a na základe celej trasy odhadnúť úroveň únavy. Aby bol systém dostatočne spoľahlivý mal by mať nasledujúce vlastnosti:

- **Okamžitá detekcia** - každý systém musí byť dostatočne rýchly a schopný zistiť nepozornosť alebo mikrospánok vodiča maximálne behom 1-2 sekúnd. Každá sekunda navyše v rýchlo idúcim vozidle dramaticky zvyšuje riziko vzniku dopravnej nehody.
- **Fungovanie v zhoršených podmienok** - systém by fungovať vo všetkých podmienkach spoľahlivo. Nemal by byť obmedzený zhoršenými svetelnými podmienkami, alebo napríklad vodičovými nepravidelnými pohybmi, ktoré sa môžu podobať na mikrospánok.
- **Upozornenie** - pre úspešné fungovanie systému je požadujúce, aby dokázal nielen detektovať problém s vodičom, ale aj účinne zabrániť následkom vznikutej situácie. To je možné vykonať napríklad hlasným zvukovým upozornením, prípadne spomalením až zastavením vozidla v prípade väčšieho problému s vodičovým vedomím.

Samotná detekcia obrazu v uzavretom priestore sa od bežnej detektie v základných princípoch nelísi. Obmedzenia nastávajú najmä nevhodnou pozíciou alebo natočením kamery, ktorá je umiestnená vo vozidle. Ak je kamera nesprávne umiestnená, alebo má nedostatočný uhol záberu, nemusí byť zdetektovaný celý snímaný objekt, čo môže viesť k chybe pri jeho detektii. Autori v práci na analýze vodiča [13], kde sa zameriavalí na oblasť tváre vodiča použili kamery umiestnené v oblasti stredu palubnej dosky. Tým dosiahli takmer priame natočenie kamery na vodiča bez toho, aby ho táto kamera výraznejšie obmedzovala vodiča vo výhľade. Kedže táto práca bola zameraná primárne na detekciu očí a úst, nebolo potrebné používať kamery s veryšokým uhlom záberu. Celý proces snímania však neboli spracovaný v reálnom čase, ale až po presune a spracovaní nahratých videí z kamery. Aby sa dal použiť takýto systém aj v reálnej premávke, bolo by potrebné aby dokázal spracovať video a vyhodnocovať vodičovo správanie

v reálnom čase. Jedným z takýchto riešení by mohlo byť napríklad použitie malej priemyselnej kamery spolu s počítačom.

Zariadenie na detekciu ospalosti vodiča ponúka aj celosvetová firma Bosch. Výrobca si však uvedomoval všetky nedostatky použitia kamery na snímanie a preto sa rozhodol vymyslieť detekciu spánku iným spôsobom. Na volante je pripojené špeciálne zariadenie, ktoré sníma otáčanie volantom. Následne rozpoznáva zmeny v priebehu dlhých cest a tým pádom aj únavu vodiča. Ich detekcia ospalosti vodiča je založená na algoritme, ktorý začína zaznamenávať správanie vodiča pri začiatí jazdy. Typickými znakmi klesajúcej koncentrácie sú fázy, počas ktorých vodič nevenuje dostatočnú pozornosť vedeniu vozidla v kombinácii s miernymi, ale rýchlymi a prudkými pohybmi volantu, aby udržal vozidlo na ceste. Takéto riešenie však môže byť limitujúce, pretože nedokáže na mikrospánok zareagovať okamžite. Z tohto dôvodu ho firma Bosch predáva hlavne ako pomocný systém na odhalenie únavy vodiča.

Mnoho podobných riešení zameraných na snímanie aktivity vodiča sa snažia zaviesť aj výrobcovia automobilov. Títo výrobcovia si veľmi dobre uvedomujú, že veľká časť dopravných nehôd býva spôsobená mikrospánkom, alebo len nepozornosťou vodiča. Príkladom môže byť výrobca automobilov značky BMW, ktorý vo svojich vozidlách ponúka ako príplatkovú výbavu kameru na snímanie správania vodiča. Táto kamera dokáže reagovať napríklad na únavu alebo zatvorenie očí vodiča. Vo všeobecnosti tento systém dokáže dokonca reagovať aj na to, že vodič má otočenú hlavu a nesleduje premávku, čo môže viesť k nebezpečnej situácii. V každom takomto prípade je vodič upozorený zvukovým znamením. Kamera je umiestnená v prístrojovej doske na mieste za volantom, odkiaľ je na tvár vodiča priamy výhľad (obr. 1). Okrem jej vhodného umiestnenia je navyše vybavená aj IR LED diódami, vďaka ktorým kamera funguje aj za znížených svetelných podmienok a v noci, kedy sa vyskytuje najväčší počet výskytu mikrospánku u vodičov všetkých vozidiel. Vďaka takému riešeniu sa dokáže predísť mnohým nebezpečným situáciám a dopravným nehodám. Výrobcovia však neuvádzajú žiadne oficiálne štatistiky o úspešnosti týchto systémov na snímanie tváre vodiča. V nasledujúcej kapitole sú popísané metódy na detekciu a snímanie ľudského tela v interieri vozidla. Takúto detekciu je možné následne využiť na analýzu chovania vodiča a predikovanie jeho správania alebo rôznych nebezpečných situácií.



Obr. 1: BMW driving assistant - umiestnenie kamery na snímanie vodiča. [1]

3 Detekcia a analýza ľudského tela v obrazoch

História detekcie postáv v obrazoch siaha až do polovice 20. storočia. Mnoho inžinierov vielo obrovský potenciál detekcie obrazu napríklad v oblastiach medicíny, priemyslu, dopravy a mnohých ďalších oblastiach. S nárastom technických možností postupne rástla motivácia využiť detekciu obrazu aj v praxi. Jeden z prvých vedeckých článkov v oblasti spracovania obrazu [14] rozoberal napríklad jednoduchú analýzu obrazu a spracovanie obrazov s dostupnými prostriedkami. Postupom času sa však počítacová technika vylepšovala a bolo možné pracovať na vývoji metód pre analýzu a detekciu objektov v obrazoch. Na detekciu chodcov alebo iných ľubovoľných objektov existuje mnoho prístupov. Veľkým fenoménom v posledných rokoch sa stali neurónové siete. Okrem neurónových sietí však stále existujú aj tradičné metódy, ktoré fungujú aj bez trénovacích dát. V práci boli využívané metódy Haar a HOG, ktoré sa radia medzi najpoužívanejšie tradičné metódy a je im venované podkapitola 3.1.

Každý obraz sa skladá z pixelov. Analýza obrazu však nespočíva v prehľadávaní jednotlivých pixelov, ale v hľadaní celých objektov v obraze. Tieto objekty je možné určovať do samostatných tried. Triedy určujú, aký druh objektu sa v obraze nachádza. Môže to byť napríklad chodec, vozidlo, dopravná značka a podobne. Aby bolo možné tieto objekty (napr. ľudí) v obraze identifikovať, je potrebné použiť spoloahlivý a rýchly spôsob detekcie.

3.1 Príznakové rozpoznávanie

Príznaky typu Haar

Metódou Haar je možné detektovať rôzne triedy objektov. Túto metódu je možné využiť napríklad na snímanie tváre vodiča alebo iných osôb. Táto metóda bola popísaná autormi Viola a Jones [15] v roku 2001. Medzi jej hlavné výhody patrí vysoká rýchlosť a spoloahlivá detekcia a vysoká nezávislosť na intenzite osvetlenia, čo môže byť klúčový faktor pri použití detekcie v interiéri vozidla. Vo všeobecnosti je tento detektor rozdelený do 4 samostatných časťí: Výpočet integrálneho obrazu, výpočet Haar príznakov, výber príznakov a kaskádový klasifikátor.

Výpočet integrálneho obrazu sa robí prevedením vstupného obrazu na integrálny obraz (obr. 2). Výpočet pre konkrétné súradnice (x, y) spočíva v súčte hodnôt jasu vľavo a nad súradnicami (x, y) . Výpočet je znázornený v rovnici 1.

1	1	1
1	1	1
1	1	1

(a)

1	2	3
2	4	6
3	6	9

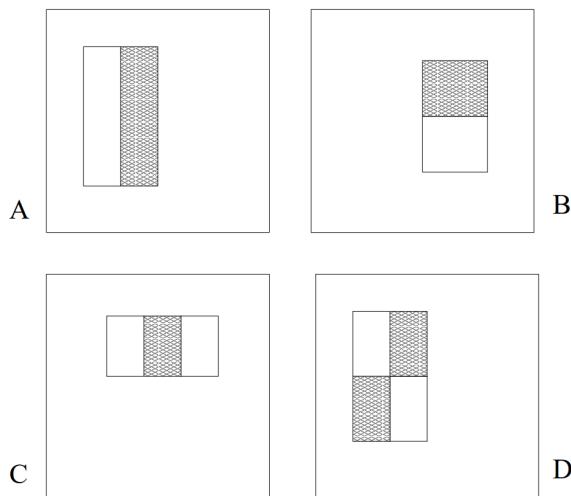
(b)

Obr. 2: Výpočet integrálneho obrazu - vstupný obraz (a), integrálny obraz (b)

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'), \quad (1)$$

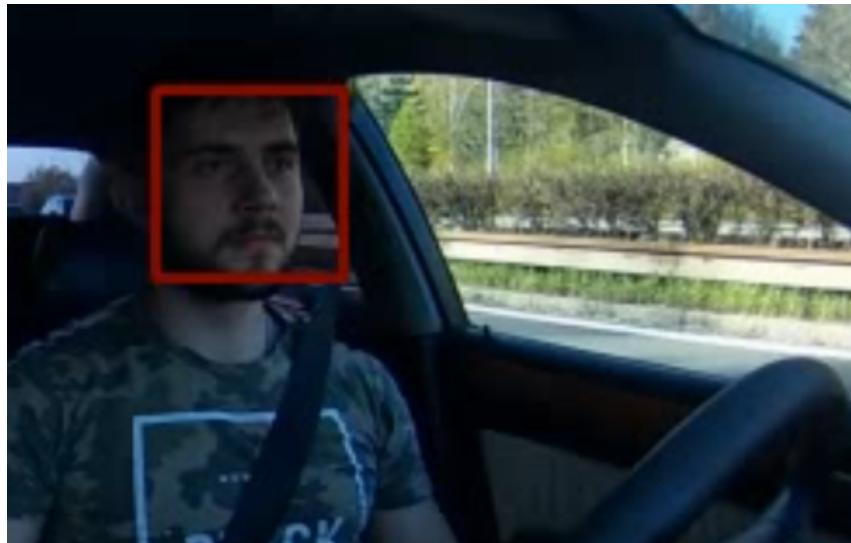
Metóda funguje na porovnávaní celých blokov pixelov. Tieto bloky (častokrát nazývané aj zhluky) môžu mať rôzne tvary, veľkosť a natočenie. Tieto bloky môžu nadobúdať rôzne tvary ale vo všeobecnosti sa používajú 3 hlavné typy príznakov:

- **Dvoj-obdlížnikové** (angl. two-rectangle) - porovnávajú sumu pixlov v obdlížnikových oblastiach, ktoré sa nachádzajú vedľa seba, vodorovne, alebo zvislo.
- **Troj-obdlížnikové** (angl. three-rectangle) - porovnávajú sumu obdlížnikových oblastí, ktoré sa nachádzajú po oboch stranách aktuálnej oblasti a sumu aktuálnej oblasti.
- **Štvor-obdlížnikové** (angl. four-rectangle) - počítajú rozdiel medzi dvoma aktuálnymi obdlížnikovými oblastami, ktoré sa dotýkajú svojimi rohmi a obdlížnikovými oblastami medzi nimi.



Obr. 3: Haar - dvoj-obdlížnikové príznaky (A, B), troj-obdlížnikové príznaky (C) a štvor-obdlížnikové príznaky(D) [2]

Znázornenie jednotlivých typov príznakov je znázornené na obrázku 3. Jednotlivé príznaky môžu byť použité pre rôzne typy obrázkov. Efektivita klesá pri použití príznaku na celý obraz. Vhodným riešením je preto skombinovať viacero príznakov. Na výber správnych efektívnych príznakov sa používajú špeciálne algoritmy. Jedným z najpoužívanejších algoritmov pre zvýšenie efektivity výberu príznakov je AdaBoost [16], ktorý vytvoril profesor Yoav Freund. Jedná sa o klasifikačný algoritmus, ktorý je schopný vytvoriť dostatočne silný klasifikátor z kombinácií viacerých slabších klasifikátorov.



Obr. 4: Haar - detekcia tváre v slabých svetelných podmienkach

Príznaky HOG

S nápadom vylepšiť detekciu objektov použitím príznakov prišli v roku 2005 Navneet Dalal a Bill Triggs [3], kde postupne vyskúšali niekoľko typov deskriptorov. V práci taktiež podrobne rozobrali možnosti a spôsoby ako správne určiť parametre ich detekčnej metódy pre správne fungovanie detekcie jednotlivých tried. Hlavná myšlienka tejto metódy spočíva vo využití súčtu gradientov, ktorý sa počíta pre každú časť obrazu, ktorý je rozdelený do viacerých častí. Metóda je vhodná napríklad pre detekciu ľudských postáv, ktoré majú charakteristický tvar. Táto metóda je rozdelená do niekolkých samostatných krokov (obr. 5):

- **Úprava obrazu** - v tomto kroku je potrebné v obraze upraviť kontrast a jas, ktoré by mohli spôsobovať problémy v nasledujúcich krokoch. Okrem tejto úpravy je možné obraz upraviť napríklad gamma filtrom.
- **Výpočet gradientov** - veľkosť gradientov sa počíta na základe vstupného obrazu a masky. Masky, ktoré sa používajú v tomto kroku sú $[-1, 0, 1]$ alebo $[-1, 0, 1]^T$. Gradienty je nutné vypočítať v obidvoch osách, čím sa získa I_x a I_y . Po získaní gradientov je potrebné vypočítať veľkosť gradientov $m(x,y)$ a ich smer $\theta(x,y)$:

$$m(x,y) = \sqrt{I_x^2 + I_y^2} \quad (2)$$

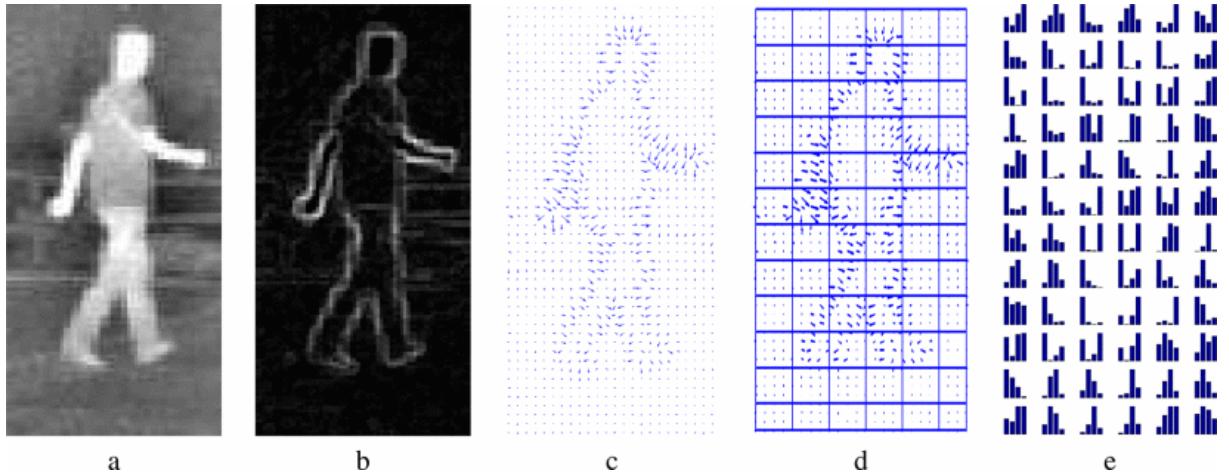
$$\theta(x,y) = \left(\frac{I_y}{I_x} \right) \quad (3)$$

- **Normalizácia** - pre správne fungovanie je potrebné obraz normalizovať, aby sa minimalizovali rozdiely medzi jednotlivými bunkami. Tento krok spočíva v skladaní viacerých buniek, čím následne vznikajú bloky.
- **Deskriptor** - je vytvorený zo vstupného obrazu do jednotlivých blokov. Jednotlivé bloky sa posúvajú a prekryvajú o daný počet pixelov. Výsledok deskriptoru je odovzdaný klasifikátoru, ktorý následne určuje do akej triedy objekt patrí. Jeden z často používaných klasifikátorov je support vector machine (SVM), ktorý napríklad používali autori vo svojej práci na efektívnu detekciu chodcov [17].



Obr. 5: HOG - séria krokov [3]

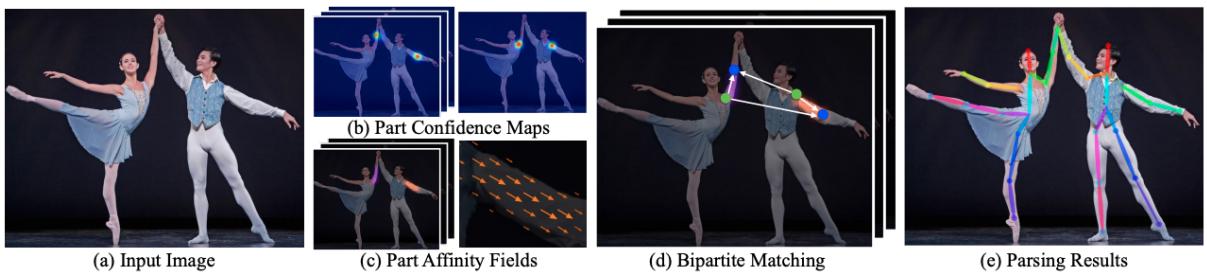
Výstupom tejto metódy je množina histogramov pre jednotlivé bloky. Histogram predstavuje grafické rozloženie intenzity jasu vstupného obrazu. Pri niektorých špecifických obrazoch je potrebné histogram najskôr vyrovnať. Tento krok je potrebný najmä pri obrazoch, ktoré sú príliš tmavé alebo príliš svetlé. Pomocou vyrovnania (angl. equalization) histogramu je možné zvýšiť kontrast obrazu. Jednotlivé kroky znázornené na vstupnom obraze chodca je možné vidieť na obrázku 6.



Obr. 6: HOG - vstupný obraz (a), normalizácia gradientu (b), orientácia gradientu (c), rozdelenie do buniek (d), vypočítané histogramy (e). [4]

3.2 OpenPose

OpenPose [5] je jedným z najpoužívanejších frameworkov na detekciu postáv v obrazoch. Prvýkrát bol uvedený verejnosti už v roku 2016. Detekcia ľudského postoja predstavuje hlavný problém s lokalizáciou častí ľudského tela ako sú ramená, lakte a členky zo vstupného obrázka alebo videa. Vo väčšine dnešných aplikácií detekcie postáv v reálnom svete sa vyžaduje vysoký stupeň presnosti, ako aj spracovanie v reálnom čase. OpenPose, ktorý bol vyvinutý výskumníkmi na univerzite Carnegie Mellon University, možno považovať za najmodernejší prístup pri detekcii ľudských v reálnom čase. Jedná sa o open-source projekt, ktorého zdrojové kódy sú verejne dostupné [18].



Obr. 7: OpenPose - odhad viacerých osôb v reálnom čase pomocou polí afinitnosti. [5]

Samotný framework je veľmi detailne vysvetlený a dobre zdokumentovaný. OpenPose bol pôvodne napísaný v jazyku C++ a používal nástroj Caffe [19]. Postupom času však autori vytvorili aj nadstavbu pre jazyk Python, s ktorým sa rozšírili možnosti jeho využitia medzi ostatnými programátormi. Základná myšlienka detekcie pomocou OpenPose sa skladá z viacerých krokov:

- **Spracovanie vstupného obrazu** - vstupný obrázok (obr. 7a) privádza ako vstup do dvojvetvovej viacstupňovej konvolučnej neurónovej siete (CNN). Dve vetvy znamenajú, že CNN produkuje dva rôzne výstupy z jedného vstupného obrazu. Viacstupňové znamená, že sieť je v každej fáze naskladaná jedna na druhú. Tento krok je analogický jednoduchému zväčšeniu hĺbky neurónovej siete s cieľom zachytiť podstatnejšie výstupy smerom k poslednému stupňom.
- **Spracovanie v dvoch vetvách** - prvá vetva predpovedá mapy dôveryhodnosti (obr. 7b) rôznych častí tela, ako je pravé oko, ľavé oko, pravé lakte a podobne. Druhá vetva zobrazená modrou farbou predpovedá afinitné polia (obr. 7c), čo predstavuje stupeň asociácie medzi rôznymi časťami tela.
- **Viacfázové spracovanie** - v prvej fáze sieť vytvorí počiatočnú sadu detekčných máp spoľahlivosti S a množinu polí afinitných častí L . Potom v každej nasledujúcej fáze predpovede z obidvoch vetiev v predchádzajúcej fáze, spolu s pôvodnými obrazovými znakmi F , sú zreťazené a použité na vytvorenie podrobnejších predpovedí. Pri implementácii OpenPose sa posledná fáza t zvolí ako číslo 6.

3.2.1 Mapa spoľahlivosti

Prvá vetva v neurónovej sieti OpenPose vytvára sadu máp spoľahlivosti S (rovnica 4). V podstate sa jedná o tabuľku, v ktorej je každej časti tela z datasetu priradená miera spoľahlivosti v rozsahu 0 až 1.

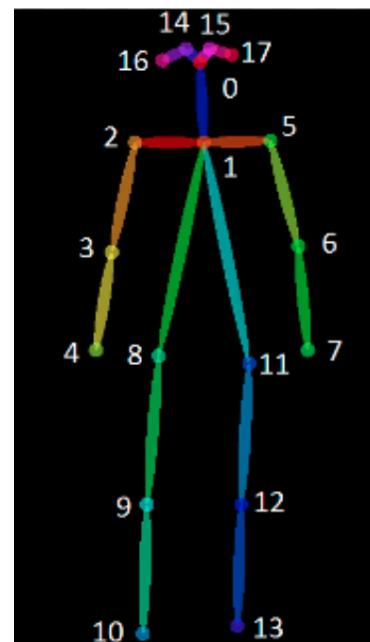
$$S = (S_1, S_2, S_3 \dots S_j) \quad (4)$$

$$S \in \mathbb{R}^{w \times h},$$

$$j \in \{1, J\}, \text{kde } J \text{ je počet všetkých častí tela}$$

Počet častí tela závisí od množiny datasetov, s ktorými je program OpenPose trénovaný. Pokiaľ ide napríklad o súbor datasetu COCO [20], $J = 19$, pretože existuje 18 rôznych kľúčových bodov tela + 1 pozadie. Obrázok 8 znázorňuje rôzne časti tela s prideleným identifikátorom pre súbor údajov COCO. Pre model trénovaný s dátovým súborom COCO bude sada S obsahovať prvky $S_1, S_2, S_3, \dots, S_{19}$. V tomto príklade je zrejmé, že prvok S_1 zodpovedá mape spoľahlivosti pre kľúčový bod s číslom 0, ktorý zodpovedá nosu (obr. 8).

Ako jednoduchý príklad môže poslúžiť predstava, že celý obraz má šírku a výšku 5 pixelov, čo vedie k vytvoreniu mapy spoľahlivosti o veľkosti 5×5 . Vo vstupnom obrázku sa nachádza iba jedna tvár. Preto pre mapu spoľahlivosti S_1 (zodpovedajúca za detekciu nosa) je možné vidieť hodnoty s vysokou spoľahlivosťou iba v oblasti, kde sa nos nachádza.



Obr. 8: COCO - označenie častí tela v COCO datasete. [6]

3.2.2 Part affinity fields (PAF)

Druhá vetva neurónovej siete vytvára množinu čiastkových afinitných polí L (rovnica 5).

$$\begin{aligned} L &= (L_1, L_2, L_3 \dots L_c) \\ L &\in \mathbb{R}^{w \times h \times 2} \\ c &\in \{1, C\}, \text{kde } C \text{ je počet všetkých končatín} \end{aligned} \tag{5}$$

Celkový počet končatín a párov závisí od datasetu, s ktorým je OpenPose trénovaný. Kvôli prehľadnosti sa uvádzajú dvojice častí tela ako končatiny, napriek tomu, že niektoré páry častí tela nie sú v skutočnosti končatinami (napríklad oko-nos, ucho-oko atď.). Pre dataset COCO je počet párov končatín, $C = 19$. Môžeme si predstaviť, že každý prvok v množine L je mapa veľkosti $w \times h$, kde každá bunka obsahuje 2D vektor predstavujúci smer párových prvkov. Napríklad na obrázku 8 je možné vidieť, že párs častí tela pozostáva z pravého ramena k pravému lakťu. Schéma potom ukazuje smerový vektor, ktorý ukazuje z pravého ramena na pravý lakť. Celý zoznam párov končatín je znázornený vo výpise 1.

```
COCO_PAIRS = [(1, 2), (1, 5), (2, 3), (3, 4), (5, 6), (6, 7), (1, 8), (8, 9),
(9, 10), (1, 11), (11, 12), (12, 13), (1, 0), (0, 14), (14, 16), (0, 15),
(15, 17), (2, 16), (5, 17)]
```

Výpis 1: Množina párov končatín v datasete COCO

Okrem Datasetu COCO Dokáže OpenPose pracovať aj s mnohými ďalšími datasetmi. OpenPose bol vyvíjaný a trénovaný napríklad s datasetmi MPI [21], BODY_25 alebo BODY_25b. Datasetsy sa líšia vo veľkosti, rýchlosťi, ale napríklad aj v presnosti samotnej detekcie. Jednotlivé datasetsy majú medzi sebou nasledujúce rozdiely:

- **COCO** - starší dataset, na ktorom bol OpenPose pôvodne vyvíjaný. Postupne sa však nahradzuje novými a modernejšími datasetmi. Jeho výhodou je, že vyžaduje menej pamäte na GPU (schopnosť pracovať s 2 GB GPU a predvoleným nastavením) a pri režime CPU pracuje rýchlejšie oproti novšiemu BODY_25.
- **BODY_25** - jedná sa o novší dataset, ktorý je rýchlejší, presnejší a obsahuje ďalšie trénovacie dátá k časťiam tela, ktoré nie sú obsiahnuté v COCO datasete ako napríklad chodidlá. Jeho nevýhodou sú hlavne vysoké hardwarové nároky.
- **MPI** - je určený pre vývoj, kde sa vyžaduje použiť štruktúru datasetu MPI. Je tiež pomalší oproti BODY_25 a oveľa menej presný.

3.3 TF Pose Estimation

Tento framework na detekciu postáv je implementovaný pomocou knižnice Tensorflow. Poskytuje tiež niekoľko variantov, ktoré sa odlišujú najmä v zmenách pre spracovanie v reálnom čase na CPU alebo zariadení s nízkou spotrebou. Z tohto dôvodu ho je možné používať napríklad na mobilných zariadeniach alebo internetových prehliadačoch použitím knižnice tensorflow.js. Tensorflow Pose Estimation používa na detekciu vlastný model s názvom PoseNet. PoseNet sa dá použiť na odhad jednej pozície alebo viacerých pozícii v obraze súčasne. To znamená, že existuje verzia algoritmu, ktorý dokáže detektovať iba jednu osobu v obraze a druhá verzia, ktorá dokáže zistiť viac osôb v obraze. Hlavnou výhodou použitím detekcie jednej osoby je rýchlejšie spracovanie a nižší výpočtový výkon. Podstatnou nevýhodou však je, že vyžaduje iba jeden objekt prítomný na obrázku. Pri súčasnej pozícii viacerých osôb v obraze tento algoritmus nedokáže zdetektovať správne ani jednu osobu. Je preto potrebné zamyslieť sa hned na začiatku, kolko osôb sa reálne môže v obraze nachádzať. Hlavná myšlienka tohto algoritmu sa skladá z viacerých krokov podobne ako pri knižnici OpenPose.

- **Detekcia pozície** - na najvyšej úrovni modelu PoseNet sa vráti objekt, ktorá obsahuje zoznam kľúčových bodov a skóre spoľahlivosti pre každú detektovanú osobu.
- **Výpočet spoľahlivosti pozície** - skóre spoľahlivosti určuje celkovú dôveru v odhadovaní pozície. Je v rozsahu od 0 až 1. Môže sa použiť na skrytie pozícii, ktoré sa nepovažujú za dostatočne výrazné.
- **Výpočet kľúčových bodov** - odhadované časti tela osoby, ako napríklad nos, pravé ucho, ľavé koleno, pravá noha atď. Obsahuje pozíciu a spoľahlivosť kľúčového bodu. PoseNet štandardne zistuje 17 kľúčových bodov.
- **Poloha kľúčového bodu** - pozostáva z 2D súradníc v pôvodnom vstupnom obrázku, kde bol zistený kľúčový bod.

Model PoseNet je nezávislý na veľkosti vstupného obrazu. To znamená, že môže predikovať polohy pozícii v rovnakom rozlíšení ako pôvodný obrázok bez ohľadu na to, či je obraz zmenšený. PoseNet môže byť nakonfigurovaný tak, aby mal vyššiu presnosť na úkor rýchlosťi detekcie nastavením výstupného kroku. Výstupný krok určuje, do akej miery zmenšujeme výstup vzhľadom na veľkosť vstupného obrázka. To ovplyvňuje veľkosť jednotlivých vrstiev a výstupy modelu. Čím vyšší je výstupný krok, tým menší je počet vrstiev v sieti a výstupoch a tým aj ich presnosť. V základnej implementácii môže mať výstupný krok hodnoty 8, 16 alebo 32. Inými slovami, výstupný krok 32 bude mať za následok najrýchlejší výkon, ale najnižšiu presnosť, zatiaľ čo 8 bude mať najvyššiu presnosť, ale najpomalší čas detekcie.

Výstup detekcie

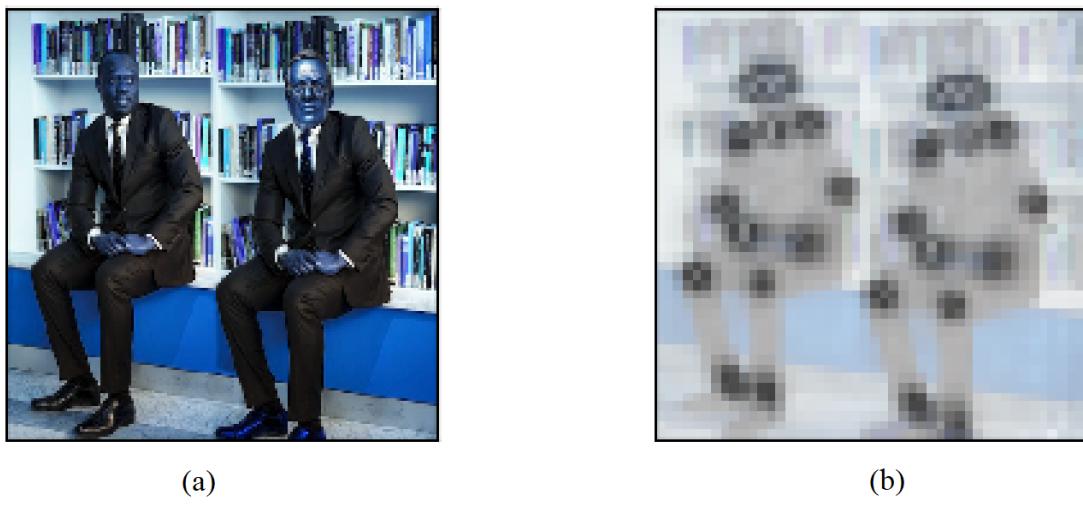
Ked PoseNet spracováva obraz, v skutočnosti vytvára mapu pravdepodobnosti (angl. heatmap) spolu s offsetovými vektormi, ktoré je možné dekódovať, aby sa v obraze našli oblasti s vysokou spoľahlivosťou. Veľkou výhodou pri detekcii postáv v obraze cez Tensorflow je teda to, že spolu s vektormi výstupného modelu dostávame aj štrukturované dátá pravdepodobnosti jednotlivých častí postavy, ktoré môžeme využiť na ďalšie spracovanie, alebo znázorniť pre lepšiu predstavu (obr. 9).

Mapa pravdepodobnosti

Každá pozícia v tejto mape pravdepodobnosti má určité skóre spoľahlivosti. Tento údaj vyjadruje pravdepodobnosť, že v danom umiestnení existuje určitá časť daného typu. Dá sa to považovať za rozdelenie pôvodného obrázka do mriežky 15×15 , kde skóre v mape pravdepodobnosti poskytuje konkrétnu hodnotu pravdepodobnosti, že každý kľúčový bod existuje v každom štvorci mriežky.

Výstupný vektor

Každý výstupný vektor je 3D vektor, ktorý má velkosť $w \times h \times 34$. Číslo 34 je dvojnásobok kľúčových bodov (2×17). Mapy pravdepodobnosti sú iba aproximáciou toho, kde sa skutočne kľúčové body nachádzajú, pričom výstupné vektory zodpovedajú svojou polohou bodom tejto mapy a používajú sa na predpovedanie presnej polohy jednotlivých častí ľudského tela. Prvých 17 rezov výstupného vektora obsahuje x-ovú súradnicu vektora a posledných 17 rezov y-ovú súradnicu. Veľkosti výstupného vektora sú v rovnakej mierke ako pôvodný vstupný obrázok.

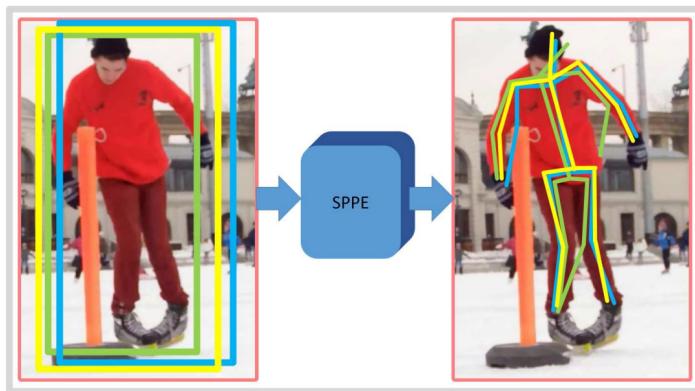


Obr. 9: Mapa pravdepodobnosti - vstupný obraz (a), mapa pravdepodobnosti výskytu končatiny postavy (b) [7]

3.4 AlphaPose

AlphaPose [8] je zameraný na detekciu ľudských postáv s veľmi vysokou presnosťou. Prvé začiatky tohto frameworku siahajú do roku 2017. Tento framework pracuje na princípe používania ohraničovacích boxov (angl. bounding box). Pre efektívne fungovanie algoritmu je použitý najmodernejší detektor objektov. Autori práce použili natrénovaný model Faster RCNN [22] a model pre detekciu postáv v obraze [23], ktorý je zameraný na SPPE detekciu. Primárna myšlienka práce spočíva v riešení dvoch hlavných problémov, ktoré pri takejto detekcii vznikajú:

- **Problém lokalizačnej chyby** - v skutočnosti je tento model dosť náchylný na chyby spočívajúce v použití ohraničovacích boxov. Aj v prípadoch, keď sú ohraničovacie rámčeky považované za správne a majú dostatočne vysokú spoľahlivosť ($I_oU > 0,5$), zistené ľudské pozície môžu byť stále nesprávne.
- **Redundantná detekcia** - model detekuje pozíciu pre každý ohraničovací box, čo vo výsledku vedie k duplicitnej detekcii rovnakej osoby (obr. 10)



Obr. 10: Problém redundantnej detekcie s použitím metódy bounding box [8]

Na vyriešenie uvedených problémov je použitý model na regionálnu detekciu viacerých postáv (RMPE). Vďaka tomu framework vylepšuje výkon algoritmov na odhadovanie ľudských postojov založených na modeli SPPE. Autori práce tiež navrhli novú symetrickú sieť pre priestorovú detekciu (SSTN), ktorá je pripojená k SPPE na extrakciu jednotlivej osoby z oblasti nepresného ohraničovacieho boxu. Na optimalizáciu tejto siete je zavedená ďalšia paralelná vetva SPPE. Na riešenie problému redundantnej detekcie je použitý parametrický NMS, ktorý eliminuje nadbytočné pózy pomocou novej metriky vzdialenosť a porovnanie podobnosti pózy. Prístup založený na údajoch je aplikovaný na optimalizáciu parametrov metriky vzdialenosť. RMPE je navrhnutý veľmi všeobecne a práve vďaka tomu je použiteľný pre rôzne ľudské detektory a ďalšie knižnice, ktoré pracujú na princípe single-person detection. AlphaPose používa dataset MPII, s ktorým prekonáva najmodernejšie metódy ako OpenPose. Tento model a zdrojové kódy [24] sú verejne dostupné a určené primárne pre vedu a výskum.

3.5 Ostatné metódy

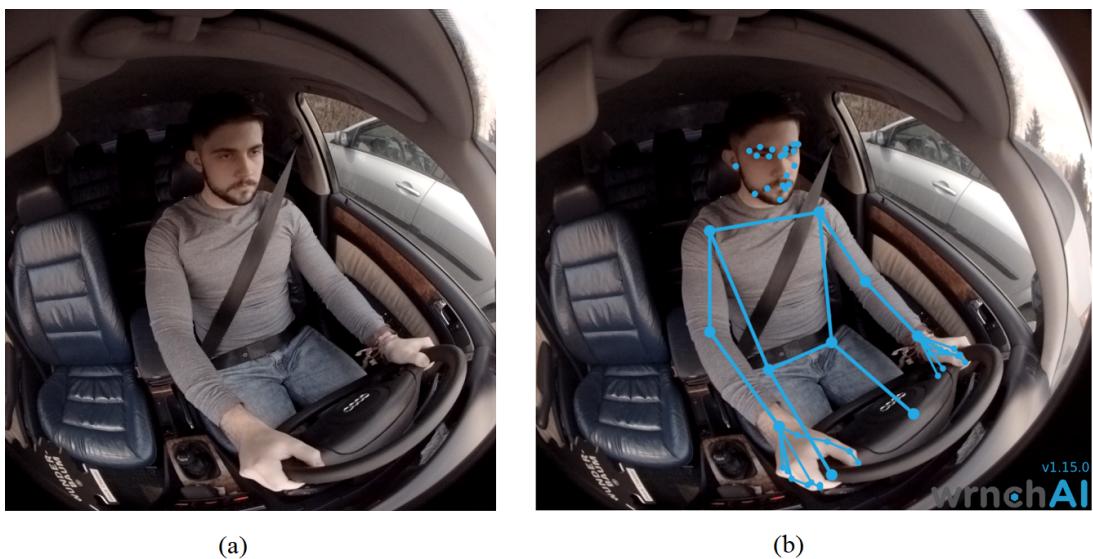
Okrem najpoužívanejších metód ako OpenPose, TF Pose Estimation, či AlphaPose existuje aj mnoho ďalších riešení. Tieto riešenia často vychádzajú zo základného princípu detekcie postáv, ktoré používajú aj tieto hlavné knižnice. Hlavný rozdiel je však použitie upravenej neurónovej siete alebo zmena rôznych parametrov. Vďaka takejto úprave je možné dosiahnuť v špecifických prípadoch vyššiu úspešnosť a presnosť detekcie alebo dokonca znížiť výpočtový čas potrebný na detekciu a spracovanie. Tieto riešenia sú však často obmedzované najmä chýbajúcou dokumentáciou, nedostatočnou implementáciou vo viacerých jazykoch, alebo nemožnosťou jednoduchej zmeny konfigurácie. Okrem voľne dostupných riešení existuje aj mnoho projektov, ktoré nemajú verejné zdrojové kódy a slúžia len pre komerčné použitie, čo môže mnoho používateľov zo začiatku odradiť. Jedným z takýchto projektov je napríklad WrnchAI, ktorému je venovaná nasledujúca podkapitola.

WrnchAI

WrnchAI [25] je platforma, ktorá je zameraná na detekciu postáv v obrazoch vytvorená v roku 2014. S myšlienkou využiť umelú inteligenciu a založiť projekt pre detekciu postáv prišiel Paul Kruszewski. Napriek tomu, že táto platforma existuje už niekoľko rokov a v niektorých článkoch [26] dokázala získať lepší čas detekcie oproti iným metódam, nestala sa veľmi populárnym nástrojom medzi vývojármami. Jedným z hlavných dôvodov je to, že knižnica nie je voľne dostupná pre vývojárov a jedná sa o platený projekt bez voľného prístupu k zdrojovému kódu. Súčasná cenová politika základnej verzie sa pohybuje na úrovni \$500 za mesiac používania s možnosťou vyskúšania trial verzie na prvý mesiac zadarmo. Trial verzia však funguje v obmedzenom režime bez prístupu ku všetkým platformám. Pre bežného používateľa sa táto cena môže zdať privyšoká, avšak vzhľadom na jeho široké možnosti využitia v rôznych oblastiach sa jedná o adekvátnu cenu. WrnchAI našiel svoje uplatnenie možnosťou využitia na rôznych typoch zariadení, ktoré sa prispôsobujú konkrétnym podmienkam. V súčasnosti WrnchAI poskytuje 4 hlavné platformy:

- **wrnchPC** - umožňuje zachytiť ľudský pohyb neobmedzeného množstva ľudí, bez toho, aby boli použité drahé senzory či kamery. WrnchPC je postavený na základoch umelej inteligencie tak, aby sa ľahko integroval do všetkých aplikácií. Použité sú modely hĺbkového učenia, aby bolo zabezpečené spoloahlivé zachytenie ľudského pohybu pre všetky činnosti a v akomkoľvek prostredí.
- **wrnchCloud** - detekuje ľudský postoj na vzdialenom serveri. WrnchCloud je nákladovo veľmi efektívne riešenie, pretože nepotrebuje žiadne vysoké hardwarové nároky okrem bežného počítača s pripojením na internet. Celý proces detekcie prebieha na serveri, čo odlaďuje zákazníka od nákupu drahých grafických kariet a iného hardwaru. Toto riešenie je škálovateľné s možnosťou spracovania viacerých vstupov súčasne. Využitie cloudového riešenia zároveň pomáha držať krok moderných trendov v internetových technológiách.

- **wrnchMobile** - umožňuje využiť mobilné zariadenia na vytvorenie programu snímania postáv v reálnom čase. Taktiež dokáže spolupracovať s mobilnými operačnými systémami ako Android alebo IOS. Jeho základným princípom je efektívne využitie výpočtového výkonu na malých prenosných zariadeniach ako sú napríklad mobilné telefóny. Tento druh platformy taktiež dokáže pracovať aj s ďalšími pokročilými nástrojmi ako je napríklad rozšírená realita. S pomocou rozšírenej reality dokáže wrnchMobile snímanú postavu detektovať a následne zobrazovať napríklad do VR okuliarov.
- **wrnchEmbedded** - poskytuje možnosť detektovať postavy na rôznych automatizovaných zariadeniach ako napríklad roboty, samoriadiace vozidlá, aby mohli v reálnom čase vidieť a interagovať s ľuďmi a ich pohybom. WrnchEmbedded pomáha zariadeniam predvídať ľudské správanie a najlepšie reagovať vo všetkých situáciách. Ich program je optimalizovaný najmä na rôzne priemyselné platformy a počítače, vďaka ktorým by mala byť ich integrácia s platformou Wrnch veľmi jednoduchá. V súčasnosti je však táto časť platformy stále v aktívnom vývoji a momentálne nie je dostupná.

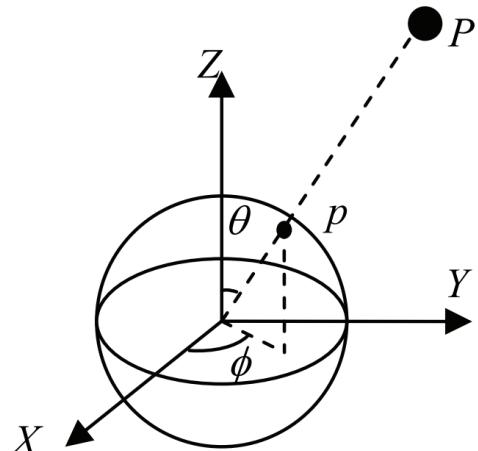


Obr. 11: WrnchAI - vstupný obraz (a), výsledok spracovania na platforme WrnchCloud(b)

Na obrázku 11 je znázornený výsledok metódy cloubovej platformy WrnchAI. Táto platforma funguje dostatočne spoľahlivo aj pri atypických polohách ľudského tela, ako je napríklad sedavá poloha vodiča za volantom. WrnchAI okrem polohy tela poskytuje aj detekciu kľúčových bodov tváre a kľúčové body rúk a prstov. Oproti ostatným knižniciam a frameworkom je možnosť použiť WrnchAI ako komplexné riešenie na detekciu celého ľudského tela, čo napríklad pri OpenPose nie je bez použitia ďalších rozšírení možné. V skúšobnej trial verzii taktiež nie je možné odstrániť vodoznak firmy, ktorý sa nachádza v pravom dolnom rohu výstupného obrázka.

4 Využitie sférických kamier na detekciu obrazu

Súčasný trh ponúka veľké množstvo kamier rôznych typov. Bežne používaným typom sú kamery, ktoré majú uhol záberu v priemere od 70° do 90° . Problém takejto kamery je zachytenie celého snímaného objektu z malej vzdialenosťi. Pri použití v interiéri vozidla by tak postava vodiča nemusela byť zachytená celá, pretože interiér bežného vozidla nie je veľmi rozsiahly. Kamera by musela byť umiestnená od vodiča v dostatočnej vzdialnosti, čo môže predstavovať problém. Okrem týchto typov však existujú aj kamery, ktoré majú uhol záberu podstatne vyšší, či dokonca umožňujú aj panoramatický záber celého svojho okolia. Na snímanie interiéru vozidla je vhodné použiť napríklad sférickú kameru, s ktorou je postava vodiča úplne zachytená. Výstup zo sférickej kamery je reprezentovaný 2D snímkou, ktorá zachytáva obraz z celého svojho okolia. Toto zobrazenie sa dá popísať modelom sférického obrazu (obr. 12). Predpoklad spočíva v tom, že existuje priestorová guľa a bod P , ktorý sa nachádza v priestore. Priesecník povrchu gule s čiarou spájajúcou bod p a stred gule je premietanie bodu p do gule. Takýmto spôsobom sa získá sférický obraz premietaním všetkých viditeľných bodov do gule. Toto sa nazýva sférická projekcia. Záber sférickej projekcie je možné vyjadriť dvomi uhlami. Prvý uhol vyjadruje záber v horizontálnom smere $\phi \in [-180^\circ, +180^\circ]$ a druhý uhol vyjadruje záber vo vertikálnom smere $\theta \in [-90^\circ, +90^\circ]$.



Obr. 12: Model sférickej projekcie. [9]

Zhotovenie takejto sférickej snímky je technicky náročnejšie ako zhotovenie bežnej fotografie. Takýto záber sa zvyčajne skladá zo sérii viacerých fotiek, ktoré sú snímané viacerými objektívmi. Pri takomto zhotovení fotografie môže dochádzať k určitým problémom, ktoré by nemuseli zaručiť správny výsledok. Pre dosiahnutie požadovanej kvality a zachytenie celého okolia kamery je potrebné splniť určité podmienky. Ideálny sférický model obrazového snímača by mal preto splňať nasledujúce vlastnosti:

- **Úplný zorný uhol** (angl. full FOV) - každá sférická kamera by mala vytvárať obraz z celého svojho okolia bez vynechania miesta v scéne. Ak sa nejaká časť scény vynechá, nejedná sa o úplný sférický záber.
- **Jeden pozorovací zdroj** - snímanie celej scény by malo byť zabezpečené z jedného rovnakého miesta, aby jednotlivé časti scény na seba správne nadväzovali. V prípade, že by scéna bola snímaná z viacerých miest, rekonštrukcia výslednej sférickej snímky by sa nemusela správne vytvoriť. Niektoré objekty v scéne by mohli byť napríklad posunuté alebo roztahnuté, čo by nezodpovedalo reálnemu stavu.
- **Snímanie celej scény v rovnakom čase** - zariadenie na snímanie by malo vyhotoviť snímok celej scény v jednom momente. Táto vlastnosť je potrebná najmä pri snímaní dynamických scén, alebo pohyblivých objektov. Pri jej nedodržaní by sa pohybujúci objekt mohol vo výslednom obraze objaviť viackrát, alebo byť rozmazaný.

Možnosti snímania sférického obrazu

Aby bolo možné získať obraz, ktorý zachytáva celú scénu, je potrebné takýto obraz najskôr nasmínať. Existuje mnoho metód a postupov, ako možno vytvoriť sférickú fotografiu. Každé riešenie má svoje výhody aj nevýhody, ktoré môžu ovplyvniť konečný výber spôsobu vytvárania sférickej fotografie. Tieto spôsoby sa častokrát líšia najmä vo finančnej, či technickej náročnosti. Je potrebné si preto uvedomiť, či výsledná fotografia musí splňať rôzne kritéria a podľa toho prispôsobiť výber. Pri použití takejto kamery v interiéri vozidla sa požaduje najmä kompaktná veľkosť, aby kamera nebránila vodičovi vo výhľade, ale aj čo najmenší počet objektívov, aby nevznikalo veľa chýb spôsobených v spájaní jednotlivých snímok. Medzi najbežnejšie možnosti zhotovenia sférických obrazov je možné zaradiť tieto spôsoby:

- **Použitie bežného fotoaparátu** - vytvorenie sférickej fotografie je možné aj pomocou bežného fotoaparátu, ktorý sa nachádza napríklad v dnešných smartfónoch. Hlavným obmedzením, ktoré toto riešenie prináša je nedostatočný uhol záberu kamery. Aby bolo možné zachytiť celú scénu, je potrebné vytvoriť fotografie celého okolia, ktoré sa následne poskladajú do mozaiky. Toto riešenie je veľmi jednoduché, avšak má obrovské nevýhody. Takýmto snímaním nie je možné zachytiť dynamickú scénu a na výslednej fotografií sa môžu pohybujúce objekty objaviť viackrát. Ďalšou nevýhodou je zdĺhavé vytváranie a spájanie jednotlivých snímok, ktoré nemusia vo výsledku na seba úplne nadväzovať. Takéto riešenie sa často používa napríklad pri fotoní panoramatickej fotky, kedy je pohyb objektov minimálny.
- **Použitie skupiny kamier** - riešenie je založené na použití viacerých bežných fotoaparátov súčasne. Takéto zariadenie dokáže v jednom momente ovládať všetky fotoaparáty a zhotoviť fotografiu, ktorú následne správne spojí do výslednej sférickej fotografie. Takéto

zariadenie je však technicky náročné a samotné fotoaparáty musia byť schopné komunikovať s hlavnou jednotkou zariadenia. Tento faktor sa vo výsledku prejaví najmä vo vysokej cene zariadenia. Okrem ceny je však zariadenie veľké a fažké, čo môže byť pre určité zamerania obmedzujúci faktor. Zariadenie spočívajúce zo skupiny viacerých kamier vytvorila napríklad firma GoPro. Skladá sa zo 6 outdoorových kamier, ktoré sú pripojené ku centrálnnej jednotke. Zariadenie je znázornené na obrázku 13.

- **Použitie všesmerovej kamery** - ďalšou populárnu metódou je snímať široké scény jediným fotoaparátom s použitím zakriveného zrkadla. Tvar zrkadla môže byť napríklad parabolický, hyperbolický alebo eliptický, vďaka čomu poskytuje 360° zorné pole v horizontálnej rovine a viac ako 100° vo vertikálnej rovine. Použitie všesmerovej kamery však neprináša úplný 360° obraz celej scény, pretože nie je možné zachytiť obraz nad šošovkou a obraz pod zakriveným zrkadlom. Z tohto dôvodu môže vzniknúť limitujúci faktor pri výbere.
- **Kombinácia hemisférických kamier** - najpoužívanejšou alternatívou na získanie sférickej fotografie je použitie dvojice širokouhlých objektívov tzv. rybie oko (angl. fish eye). Takýto objektív sa od obyčajného líši najmä svojím obrovským uhlom záberu, ktorý mnohokrát prekonáva hranicu 180° . Samotná sférická kamera sa skladá z dvoch oproti sebe umiestneniach objektívov, ktoré sa následne spracujú do výslednej sférickej fotografie. Hlavnou výhodou takéhoto riešenia je cena, dostatočne kvalitný obraz bez väčších rozdielov medzi jednotlivými snímkami objektívov a rýchle vytvorenie snímky.



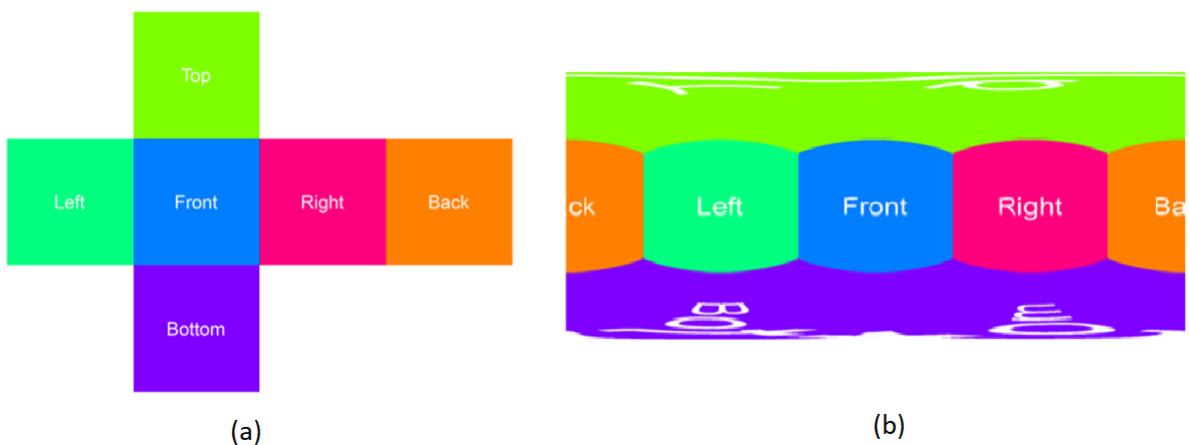
Obr. 13: GoPro Omni - séria synchronizovaných kamier pre zachytenie sférickej fotografie. [10]

Formát sférickej fotografie

Panoramicke projekcie sa používajú na mapovanie úplnej alebo čiastočnej 3D scény na dvojrozmerný povrch. Napríklad cylindrické projekcie sprostredkujú scénu viditeľnú vo všetkých smeroch, s výnimkou priamo nad kamerou a pod kamerou. To znamená, že horná a dolná časť

imaginárneho valca v takýchto obrázkoch chýba. Cylindrickú projekciu je teda možné zobraziť aj ako obyčajnú fotografiu, ktorá však bude veľmi široká z dôvodu zachytania scény v celom horizontálnom pásme. Na rozdiel od cylindrického pohľadu, sférické projekcie majú vertikálny uhol pozorovania 180° a horizontálny uhol pohľadu 360° . Obsahujú svetelné údaje pochádzajúce zo všetkých smerov a preto je možné ich vizualizovať tak, že premietajú jednotlivé body do gule. Medzi najznámejšie a súčasne používané formáty patria zobrazenie pomocou kubickej mapy (angl. cubemap) a ekvidistantné zobrazenie (angl. equirectangular). Tieto projekcie sa líšia v určitých vlastnostiach:

- **Kubické zobrazenie** - pozostáva zo 6 samostatných plôch kocky na vyplnenie celej scény. Tieto mapy sa často vytvárajú zobrazením scény pomocou skupiny šiestich 90-stupňových kamier, ktoré poskytujú textúru vľavo, spredu, vpravo, zozadu, zhora a dole. Šesť obrázkov je zvyčajne usporiadaných ako rozložená kocka. Každá plocha má zodpovedajúcu textúrnu mapu. Po zložení je pohľad premapovaný na plochy kocky, a pripomínajú kríž položený na bok. Jednotlivé plochy sa dokážu spojiť veľmi jednoducho. Na jednej strane je kubická projekcia akousi povrchovou textúrou ako obyčajná 2D textúra. Na druhej strane je to určitý formát zobrazenia sférickej fotografie, pretože každý bod v 3D súradnicovom priestore textúry zodpovedá ploche kocky, ku ktorej je najbližšie.
- **Ekvidistantné zobrazenie** - tento formát sa stal populárnym v sociálnych sieťach a našiel uplatnenie napríklad v 3D grafických programoch, simuláciách interiérov, panoramatických filmoch či počítačových videohráč. Táto projekcia pozostáva z jedného obrázka v tvare obdĺžnika, ktorého šírka a výška sú v pomere 2:1. Formát sférickej fotografie je podstatne odlišný od obyčajnej fotografie. Aby bolo možné zachytiť celú priestorovú scénu, je potrebné reprezentovať priestorový záber.



Obr. 14: Znázornenie sférických projekcií - kubické zobrazenie (a), ekvidistantné zobrazenie (b)

Konverzia rovnostranného obrázka na kubickú mapu sa najčastejšie používa pre niektoré riešenia virtuálneho prostredia alebo pri úprave severných a južných pólov sférických panorám. Rovnostranné obrázky sú roztiahnuté v horizontálnom smere. Je to dôvod na značné množstvo redundancie údajov v blízkosti pólov. Pri zmenšovaní obrázka v editore sa efektívne rozlíšenie textúry zníži podľa očakávania - s výnimkou blízkych pólov. To môže spôsobiť rôzne artefakty v obrazu, ktoré sa zobrazia pri zobrazovaní 3D fotografie. Rovnostranná projekcia je preto vhodná na simuláciu iba tých prostredí, v ktorých sú deformácie textúry v hornej a dolnej časti gule zanedbatelné. Riešením je prepnnúť na menej zdeformovanú projekciu pred zmenšením mierky, rozmazaním alebo zaostrením panorámy a v prípade potreby prepnnúť do sférického režimu ne-skôr. Kubická projekcia našla využitie v počítačovej grafike v reálnom čase. Mapovanie prostredia alebo skôr guľové mapovanie sa používa na vytváranie lesklých alebo reflexných objektov. V takom prípade by textúra mapy mala byť pohľadom na scénu, ktorá sa odráža v lesklej gule. V tejto práci však nie je úplný sférický záber potrebný. Hlavným snímaným objektom je vodič, ktorý dokáže byť zachytený aj kamerou s dostatočne širokým záberom napríklad športovou akčou kamerou. Z tohto dôvodu sa v programe využíva iba časť výstupu kamery, v ktorej sa nachádza celá postava vodiča.



Obr. 15: Rovnaká sférická fotografia v rôznych zobrazeniach - kubické zobrazenie (a), ekvidistantné zobrazenie (b)

5 Vlastná implementácia

Táto kapitola je zameraná na popis implementácie vlastného programu. Hlavnou podstatou výsledného programu je detekcia vodiča z videa alebo z jednotlivých obrázkov nahratých kamerou z interiéru vozidla a analýza jeho správania. V kapitole sú tiež detailne popísané možnosti, problémy a riešenia nahrávania samotných videí z interiéru vozidla pri rôznych pracovných podmienkach. Z dôvodu technickej náročnosti sú tieto videá spracované až dodatočne po ich stiahnutí do počítača, kde sú následne spracované programom. Riešenie teda neumožňuje spracovanie videa v reálnom čase počas jazdy. To však neznamená, že taká možnosť neexistuje. Tento program môže poslúžiť ako inšpirácia pre budúce riešenia, ktoré by sa zameriavali na spracovanie podobného druhu aj v reálnom čase.

Program by mal byť schopný detektovať vodičove správanie a znázorňovať stav daného vodiča s vyhodnotením situácie. Medzi takúto detekciu patrí napríklad analýza posedu v aute, umiestnenie rúk alebo natočenie hlavy. Program musí podporovať použitie viacerých frameworkov na detekciu ľudských postáv a umožniť vzájomné porovnanie medzi sebou formou vhodného výstupu. Základné metódy, ktoré musí program podporovať sú Tensorflow Pose Estimation a OpenPose. Tieto frameworky boli zvolené najmä z dôvodu, že sa jedná o najviac používané metódy na detekciu postáv v súčasnosti. Okrem toho majú výborne zdokumentované zdrojové kódy a dostatočné množstvo informácií pre vlastnú implementáciu. Okrem základnej detekcie postáv program umožňuje aj rozšírený mód. Tento rozšírený mód sa zameriava na detekciu ďalších vlastností šoféra vozidla ako napríklad natočenie hlavy. Z týchto informácií je potom možná ďalšia analýza vodičovho správania a vyhodnotenie rizík pri vedení vozidla. Program je rozdeľený do jednotlivých samostatných celkov, ktoré sú na sebe nezávislé. Každý z týchto celkov je zameraný na konkrétnu časť:

- **Detekcia vodiča** (Kapitola 5.4) - základným prvkom programu je analýza vodičovho posedu. V kapitole je popísané využitie jednotlivých frameworkov na detekciu a analýza ľudského tela zo zozbieraných údajov pomocou jednotlivých frameworkov a ich vzájomné porovnanie medzi sebou.
- **Analýza pozície vodiča** (Kapitola 5.5) - v tejto časti je popísaná implementácia neurónovej siete, ktorá vyhodnocuje výstup z detekcie postavy. Neurónová sieť je zameraná hlavne na kontrolu správnej pozíciu vodiča za volantom
- **Kontrola natočenia hlavy** (Kapitola 5.6) - je zameraná na rotáciu a natočenie hlavy vodiča pomocou detekcie tvárových landmarkov a knižnice dlib [27]. Z tejto detekcie je možné určiť, či vodič správne sleduje situáciu v premávke.

5.1 Požiadavky a návrh programu

Každý vyvíjaný program, by mal spĺňať určité očakávania. Tieto očakávania a požiadavky by mali byť správne zadefinované ešte samotným vývojom. Tieto požiadavky okrem toho, že určia samotnú funkciu programu, zároveň umožňujú analyzovať splnenie jednotlivých bodov. Požiadavky by mali byť priebežne spracované. Pri správnej analýze ešte pred samotným vývojom programu je potom jednoduché kontrolovať splnenie jednotlivých bodov. Program, ktorý je navrhnutý v tejto diplomovej práci obsahuje niekoľko dôležitých podmienok, ktoré zaručia dostatočnú kvalitu výstupu programu a jeho použitie. Jednotlivé požiadavky sú zhrnuté v nasledujúcich bodoch:

- **Jednoduché používanie** - program musí byť jednoduchý na používanie. To znamená, že by mal obsahovať iba určitú funkciu, ktorej výsledok je požadovaný a zrejmý. K programu by mala byť zároveň vytvorená jednoduchá dokumentácia, ktorá by obsahovala jednotlivé príkazy a popis parametrov.
- **CLI podpora** - pre univerzálnosť použitia by mal program byť spustiteľný z príkazového riadku. Jednotlivé časti programu by mali byť jednoducho používané pomocou voliteľných parametrov.
- **Nezávislosť na použitej kamere** - program by mal byť schopný spracovať akékolvek video, ktoré bude nahraté z interiéru vozidla. Nemal by byť závislý na formáte, ani type samotných videí.
- **Podpora viacerých frameworkov na detekciu postáv** - program musí podporovať aspoň 2 rôzne frameworky, ktoré slúžia na detekciu postáv v obrazu. Ich použitie by nemalo byť závislé na type videa alebo použití inej funkcionality. Táto podmienka vyžaduje zjednotenie vstupov a výstupov pre jednotlivé frameworky na detekciu postáv v obrazu.
- **Znázornenie výsledku detektie** - pre program by mal umožňovať výpis jednotlivých časti programov priamo do obrazu.
- **Štatistické údaje** - program by po svojom úspešnom skončení mal byť schopný zozbierať a vyhodnotiť údaje z celého behu programu. Dôležité údaje sú napríklad čas spracovania jednej snímky, celková úspešnosť a celková doba programu.
- **Univerzálnosť** - aby bolo jednoduché použiť program na viacerých operačných systémov, je potrebné ho napísat tak, aby nevyužíval žiadnu funkciu, ktorá je závislá na konkrétnom operačnom systéme.

5.2 Použité technológie

Programovací jazyk

Hlavný program je napísaný v jazyku Python. Tento jazyk bol použitý z dôvodu vynikajúcej dostupnosti knižníc na spracovanie obrazu, ale aj knižníc na prácu s neurónovými sietami. Okrem toho je jazyk Python veľmi dobre zdokumentovaný a má vybudovanú širokú komunitu medzi programátormi. Okrem jazyka Python taktiež bolo zvažované použitie jazyka C++. Napriek mnohým nesporným výhodám jazyka C++ padlo rozhodnutie pre jazyk Python najmä po dôkladnom zvážení nasledujúcich možností:

- **Automatická správa pamäte** - Python má narozenie od C++ implicitne vyriešenú automatickú alokáciu a dealokáciu pamäte. To dáva programátorovi výhodu v ušetrení času za cenu minimálneho zníženia výkonu aplikácie.
- **Široká podpora knižníc** - v súčasnosti pre jazyk Python existuje obrovské množstvo volne dostupných knižníc, ktoré ulahčujú prácu pri vývoji softvéru. Jedná sa najmä o knižnice zamerané na prácu s neurónovými sietami a spracovanie obrazu.
- **Jednoduchosť jazyka** - Python je narozenie od C++ jednoduchší a pohodlnnejší na používanie. Poskytuje mnohé výhody, ktoré jazyk C++ nepodporuje ako napríklad dynamické dátové typy, jednoduchá inštalácia a použitie potrebných balíčkov v programe.

OpenCV

OpenCV je multiplatformová knižnica, ktorá obsahuje funkcie pre spracovanie a manipuláciu s obrazom v reálnom čase. Vďaka licencii BSD je možné používať OpenCV zadarmo nielen pre akademické, ale aj pre komerčné účely. S jej využitím je možné sa stretnúť pri analýze a spracovanie snímok z rôznych oblastí. Aplikácie, ktoré sú napísané a optimalizované v C/C++, potom môžu pre svoj beh využívať viacjadrové procesory. V súčasnosti sa výrazne rozšírila podpora aj pre jazyk Python a CUDA pre spracovanej na grafickom procesore. V programe je OpenCV využité na základnú prácu s obrazmi ako načítanie snímky z videa, zmena veľkosti, detekcia hrán a podobne.

TensorFlow

TensorFlow [28] bol pôvodne vyvinutý výskumníkmi a inžiniermi pracujúcimi v tíme Google Brain v rámci organizácie Google Research Intelligence Research na vykonávanie strojového učenia a výskumu neurónových sietí. Framework je dostatočne všeobecný na to, aby sa dal použiť aj v mnohých ďalších doménach. Tensorflow poskytuje stabilné API pre jazyky Python a C++, ako aj nezaradené kompatibilné API pre iné jazyky. V programe je Tensorflow využitý najmä na detekciu postáv a prácu s neurónovými sietami.

Keras

Keras [29] je framework pre hĺbkové učenie pôvodne vyvinutý pre jazyk Python. Poskytuje pohodlný spôsob, ako definovať a trénovať takmer akýkoľvek druh hlbokého učenia. Keras je vysokoúrovňové API pre neurónové siete, ktoré je schopné pracovať s inými knižnicami ako napríklad nad Tensorflow, Theano [30] a CNTK. V aplikácii je Keras využitý na vytvorenie modelu a jednotlivých vrstiev neurónovej siete.

5.3 Vytvorenie a spracovanie videa

V tejto sekcií je postupne spísaný postup a problematika, ktorá sa týka nahrávania videí potrebných pre analýzu vodiča v tejto diplomovej práci. Samotný program je pôvodne navrhnutý na spracovanie videa v 360 stupňovom formáte. Implementácia však nie je žiadnym spôsobom limitovaná a program je schopný spracovať aj obyčajné video nahraté bežne dostupnou kamerou. Hlavným dôvodom použitia sférickej kamery je jej uhol záberu. Vďaka širokouhlému obrazu je možné získať obraz celého vodiča sediaceho za volantom. Získanie takéhoto obrazu umožňuje analyzovať jednotlivé časti tela, tvár, ale aj vodičovo správanie.

Počas vypracovania práce boli k dispozícii dve širokouhlé kamery od rôznych výrobcov. Na kamerách je umiestnené minimálne množstvo tlačidiel, ktoré slúžia primárne iba na zapnutie a vypnutie kamery. Celé ovládanie je cez intuitívne mobilné aplikácie, ktoré sú dostupné na stiahnutie na stránkach výrobcov pre všetky mobilné platformy používané v súčasnosti. Každá kamera obsahuje dva širokouhlé objektívy, s ktorými sa vytvára sférický záber okolitej scény. Kamery majú odlišné parametre a vlastnosti, ktoré sú zhrnuté v tabuľke 1. Hlavný rozdiel je možné vidieť najmä v maximálnom rozlíšení, ktoré je pri videu pre kameru GoPro Fusion takmer dvojnásobné oproti rozlíšeniu kamery Ricoh Theta V. Pri spracovaní videa a detekcii vodiča však nemusí rozlíšenie hrať významnú úlohu. Pri detekcii objektov sférickou kamerou zohrávajú úlohu aj ostatné parametre ako napríklad svetelnosť použitých senzorov.

	GoPro Fusion	Ricoh Theta V
Senzor	FishEye CMOS $2 \times 18\text{MP}$	FishEye CMOS $2 \times 12\text{MP}$
Max. rozlíšenie (foto)	18MP	14.4MP
Max. rozlíšenie (video)	13.7MP	7.3MP
Svetelnosť senzora	f/2.0	f/2.0
Pamäť	SD karta (256GB)	19GB

Tabuľka 1: Technické parametre sférických kamier

Aj napriek tomu, že svetelnosť senzora je totožná pri obidvoch kamerách, rozdiel v kvalite výstupného videa je ľahko rozpoznateľný najmä pri znížených svetelných podmienkach. Kým za denného svetla sú kvalita aj kontrast obrazu skoro totožné pre obidve kamery, pri tmavých scénach je obraz z kamery Ricoh Theta V nekvalitný a obsahuje vysokú úroveň šumu za cenu vyšej priepustnosti svetelnosti (obr. 16). Pri natáčaní tmavých scén bolo použité na obidvoch kamerách ISO 1600. Napriek rovnakým nastaveniam je z obrázku vidieť, že výstup z kamery GoPro Fusion je značne tmavší. Na tomto obrázku zároveň vidieť problém, ktorý nastáva pri znížených svetelných podmienkach. Kedže ani jedna kamera nie je vybavená infračerveným svetlom, ktoré by pomáhalo v noci, detekcia kamerou nie je veľmi účinná. Svetelnosť scény je možné upraviť zvýšením ISO ale za cenu vyššieho šumu v obrazze. Z tohto dôvodu bolo na kamerách nastavené vždy automatické ISO, ktoré sa prispôsobovalo okolitým svetelným podmienkam. Hlavne z tohto dôvodu väčšinou prebiehal zber záberov počas denného svetla. Pri použití kamery v noci by bolo potrebné nájsť rozumný spôsob, ako scénu nasvetliť aby postava vodiča bola viditeľná. Pri spracovaní videí z obidvoch kamier, bolo používané iba video z prednej kamery. Pre potreby tejto práce bol potrebný záber celého tela vodiča za volantom. Z dôvodu, že táto práca je zameraná iba na detekciu vodiča je záber jedným objektívom postačujúci a nemá zmysel spracovať výstup z druhého objektívu. Takýmto spôsobom bolo vytvorených približne 15 minút záznamov z každej kamery v rôznych podmienkach.



(a)



(b)

Obr. 16: Porovnanie výstupu z kamier (ISO 1600) - GoPro Fusion (a), Ricoh Theta V (b)

5.4 Detekcia vodiča

Po úspešnom vytvorení testovacích videí prichádza na rad detekcia vodiča. Pred samotnou detekciou je nutné video najsť upraviť, aby bolo podľa možnosti v čo najvhodnejšej forme pre jednotlivé frameworky. Dôležitá úprava, ktorú je potrebné urobiť je zmena rozlíšenia. Všetky frameworky na detekciu postáv v obrazoch pracujú odlišne vzhladom na zvolené rozlíšenie. Úpravou rozlíšenia je tiež možné možné ľahšie testovať a hľadať najvhodnejšie parametre pre dosiahnutie najvyššej úspešnosti. Na úpravu rozlíšenia je vytvorená funkcia, ktorá na základe požadovanej výšky snímky zmení veľkosť snímky tak, aby bol zachovaný správny pomer strán. Tým je zabezpečené, že obraz nebude žiadnym spôsobom deformovaný a výsledné rozlíšenie je ľahko nastaviteľné. Táto funkcia bude využívaná aj v ďalších častiach práce.

Obidva použité frameworky pracujú na odlišných princípoch. Každý framework má dostačne spracovanú dokumentáciu, z ktorej sú odvodené metódy na samotnú implementáciu. Okrem toho používajú odlišné datasety, ktoré sa dajú parametrizovať a vybrať ten najvhodnejší. Každý framework vyžaduje správny postup krokov pre použitie vo vývojárskom prostredí. Po správnej inštalácii všetkých potrebných knižníc a balíčkov je potrebné ich správne nakonfigurovať pomocou parametrov.

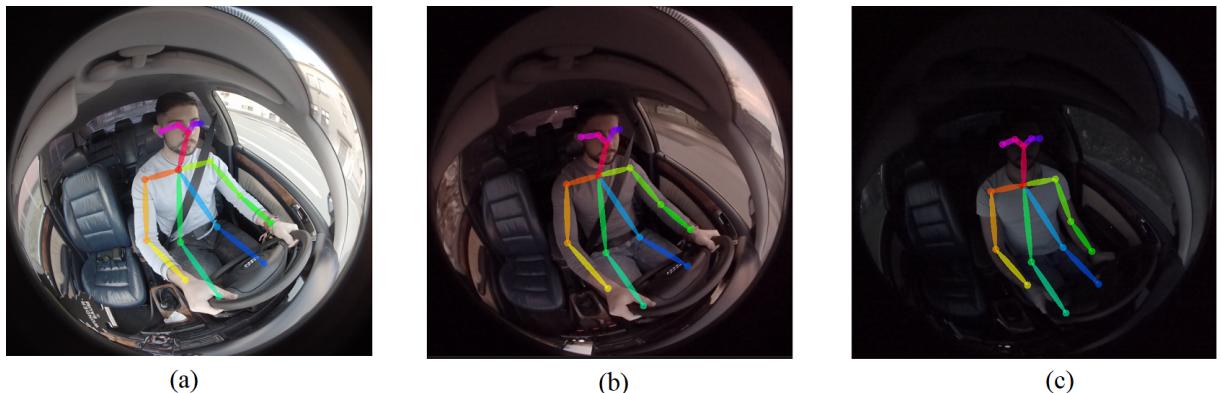
OpenPose

Pri implementácii OpenPose bol použitý dataset MPI a dataset COCO. Výber datasetu je možný cez parameter. OpenPose podporuje aj použitie moderného a presnejšieho datasetu BODY_25, avšak z dôvodu nedostatočného výpočtového výkonu nebol v tejto práci používaný. Ako prvé je potrebné nainštalovať samotnú knižnicu OpenPose, ktorá sa inštaluje cez nástroj CMake. Po úspešnej inštalácii je možné knižnicu používať v ľubovoľnej implementácii. Aby OpenPose správne fungoval, je potrebné ho taktiež nakonfigurovať pomocou nasledujúcich parametrov:

- **model_folder** - parameter, ktorý slúži na nastavenie cesty k použitému datasetu. Program je navrhnutý tak, aby tieto modely boli umiestené v rovnakej zložke ako sa nachádza program.
- **model_pose** - názov používaneho datasetu. Hodnota môže byť MPI, COCO, alebo BODY_25. Dataset BODY_25 však v tejto práci nie je použitý. Datasety musia byť pred použitím najsť stiahnuté.
- **number_people_max** - parameter určuje, kolko ľudských postáv má byť v obraze detekovaných. Každá zdetekovaná postava v obraze je číselne ohodnotená. Pri parametri nastavenom na hodnotu 1 sa vyberie iba jedna postava, ktorá ma toto číselné skóre najvyššie. Tento parameter je vhodné používať napríklad pri viacerých postavách v obraze, ktoré nemusia byť primárnym cieľom detekcie. Pri implementácii je tento parameter nastavený na 1, pretože na detekciu je dôležitá iba najvýraznejšia postava v obraze, ktorou je vodič.

- **net_resolution** - parameter, ktorý určuje rozlíšenie obrazu, v ktorom sa má spracovať. Toto rozlíšenie musí obsahovať násobky čísla 16. Ak sa zvýši, presnosť sa potenciálne zvýší. Ak sa rozlíšenie zníži, rýchlosť sa zvýši, ale za cenu nižšej presnosti detekcie. Na dosiahnutie maximálneho vyváženia rýchlosť a presnosti by mal byť zachovaný čo najbližší pomer strán k obrázkom alebo videám, ktoré sa majú spracovať.

OpenPose obsahuje mnoho ďalších nastaviteľných parametrov, ktoré sú zvyčajne potrebné v špecifických alebo individuálnych prípadoch detekcie. Pri detekcii postavy v interiéri vozidla nastavenie uvedených parametrov postačuje. Po úspešnej inicializácii knižnice prichádza na rad samotná detekcia. Hlavná metóda na detekciu berie ako parameter obrázok v ľubovoľnom formáte. Pred použitím je vhodné na obrázku znížiť rozlíšenie prípadne nastaviť oblasť záujmu. Vďaka týmto vylepšeniam je možné dosiahnuť lepšie výsledky a priaznivý čas detekcie. Aby sa výsledky medzi metódami dali porovnávať, na meranie času je použitá knižnica *time*. Výstup metódy je realizovaný formou výstupného objektu. Tento objekt obsahuje výsledný obraz, kde sú ilustrované detekované časti tela, ale aj zoznam jednotlivých častí s pozíciou podľa použitého datasetu. Tento zoznam je užitočný najmä pri ďalšom spracovaní časti ľudského tela. Testovacie videá boli natočené kamerou GoPro Fusion pri rôznych svetelných podmienkach. Ukážka výsledku detekcie pri rozličných svetelných podmienkach je znázornená na obrázku 17.



Obr. 17: Natívny výstup metódy OpenPose - normálne svetelné podmienky (a), zhoršené svetelné podmienky (b), nevyhovujúce svetelné podmienky (c).

Na obrázkoch je znázornený výstup z metódy OpenPose testovaný na bežnej polohe vodiča pri rôznych svetelných podmienkach. Metóda si úspešne dokázala poradiť aj so zníženými či dokonca nevyhovujúcimi svetelnými podmienkami. Detekcia vodiča z ukážky bola testovaná na rozlíšení 729×700 . Z 3392 snímok z testovacieho videa bola postava vodiča zdetektovaná $3392 \times$. To znamená, že úspešnosť detekcie sa pohybovala na úrovni 100% aj pri nevyhovujúcich svetelných podmienkach. Metóda pracovala spoločne ale nedostatočne rýchlo. Pri zvolenom rozlíšení sa spracovanie jednej snímky pohybovalo v rozmedzí 780ms až 950ms. Tieto údaje sú podrobne zhrnuté v tabuľkách 2 a 3.

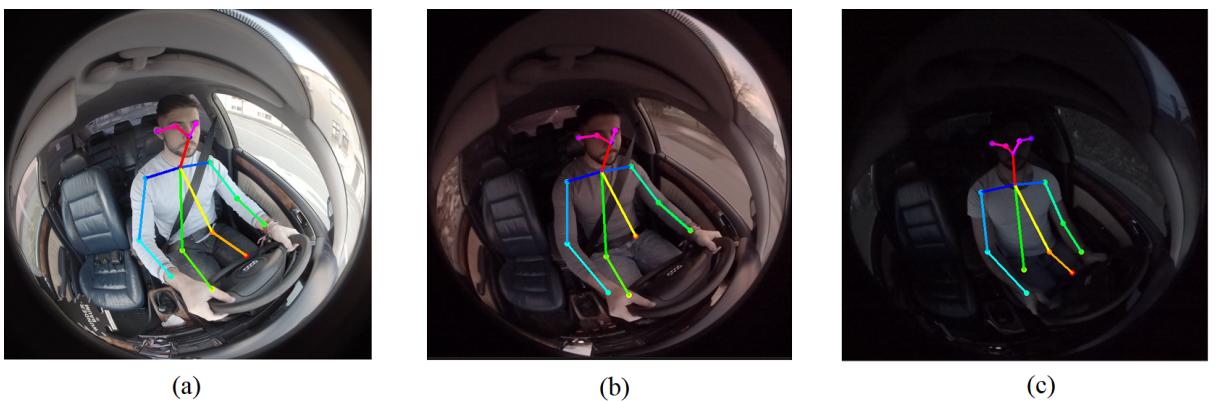
TF Pose estimation

Framework TF Pose estimation je založený na knižnici Tensorflow, z ktorej využíva primárne prácu s neurónovými sieťami, ale s mierne upravenou architektúrou. Framework poskytuje po-kročilú detekciu postáv v obrazoch. Základné použitie frameworku TF Pose Estimation je veľmi podobné použitiu OpenPose. Základný krok je nainštalovať požadované balíčky a knižnice, aby framework mohol správne fungovať. Narozenie od OpenPose však tento framework používa rozdielne datasety. Tieto datasety sú optimalizované primárne pre zariadenia s nižším výpočtovým výkonom. Vďaka tomu je výhodné túto knižnicu použiť napríklad na detektovanie ľudských postáv priamo na mobilných zariadeniach pri optimálnej časovej náročnosti. To dáva výhodu oproti použitiu iných frameworkov. Program je však navrhnutý pre spracovanie na počítačoch, takže túto výhodu nie je možné využiť. Ak by však detekcia mala prebiehať priamo na mobilnom zariadení, TF Pose Estimation by bol jedným z vhodných riešení. K frameworku je vytvorených viacerých datasetov, ktoré sa v určitých vlastnostiach líšia:

- **cmu** - dataset, ktorý je založený na jednoduchom modeli postavy a bol pôvodne využitý pri vytváraní frameworku. Aj napriek tomu, že existujú vhodnejšie a modernejšie datasety, ostáva stále podporovaný kvôli spätej kompatibilite. Dataset *cmu* je založený na knižnici Caffe. Dataset obsahuje upravené váhy siete na efektívne použitie s knižnicou Tensorflow.
- **dsconv** - používa rovnakú architektúru ako model *cmu*, avšak bez hĺbkovo oddeliteľnej konvolúcii. Model však nie je dostatočne rýchly ani presný pre bežné použitie. Z tohto dôvodu nie je odporučený na používanie ani samotným tvorcom frameworku.
- **mobilenet** - je založený na modeli mobilenet, určený pre mobilné platformy. Na extrakciu vlastností je použitých 12 konvolučných vrstiev. Jedná sa o model, ktorý je upravený aj pre možnosť detekcie menej výrazných postav v obraze. Dataset sa člení na 3 typy podľa použitia: *mobilenet*, *mobilenet_fast*, *mobilenet_accurate*
- **mobilenet_v2** - jedná sa o najnovší dataset, ktorý je založený na vylepšených vlastnostiach datasetu *mobilenet*. Tento dataset obsahuje viaceré mutácie, ktoré vznikli hlavne kvôli veľkosti samotného datasetu a optimalizovaní pre použitie na mobilných zariadeniach s obmedzenou veľkosťou úložiska.

Použitie frameworku spočíva v jeho inicializácii na začiatku programu. Pri inicializácii je potrebné určiť, ktorý dataset sa použije a zároveň rozlíšenie, v akom sa obrázok bude spracovať. Od použitého rozlíšenia závisí viaceré faktorov. Ak je rozlíšenie príliš malé, detekcia je rýchla ale nepresná a mnohokrát sa postava v obraze ani nenájde. Ak je rozlíšenie príliš vysoke, detekcia má súčasne vysokú úspešnosť, ale doba spracovania jedného obrázku sa pohybuje v jednotkách sekúnd, čo pre detekciu vodičovho správania nie je vhodné. Obidve pramiatere je možné cez argument upraviť a následne sledovať úspešnosť a čas detekcie. Výstup metódy detekcie poskytuje súradnice jednotlivých častí tela, ktoré sú v percentuálnom pomere vzhladom

k výške a šírke vstupného obrázku. Aby bolo možné s výsledkami viacerých metód pracovať je potrebné výstupy metód najskôr zjednotiť do rovnakého formátu. Okrem týchto súradníc metóda poskytuje aj vykreslenie ľudského tela do pôvodného vstupného obrazu. Túto možnosť je vhodné použiť pre základné overenie funkčnosti detekcie, čo je znázornené na obrázku 18. Ako je vidieť, metóda dokáže pracovať aj pri zhoršených, či dokonca nevyhovujúcich podmienkach. Z testovacieho videa natočeného po západe slnka, kedy do interiéru už nepreniká dostatočné množstvo svetla, sa z 3392 snímok pozícia vodiča zdetekovala 3390×. Pri horších svetelných podmienkach sa už pomerne často začal opakovat problém toho, že nohy vodiča už neboli v tme zdetekované. Pozícia nôh však pre analýzu vodičovho správania nie je až taká kľúčová, preto je možné metódu používať aj pri zhoršených svetelných podmienkach. Analyzovanie pozície vodiča je zamerané hlavne na vrchnú časť tela.



Obr. 18: Natívny výstup metódy TF pose - normálne svetelné podmienky (a), zhoršené svetelné podmienky (b), nevyhovujúce svetelné podmienky (c).

Zjednotenie výstupov

Hlavným problémom pri použití viacerých knižníc na detekciu v jednom programe je zjednotenie ich vstupov a výstupov. Aj napriek tomu, že pracujú na rovnakých alebo podobných princípoch, každá knižnica potrebuje iné vstupné parametre. Za rozdiel na výstupe metód je zodpovedný aj samotný dataset. Jednotlivé datasety obsahujú rôzny počet častí tela, ktoré nemusia byť medzi sebou vzájomne kompatibilné. Pre používanie viacerých frameworkov v programe je vytvorená metóda, ktorá zabezpečuje zjednotený vstup aj výstup metód na detekciu. Pri vstupe je dôležité obrázok najmä upraviť a podľa parametru zavolať výslednú metódu konkrétnej knižnice na detekciu. Pre spracovanie výstupov je vytvorený konvertor, ktorý zabezpečuje rovnaký výstup pre každú metódu. Výstup je realizovaný formou zoznamu, ktorý obsahuje súradnice jednotlivých častí tela. Do úvahy však treba počítať aj prípad, že nemusia byť všetky časti tela zdetekované alebo môžu v konkrétnom frameworku chýbať. Pre takýto prípad je vhodné ošetriť každú časť tela osobitne. Zjednotenie výstupu frameworkov TF Pose Estimation v jazyku Python je zobrazené vo výpise 2.

```

if(pose_type == "OP_POSE"):
    head = (int(human[0][0]), int((human[0][1] + human[1][1])/2))
    neck = (int(human[1][0]), int(human[1][1]))
    shoulder_left = (int(human[5][0]), int(human[5][1]))
    shoulder_right = (int(human[2][0]), int(human[2][1]))
    hip_left = (int(human[11][0]), int(human[11][1]))
    hip_right = (int(human[8][0]), int(human[8][1]))
    elbow_left = (int(human[6][0]), int(human[6][1]))
    elbow_right = (int(human[3][0]), int(human[3][1]))
    wrist_left = (int(human[7][0]), int(human[7][1]))
    wrist_right = (int(human[4][0]), int(human[4][1]))

elif (pose_type == "TF_POSE"):

    head = (int(human.body_parts[0].x * image_w + 0.5), int(human.body_parts[0].y * image_h + 0.5))
    neck = (int(human.body_parts[1].x * image_w + 0.5), int(human.body_parts[1].y * image_h + 0.5))
    shoulder_left = (int(human.body_parts[5].x * image_w + 0.5), int(human.body_parts[5].y * image_h + 0.5))
    shoulder_right = (int(human.body_parts[2].x * image_w + 0.5), int(human.body_parts[2].y * image_h + 0.5))
    hip_left = (int(human.body_parts[11].x * image_w + 0.5), int(human.body_parts[11].y * image_h + 0.5))
    hip_right = (int(human.body_parts[8].x * image_w + 0.5), int(human.body_parts[8].y * image_h + 0.5))
    elbow_left = (int(human.body_parts[6].x * image_w + 0.5), int(human.body_parts[6].y * image_h + 0.5))
    elbow_right = (int(human.body_parts[3].x * image_w + 0.5), int(human.body_parts[3].y * image_h + 0.5))
    wrist_left = (int(human.body_parts[7].x * image_w + 0.5), int(human.body_parts[7].y * image_h + 0.5))
    wrist_right = (int(human.body_parts[4].x * image_w + 0.5), int(human.body_parts[4].y * image_h + 0.5))

return [head, neck, shoulder_left, shoulder_right, hip_left, hip_right,
        elbow_left, elbow_right, wrist_left, wrist_right, knee_left, knee_right]

```

Výpis 2: Mapovanie častí ľudského tela OpenPose a TF Pose Estimation

Porovnanie výsledkov metód na detekciu postáv

Testovanie úspešnosti prebehlo na viacerých videách pri rôznych pracovných a svetelných podmienkach. Aby bola simulovaná bežná jazda v aute, na simuláciu bolo primárne použité video nahraté v slabších svetelných podmienkach ako je v interiéri vozidla obvyklé. Počas testovacieho videa vodič vykonáva bežnú činnosť ako držanie a točenie volantu, pohyb riadiacou pákou ale aj zapínanie bezpečnostného pásu. Pri zapínaní bezpečnostného pásu dochádzalo k prudšiemu pohybu a kamera pri slabom svetle tento pohyb nedokázala zachytiť bez značného rozmazania videa. Práve pri tomto kroku vznikalo najviac prípadov chýbajúcej postavy vodiča. Úspešnosť detekcie bola meraná na základe zdetektovanej postavy v obraze a dostatočným počtom detektovávajúcich častí tela. Aby sa výsledok detekcie označil ako úspešný, museli byť zdetektované nasledujúce časti: hlava, krk, ramená a lakte. V prípade, že tieto časti neboli v obraze nájdené, detekcia bola označená ako neúspešná. Samotné označenie častí tela prebiehalo spoľahlivo. To znamená, že napríklad ramená sa nachádzali na mieste, kde ramená naozaj boli. Pri testovaní nedochádzalo k chybnej detekcii a nesprávnemu označeniu častí tela. V tabuľkách 2, 3 je znázornená úspešnosť a časová náročnosť pre framework OpenPose pri rôznych rozlíšeniach vstupného obrázku. Ako je z tabuľiek vidieť, miera úspešnosti je pomerne vysoká, ale pri nižšom rozlíšení vstupného obrazu je detekcia v priemere o niekoľko desiatok milisekúnd rýchlejšia so stále dostačujúcou úspešnosťou okolo 99%. Veľký problém tejto detekcie tvorí práve vysoká časová náročnosť, ktorá neumožňuje spracovávať video v reálnom čase.

Rozlíšenie vstupného obrazu [px]	Úspešnosť [%]	Čas spracovania snímky [ms]
938×900	100	950
521×500	100	820
417×400	100	780
208×200	100	775
104×100	99.67	776
52×50	99.43	776

Tabuľka 2: Úspešnosť detekcie OpenPose - slabé svetelné podmienky, model COCO

Rozlíšenie vstupného obrazu [px]	Úspešnosť [%]	Čas spracovania snímky [ms]
938×900	91.35	850
521×500	89.43	770
417×400	88.64	760
208×200	86.87	730
104×100	65.98	724
52×50	5	728

Tabuľka 3: Úspešnosť detekcie OpenPose, slabé svetelné podmienky, model MPI

Pri testovaní TF Pose Estimation sa postupovalo podobne ako pri predchádzajúcim framework OpenPose. Na testovanie bola použitá rovnaká ukážka videa, ktoré bolo nahraté v znížených svetelných podmienkach. Test prebehol pri použití datasetu *mobilenet_thin* ktorý je odporúčaný autormi TF Pose Estimation. Pri použití iných datasetov často dochádzalo k fantómovým detekciám postavy, čo by mohlo negatívne ovplyvniť funkčnosť celého programu. Okrem zmeny datasetu TF Pose Estimation poskytuje aj možnosť volby rozlíšenia trénovacieho modelu. Zmenou tohto rozlíšenia je možné ovplyvňovať hlavne rýchlosť a presnosť detekcie. Zmena tohto parametru je významná najmä na zariadeniach s nižším výpočtovým výkonom. Test prebehol pri 3 rôznych rozlíšeniach trénovacích modelov. Výsledky sú zhrnuté v tabuľkách 4, 5, 6. Ako je z výsledkov vidieť, detekcia sa vo väčšine testov pohybovala na vysokej úrovni ($> 90\%$) pri použití rozlíšenia trénovacieho vstupu 216×184 pixelov. Naroďod od frameworku OpenPose však detekcia bola niekolkonásobne rýchlejšia. Práve rýchlosť je kľúčový parameter, od ktorého môže výsledný program závisieť. Pri výbere parametrov je potrebné zamyslieť sa nad tým, či vo výsledku je dôležitejšia vyššia presnosť alebo rýchlosť detekcie.

Rozlíšenie vstupného obrazu [px]	Úspešnosť [%]	Čas spracovania snímky [ms]
938×900	99.89	690
521×500	99.89	600
417×400	99.86	590
208×200	99.62	578
104×100	92.01	558
52×50	56.70	550

Tabuľka 4: Úspešnosť detekcie TF Pose Estimation, dataset *mobilenet_thin*, rozlíšenie trénovacieho vstupu 432×368 pixelov

Rozlíšenie vstupného obrazu [px]	Úspešnosť [%]	Čas spracovania snímky [ms]
938×900	97.94	344
521×500	97.64	257
417×400	97.82	245
208×200	96.08	220
104×100	91.10	213
52×50	85.18	206

Tabuľka 5: Úspešnosť detekcie TF Pose Estimation, dataset *mobilenet_thin*, rozlíšenie trénovacieho vstupu 216×184 pixelov

Rozlíšenie vstupného obrazu [px]	Úspešnosť [%]	Čas spracovania snímky [ms]
938×900	90.89	320
521×500	87.83	242
417×400	88.20	228
208×200	87.00	211
104×100	80.15	199
52×50	61.54	192

Tabuľka 6: Úspešnosť detekcie TF Pose Estimation, dataset *mobilenet_thin*, rozlíšenie trénova- cieho vstupu 192×144 pixelov

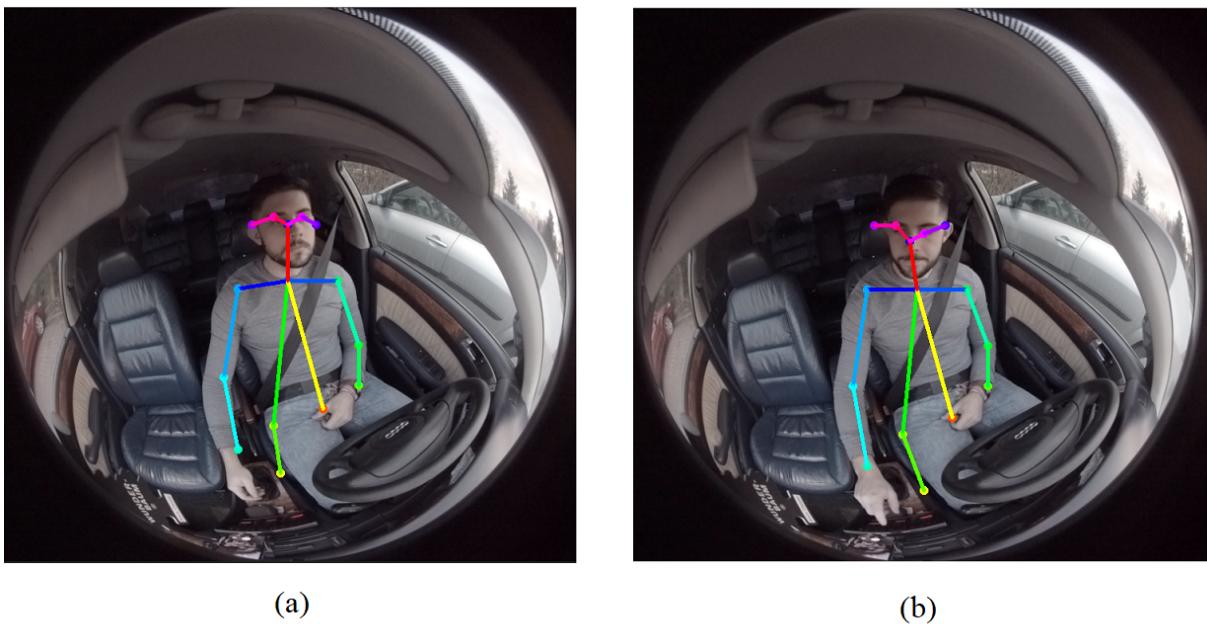
Po otestovaní a zhodnotení výsledkov metód je dôležité vybrať tú správnu metódu na použitie ďalšej analýzy vodiča. Do úvahy treba brať viaceré faktory, ktoré sú klúčové pre výsledný program. Vzhľadom na diametrálne odlišné časové náročnosti jednotlivých metód a pomerne vysokú úspešnosť obidvoch metód, bude lepšie využiť metódu s rýchlejšou detekciou. Pri použití tejto detekcie treba počítať s možnými chybami, ktoré je možné odstrániť jednoduchým filtrom, alebo použitím prahovej hodnoty na počet snímkov s chybovou detekciou. Pri bežnom videu sa pohybuje rýchlosť snímkovania okolo 30FPS. Aby detekcia teda bola schopná rozoznať chybu do 500 milisekúnd, je potrebné nastaviť prah na 15 snímok s chybnou detekciou. Ako optimálna hodnota však stačí aj hodnota s počtom 10 chybných snímkov v rade. Po prekročení tejto hodnoty program vyhodnotí výsledok ako chybový, teda postava sa v obraze nenachádza, alebo jej poloha neumožňuje správnu detekciu. Takéto upozornenie je v programe implemen- tované formou výstražného textu na obrazovke. Pri použití vo vozidle by bolo vhodné najmä zvukové upozornenie, ktoré by vodič nemohol prehliadnuť. Pri ignorovaní tohto varovania by vozidlo vykonalo napríklad brzdový manéver vzhľadom na okolitú premávku. Z výsledku týchto metód je možné odhadnúť napríklad, či vodič správne drží volant, alebo používa riadiacu páku. Ak vodič nevykonáva žiadnu z týchto činností, pravdepodobne nevenuje dostatočnú pozornosť riadeniu vozidla. Takáto situácia je veľmi nebezpečná a môže viest až k dopravnej nehode. Vy- užitie výsledkov týchto metód bude slúžiť najmä na vyhodnotenie vodičovej pozície pomocou natrénoanej neurónovej siete. Tejto analýze a vyhodnoteniu vodičovho správania je venovaná kapitola 5.5. Pri správnom natočení kamery na vodiča a normálnom posede bez prudkých po- hybov, by postava mala byť správne detekovaná aj pri veľmi slabých svetelných podmienkach. Pri jazde v noci však postavu už bez prídavného osvetlenia nie je možné detektovať.

5.5 Analýza správania vodiča

Pri predchádzajúcich kapitolách boli zbierané rôzne druhy informácií o polohe vodiča, ktoré je možné ďalej využiť a analyzovať. Pomocou knižníc a metód na detekciu postáv je možné získať napríklad pozíciu hlavy alebo pozície jednotlivých končatín. Tieto informácie sa dajú napríklad znázorniť na vstupnom obrázku. Z takejto volby však nie je možné ďalej analyzovať samotnú polohu vodiča a slúži iba na informatívne účely. Hlavná myšlienka tejto kapitoly spočíva v ďalšom spracovaní zozbieraných informácií a dát. Pomocou získaných údajov je možné napríklad vyhodnotiť, či vodič sedí za volantom v správnej polohe, prípadne kontrolovať základné pohyby, ktoré môže za volantom vykonávať. Pri sediacej polohe vo vozidle môže vodič vykonávať určité druhy pohybov. Tieto pohyby môžu byť súčasťou vedenia vozidla, ale zároveň to môžu byť aj pohyby, ktoré nemusia mať priamy súvis s vedením vozidla. Z tohto dôvodu boli analyzované rôzne polohy vodiča, ktoré je možné rozdeliť do niekoľkých skupín:

- **Držanie volantu** - vodič sedí vzpriamene a drží alebo točí volantom. To znamená, že obidve horné končatiny držia volant. Pri tomto pohybe sa poloha rúk nachádza v dolnej pravej časti záberu. V práci je označovaná ako *STEERING*
- **Používanie riadiacej páky** - vodič používa jednu ruku na zmenu prevodových stupňov, samozrejme v závislosti na type prevodovky vozidla. Ruka sa zároveň nachádza v polohe, kedy nie je úplne zrejmé či je ruka položená na riadiacej páke, alebo používa iné periférie vozidla umiestnené na stredovej konzole vozidla. Medzi takéto periférie patrí napríklad ovládanie klimatizácie alebo rádia. (obr. 19). V práci je označovaná ako *SHIFTING*
- **Iná poloha** - okrem základných pozícii môže vodič sedieť aj v inej polohe a vykonávať netypický pohyb celým telom alebo končatinami. Takáto poloha však už nemusí zodpovedať správnej pozícii pri vedení vozidla. Vodič napríklad nemusí v tomto prípade držať volant, alebo môže byť vytočený do zadnej časti vozidla, čo môže byť počas jazdy veľmi nebezpečné. V takomto prípade sa jedná o polohu, ktorá bude hodnotená ako nebezpečná. V práci je označovaná ako *WRONG*

Po zadefinovaní skupín poloh vodiča prichádza na rad, akým spôsobom sa pozície budú vyhodnocovať. Jedným zo spôsobov je napríklad sledovať horné končatiny vodiča. Pomocou zozbieraných dát je možné sledovať ich pozíciu alebo natočenie v rámci detektovanej postavy. Takéto riešenie je síce jednoduché, ale prináša určité nevýhody. Vyhodnocovanie polohy takýmto spôsobom by sa líšilo vzhľadom na rôzne faktory. Výsledok by závisel od umiestnenia kamery, ale aj napríklad výšky vodiča prípadne preferencií posedu človeka za volantom. Niektorí vodiči uprednostňujú nižšie nastavené sedadlo, niektorí vodiči zase vyššie nastavenie sedadla pre lepší výhľad z vozidla. Pri takýchto polohách sa zároveň mení aj uhol, ktorý ruky zvierajú. Z tohto dôvodu je preto lepšie použiť obecný klasifikátor vytvorený pomocou konvolučnej neurónovej siete.



(a)

(b)

Obr. 19: Minimálny rozdiel detekcie postavy v rôznych situáciach - používanie riadiacej páky (a), obsluha klimatizácie alebo rádia (b).

Popis neurónovej siete

Na vyhodnocovanie pozície vodiča vo vozidle je v programe použitá konvolučná neurónová sieť, ktorá dokáže polohu vodiča klasifikovať do jednotlivých tried. Na implementáciu je použitá knižnica Keras, ktorá je dostupná aj pre jazyk Python. Keras je jednoduchý nástroj postavený na knižnici Tensorflow. Slúži na vytváranie neurónových sietí s jednoduchou možnosťou úpravy jednotlivých vrstiev a rôznych parametrov. Podstata neurónovej siete spočíva vo vstupe, ktorý sa transformuje cez niekoľko definovaných tzv. skrytých vrstiev. Vo výstupnej vrstve sa nachádza výsledok vo forme tried so zodpovedajúcou pravdepodobnosťou danej triedy. Aby siet fungovala správne, je potrebné ju natrénovať na datasete ohodnotených vstupov. Používanie neurónovej siete na vyhodnotenie pozície vodiča sa skladá z nasledujúcich častí:

- **Predspracovanie dát** - všetky vstupy pred spracovaním je potrebné najskôr upraviť do vhodnej podoby, aby neurónová siet bola schopná s týmito údajmi pracovať a zároveň testovacie dátá boli v rovnakom formáte ako trénovacie dátá.
- **Model** - každá neurónová siet obsahuje model, ktorý predstavuje jej jadro. To znamená, že model špecifikuje počet a druh skrytých vrstiev v neurónovej sieti, ich veľkosť, vstupnú a výstupnú vrstvu.
- **Optimalizácia** - pre správne fungovanie siete je potrebné vytvoriť optimalizátor, ktorý určuje rýchlosť učenia. Počas trénovania sa upravuje hodnota *loss*, od ktorej závisí pokročenosť naučenia neurónovej siete.

- **Fit model** - predstavuje tréningový krok neurónovej siete. Definuje počet iterácií, počas ktorých sa bude sieť trénovať.

Na začiatok je potrebné vytvoriť vhodný formát datasetu, na ktorom sa neurónová sieť bude trénovať. Kedže podobné datasety vytvorené sférickou kamerou v súčasnosti nie sú veľmi dostupné, je potrebné si takýto dataset vytvoriť. Vytvorený dataset sa skladá z množiny obrázkov, ktoré obsahujú postavu vodiča a bod reprezentujúci hlavu. Pre jednotný format sa detekovaná postava vykreslí na čierne pozadie a uloží ako miniatúrny obrázok. Veľkosť obrázka závisí aj na rýchlosťi trénovania. Z tohto dôvodu bola šírka zvolená na 64 pixelov a výška sa automaticky prispôsobí podľa tvaru postavy. Obrázky boli zbierané z viacerých videí a rozdelené do troch vyššie uvedených tried. Ukážku jednotlivých datasetov je možné vidieť na obrázkoch 20, 21 a 22. Takýmto spôsobom bolo zozbieraných 11 957 trénovacích obrázkov. Rozpis pre jednotlivé triedy je znázornený v tabuľke 7.



Obr. 20: Ukážka trénovacieho datasetu - držanie volantu (STEERING)



Obr. 21: Ukážka trénovacieho datasetu - používanie riadiacej páky (SHIFTING)



Obr. 22: Ukážka trénovacieho datasetu - nesprávna pozícia (WRONG)

Trieda datasetu	Počet trénovacích obrázkov
Používanie radiacej páky	2985
Držanie volantu	2991
Nesprávna pozícia	5981

Tabuľka 7: Počet trénovacích obrázkov pre jednotlivé triedy

Po úspešnom vytvorení trénovacieho datasetu je potrebné siet správne natrénovať. Tento krok je veľmi dôležitý a zlé natrénovanie siete vedie k neúspechu celej detekcie. Je potrebné zvoliť správne vrstvy a ich počet, nakoľko sa od toho odvíja úspešnosť natrénovania siete. Aby sa priebežne testovala úspešnosť siete, je vhodné z datasetu určité percento obrázkov zvoliť na testovanie. Z tohto dôvodu bola zvolená hodnota 10%, ktorá je pre informatívne účely postačujúca. Aj keď sa na prvý pohľad môže zdať, že trénovacie vstupy vyzerajú rovnako, líšia sa v určitých typických vlastnostiach, ktoré sú typické pre celú triedu. Jedným z takýchto znakov je napríklad umiestnenie pravej ruky pri porovnaní medzi datasetom na obrázku 20 a 21. Aby sa všetky tieto znaky identifikovali správne, je potrebné pridať niekoľko skrytých vrstiev, ktoré tieto znaky zachytia. Model siete sa skladá zo vstupnej vrstvy, 4 skrytých vrstiev a výstupnej vrstvy, ktoré sú vizualizované na obrázku 24. Funkcia jednotlivých vrstiev v modeli je nasledujúca:

- **Vstupná vrstva** - jedná sa o prvú vrstvu na spracovanie trénovacích obrázkov. Na vstupe bola použitá konvolučná vrstva, ktorá sa skladá z 32 filtrov s veľkosťou 3×3 . Jej hlavnou úlohou je previesť fyzický obrázok na dátu, ktoré sú schopné spracovať nasledujúce vrstvy siete.
- **1. skrytá vrstva** - skladá sa z konvolučnej vrstvy s použitím 64 filtrov s rovnakou veľkosťou ako pri predchádzajúcej vrstve. Jej hlavnou úlohou je zvýšenie úspešnosti detekcie.
- **2. skrytá vrstva** - jedná sa o redukovaciu vrstvy, ktorej hlavnou úlohou je zníženie počtu dimenzií z predchádzajúcich vrstiev. Aby nedochádzalo k zbytočne vysokej náročnosti trénovania siete. Tieto dimenzie sa môžu redukovať na základe rôznych hodnôt. Pri vytváraní modelu siete bola použitá vrstva *MaxPooling2D* s veľkosťou filtra 2×2 . Ukážka fungovania redukovacej vrstvy je znázornená na obrázku 23.
- **3. skrytá vrstva** - medzi konvolučnou vrstvou a plne prepojenou vrstvou typu *Dense* je použitá vrstva *Flatten*, ktorá transformuje dvojrozmernú maticu príznakov na vektor, ktorý nasleduje do klasifikátora neurónovej siete.
- **4. skrytá vrstva** - reprezentuje plne prepojenú vrstvu (angl. fully connected layer). Každý vstupný uzol je pripojený ku každému výstupnému uzlu.
- **Výstupná vrstva** - výstupná vrstva sa skladá z 3 filtrov, ktoré reprezentujú jednotlivé triedy objektov s príslušnou hodnotou. Ako aktivačná funkcia je použitá funkcia *Softmax*. Je to dôležitá aktivačná funkcia, pretože nielen mapuje výstup do rozsahu $[0, 1]$, ale tiež mapuje každý výstup tak, že celkový súčet bol rovný 1. Výstupom *Softmax* je preto rozdelenie pravdepodobnosti jednotlivých tried objektov.

Knižnica Keras obsahuje nástroje, pomocou ktorých je možné model vytvoriť veľmi jednoducho. Vrstvy sa pridávajú v poradí, v akom sa majú vykonat. Ukážka modelu siete v jazyku Python je znázornená vo výpise 3.

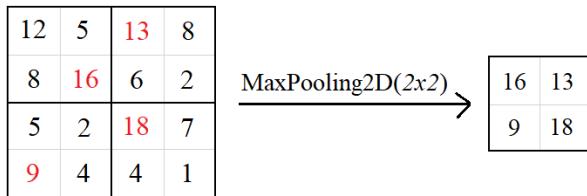
```

model = keras.Sequential([
    keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu',
        input_shape=(size, size, 1)),
    keras.layers.Conv2D(64, kernel_size=(3, 3), activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(3, activation='softmax')
])

```

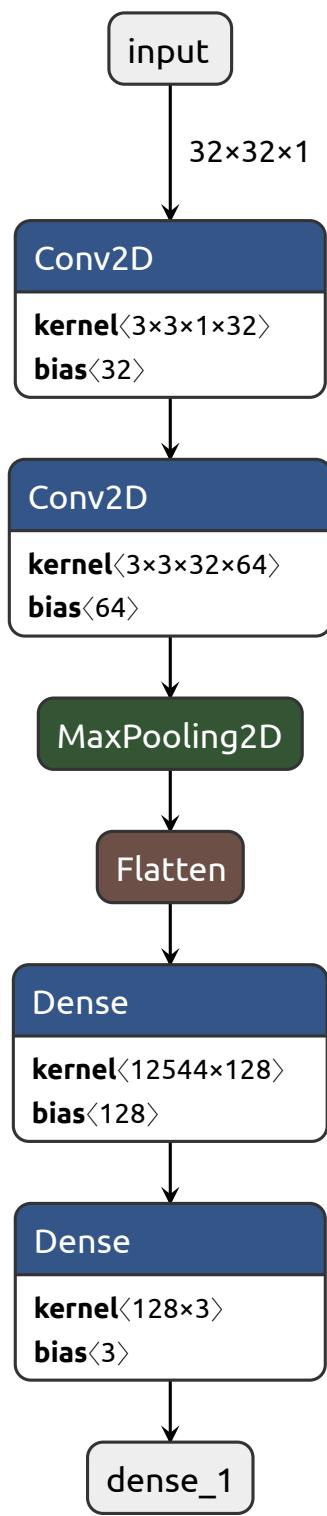
Výpis 3: Vrstvy neurónovej siete

Po vytvorení modelu nasleduje krok s trénovaním siete. Je potrebné zvoliť správne množstvo iterácií (angl. epochs), aby sa model natrénoval s dostatočne vysokou presnosťou. Ak by tento parameter bol príliš nízky, znamenalo by to nedostatočné natrénovanie a presnosť siete by nebola dostatočne vysoká. Naopak, po určitom počte trénovacích iterácií je možné pozorovať určitú stágnáciu v úspešnosti siete a v parametri straty (angl. loss). Pokiaľ sa tieto parametre ďalšou iteráciou už nemenia výraznejším tempom, je potrebné trénovanie ukončiť. Po odsledovaní týchto hodnôt bol počet iterácií zvolený na 12 a spracovanie v dávkach o veľkosti 128. Jednotlivé vrstvy a parametre siete boli priebežne upravované a výsledky trénovania zaznamenané. Model siete je znázornený na obrázku 24.



Obr. 23: Ukážka fungovania redukčnej vrstvy typu *MaxPooling2D*

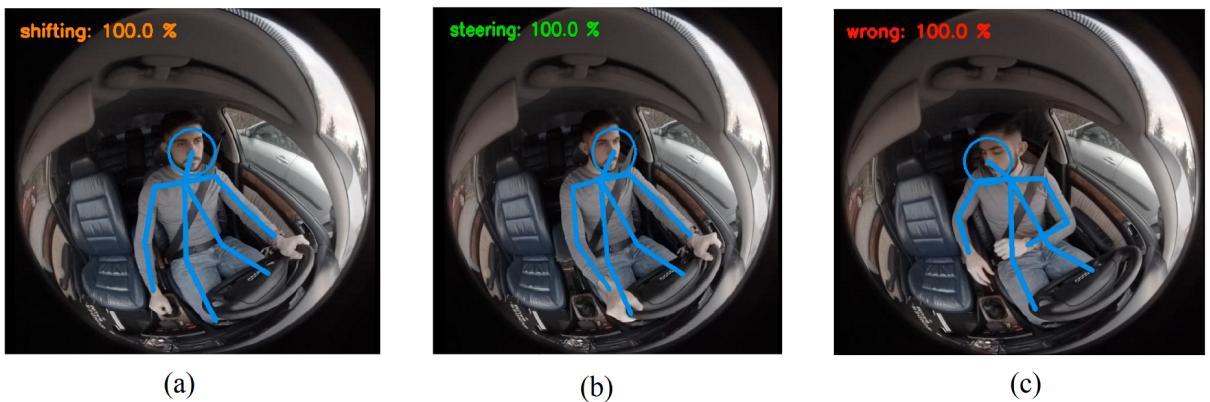
Priebeh trénovania modelu je priebežne vypisovaný v konzole. Z tohto výpisu je možné zistiť rôzne informácie ako napríklad aktuálnu úspešnosť, stratu či aktuálnu iteráciu. Pri použitých parametroch modelu sa dĺžka trénovania pohybovala v intervale 20 až 25 minút. Táto doba nie je veľmi vysoká najmä z dôvodu, že sú použité veľmi jednoduché obrázky, na ktorých sú príznaky ľahko a rýchlo rozpoznateľné. Ďalším možným dôvodom môže byť aj nedostatočne veľký počet obrázkov v trénovacom datasete. Po skončení sa výsledná úspešnosť trénovania pohybovala v intervale 75 až 94%. Táto hodnota je však iba informatívna a pochádza z priebežne testovanej vzorky náhodne zvolených 598 obrázkov z trénovacej množiny obrázkov. Pre zistenie skutočnej úspešnosti je potrebné vykonať testovanie na ďalších testovacích videách. Po skončení trénovania je model uložený do súboru, aby mohol byť použitý aj neskôr.



Obr. 24: Vizualizácia modelu siete pomocou nástroja Netron [11]

Spracovanie výsledkov siete

Po úspešnom natrénovaní modelu prichádza na rad jej použitie. Aby sa výsledok priblížil k čo najvyššej úspešnosti, je potrebné každý obrázok pred vstupom do siete predspracovať. Jednotlivé snímky z videa sú modifikované podobne ako pri trénovaní siete. Dôležitým krokom je rovnako aj úprava rozlíšenia, pretože framework na detekciu postáv je možné používať v rôznych rozlíšeniach. Aby takáto zmena rozlíšenia nespôsobovala problémy, každý snímok je upravený na rozlíšenie 32×32 pixelov. Výsledok neurónovej siete predstavuje percentuálne ohodnenie pravdepodobnosti pre samotné triedy polohy. Ako finálna trieda je následne určená trieda s najvyššou hodnotou pravdepodobnosti. Názov triedy aj s hodnotou pravdepodobnosti je vykreslený do vstupného obrázku s vodičom (obr. 25).



Obr. 25: Výsledok detekcie neurónovej siete

Pri vyhodnocovaní boli zistené viaceré nedostatky detekcie. Veľkým problémom, ktorý sa vyskytoval pri detekcii bola častá zmena tried ale s prevahou skutočnej polohy. To znamená, že ak vodič skutočne držal volant a šoféroval, v priebehu 100 snímok bola poloha držania volantu detekovaná v priemere $85\times$. Zvyšných 15 detekcií predstavovalo ostatné triedy, ktoré sa v náhodnom intervale určili ako výstup siete. Pri malom FPS je táto zmena nepostrehnutelná. Problém nastáva, ak sa FPS pohybuje okolo 30 snímok za sekundu, čo je bežný štandard dnešných videí. Vtedy nastáva nepríjemné blikanie nápisu triedy v ľavom hornom rohu videa, ktoré má rušivý efekt. Keďže sa jedná o náhodnú chybu, ktorá sa objavuje nepravidelne, je potrebné nájsť spôsob ako ju eliminovať. Z tohto dôvodu je potrebné použiť filtrovanie na zamedzenie podobných problémov. Jedným z ideálnych kandidátov na vyriešenie tohto problému je napríklad použitie Kalmanovho filtra [31], ktorý dokáže takéto nedostatky v súvisom toku hodnôt úspešne odfiltrovať. Kalmanov filter je možné použiť aj v rôznych iných oblastiach ako napríklad odstránenie šumu zo signálu a podobne. Pre implementačnú náročnosť však v tejto práci neboli použití. Na odfiltrovanie chybných hodnôt je použitý jednoduchý filter, ktorého výsledok zavisí na predchádzajúcich hodnotách. Ak sa výsledok siete zmení, musí sa zmeniť niekoľkokrát za sebou podľa nastavenej prahovej hodnoty. Po viacerých testovaniach výsledkov bola prahová hodnota zvolená

na hodnotu 5. To znamená, že výsledok filtra sa nezmení, pokiaľ do filtra nevstúpi iná hodnota $5 \times$ za sebou. Implementácia filtra v jazyku Python je znázornená vo výpise 4.

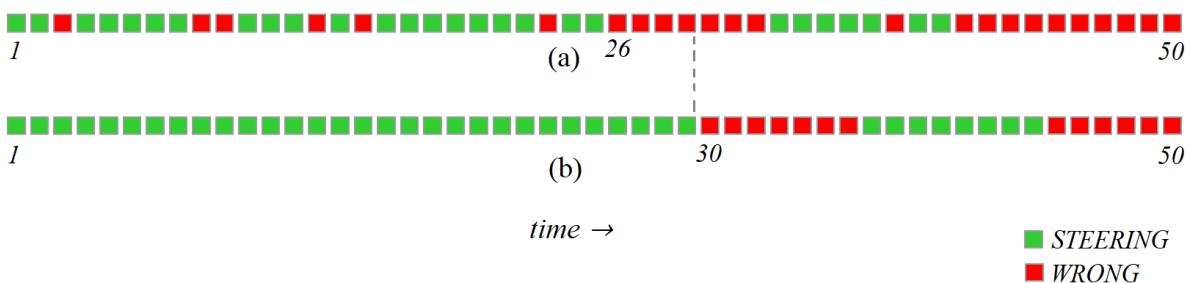
```

nn_count_threshold = 5
prev_class_name = None
def nn_filter(class_name):
    if prev_class_name is None or class_name == prev_class_name:
        # nothing changed
        nn_counter = 0
        prev_class_name = class_name
    else:
        # result changed, start counting different results
        nn_counter += 1
        # if count is higher than threshold, new value is applied
        if nn_counter > nn_count_threshold:
            nn_counter = 0
            prev_class_name = class_name
    return prev_class_name

```

Výpis 4: Jednoduchý filter na odstránenie náhodných hodnôt s použitím prahovej hodnoty

Použitie filtrácie je užitočné, ale prináša so sebou aj určité negatíva. Hlavný problém, ktorý nastáva je posun filtrovaných dát v čase. Filter odstraňuje nesprávne hodnoty, ktoré vznikajú náhodne počas detekcie. Skutočná hodnota na výstupe sa zmení až po prekročení prahu, ktorý je nastavený na 5 snímkov. Na obrázku 26 je zobrazená ukážka výsledku detekcie neurónovej siete v čase. Zelená farba predstavuje triedu *STEERING* - teda používanie volantu. Na obrázku je možné vidieť aj červenú farbu, ktorá reprezentuje triedu *WRONG* - nesprávna pozícia šoféra. Na príklade je znázornená funkčnosť filtra, ktorý úspešne odstraňuje náhodné chyby siete. Pri reálnej zmene triedy je možné vidieť vofiltrácii posun, ktorý predstavuje 5 snímkov. Hodnota detekovaná na snímku číslo 26 sa pri filtre prejaví až na snímku č. 30. Táto hodnota je však zanedbateľná, pretože pri bežnom videu natočenom pri 30FPS tento posun predstavuje približne 166ms. Takýto časový posun je v tomto prípade zanedbateľný.



Obr. 26: Ukážka hodnôt neurónovej siete - bez filtrácie(a), s použitou filtráciou (b)

Meranie úspešnosti

Na meranie úspešnosti boli použité viaceré vzorky rôznych typov videí. Hlavným zámerom bolo použiť čo najmenší počet snímok z videí na ktorých sa model siete trénoval. Tým pádom je zaručené, že testovanie bude prebiehať na iných obrázkoch. Meranie úspešnosti bolo v prvej časti vykonané na troch videách, pričom každé video reprezentuje jedinú triedu polohy. Pri použití takéhoto videa je veľmi jednoduché spočítať úspešnosť P_i pre danú triedu. Výpočet na úspešnosť je znázornený vo vzorci 6. Hodnota r_i predstavuje počet snímok videa, na ktorých je detekovaná poloha zodpovedajúca aktuálnemu videu. Parameter r_{total} je počet všetkých snímok vo videu.

$$P_i = \frac{r_i}{r_{total}} \times 100 \quad (6)$$

Meranie úspešnosti videa, ktoré obsahuje všetky typy pozícii nie je možné takýmto spôsobom vykonať. Práve z tohto dôvodu bolo spracované manuálne. Meranie úspešnosti takýmto spôsobom je však časovo veľmi náročné a preto je vhodné ho použiť iba pri malej testovacej množine. Video je dlhé 98 sekúnd, čo predstavuje približne 5900 snímok celkovo, pretože bolo vytvorené pri použití 60FPS. Pri takomto meraní však treba brať do úvahy aj rôzne prechodné stavy, kedy vodič napríklad púšta volant a rukou sa približuje ku riadiacej páke. Takýto stav nie je jednoznačný a preto neboli v meraní úspešnosti zohľadnený. Výsledná úspešnosť je zhrnutá v tabuľke 8.

Typ videa	Úspešnosť bez filtrácie [%]	Úspešnosť s filtráciou [%]
Používanie volantu	97.51	98.96
Držanie riadiacej páky	77.83	81.65
Nesprávna pozícia	86.41	87.75
Kombinované	81.25	83.84

Tabuľka 8: Úspešnosť vyhodnotenia správania pomocou neurónovej siete

Pri trénovaní a testovaní však treba brať do úvahy aj umiestnenie kamery. Ak by kamera zmenila umiestnenie napríklad ku dverám, detekcia polohy bude úspešná, ale vyhodnotenie polohy neurónovou sieťou už úspešné nebude. V takom prípade je nutné sieť natrénovať odznovu.

Analýzou pomocou tohto klasifikátora je možné zistiť typ polohy vodiča za volantom a určiť záver, či sa vodič riadne venuje riadeniu vozidla. Tento údaj môže byť informatívny, ale môže byť použitý aj pri ďalších detekciách, ktoré budú analyzovať vodiča podrobnejšie.

5.6 Detekcia orientácie hlavy

Analýza správania vodiča za volantom je veľmi rozsiahla komplexná téma. Okrem detektie pozície vodiča za volantom, ktorá je spracovaná v predchádzajúcej kapitole je možnosť skúmať ľudské správanie aj z rôznych ďalších zdrojov. Samotná pozícia síce napovedá, ako vodič sedí a či správne drží volant, avšak stále z nej nie je možné odvodiť veľa dôležitých informácií. Medzi najčastejšie príčiny nehôd v nočných hodinách patrí do vysokej miery mikrospánok. Podľa štúdii [32], ktorú vykonali McCartt, Ribner, Pack a Hammer v roku 1996 približne 55% z 1000 opýtaných vodičov uviedlo, že jazdili s miernymi príznakmi mikrospánku a 23% zaspalo dokonca na pár sekúnd počas jazdy. To potvrdzuje ďalšie zistenia, že ospalosť môže zohrávať významnú úlohu pri zrážkach, ktoré sú chybne pripisované iným príčinám. Vznik mikrospánku je zvyčajne spôsobený únavou ľudského tela a človek si veľakrát ani nedokáže pripustiť, že je unavený. Takéto podcenenie situácie zvyčajne končí krátkodobým mikrospánkom vodiča, ktoré je veľmi nebezpečné aj pre ostatných účastníkov premávky. Pri samotnom mikrospánku dochádza k viacerým kľúčovým okolnostiam, ktoré je možné včas odhaliť. Okrem straty pozornosti a vnímania okolia často dochádza aj k zatvoreniu očných zreničiek alebo prudkému pohybu hlavy smerom nadol. Pred týmito udalosťami je možné napríklad detektovať očné zreničky, alebo natočenie hlavy. Ľudské zreničky sú vzhľadom k veľkosti tela veľmi malé. Obzvlášť pri použití širokouhlnej alebo sférickej kamery nie je možné v obrazu detektovať zreničky s dostatočne vysokou presnosťou, či dokonca detektovať ich otvorenie a zatvorenie. Skúmanie stavu zreničiek je špecifická téma a nie je predmetom tejto práce, aj keď môže slúžiť ako významný spôsob pre analýzu ľudského správania a predikovania ospalosti vodiča. Sférickou kamerou je ale možné zachytiť celú tvár v dostatočnej kvalite pre ďalšie spracovanie. Z týchto dôvodov je táto kapitola primárne na orientáciu a otáčanie hlavy. Pri prudkom otočení hlavy nadol je možné predikovať, že človek napríklad stratal vedomie alebo zaspal. Vodič sa zároveň často otáča do strán hlavne pri cúvaní alebo pri státí na križovatke pre získanie rozhľadu v okolí vozidla. Pri priamej jazde však vodič nemá žiadny dôvod hlavou otáčať alebo výraznejšie meniť jej pozíciu. Z tohto dôvodu do programu bola pridaná detekcia na orientáciu hlavy vodiča. Hlavný spôsob spočíva v detektii kľúčových bodov tváre pomocou natrénovaných modelov. Na detekciu bol použitá funkcia z knižnice *dlib*, ktorá je založená na použití histogramov orientovaných gradientov (kap. 3.1) s použitím SVM. Metóda sa značí vysokou rýchlosťou a presnosťou detektie. Hlavnou metódou detektie je *get_frontal_face_detector()* z knižnice *dlib*, ktorá slúži na detekciu tvári. Pre túto implementáciu bol zvolený existujúci natrénovaný model *shape_predictor_68_face_landmarks*, ktorý je dostupný na webovej stránke modelov knižnice *dlib* [33]. Tento model umožňuje detektovať až 68 bodov tvoriacich tváru. Tieto body sú znázornené na obrázku 27. Výstup tejto detektie je možné využiť viacerými spôsobmi. Jedným spôsobom je napríklad detekcia orientácie tváre vzhľadom na pozíciu kamery. Pri správnej analýze je vo výsledku možné odhadnúť, či vodič sleduje vozovku, alebo jeho hlava je vytočená do inej strany. Aby bolo možné odhadnúť smerovanie tváre, je potrebné tváru transformovať do 3D súradníc.



Obr. 27: Detekcia tváre pomocou landmarkov. Označenie bodov (a), detekcia na reálnej tvári vodiča (b)

Problém odhadu pozície z obrazu do pozície reálneho sveta sa často označuje ako problém perspektívneho n-bodu alebo PNP. Podobnému zisťovaniu natočenia hlavy sa venovalo mnoho autorov. Jeden z článkov [34], kde sa autor venoval transformácii 2D tváre do 3D projekcie spočíva v detekcii landmarkov a následnej transformácii pomocou referenčných bodov. Cieľom práce autora bolo nájsť pozíciu objektu, ktorý je zachytený kamerou, zistiť umiestnenie 3D bodov na objekte a zodpovedajúcu 2D projekciu v obraze, ktorú následne dokázal znázorniť. Podobný spôsob zisťovania natočenia hlavy je možné využiť aj na detekciu otáčania hlavy vodiča vo vozidle. Po zistení tejto informácie je možné úroveň otočenia hlavy spracovať ďalej. Pri pohybe objektu vzhľadom na kameru môžu nastať 2 rôzne druhy pohybov:

- **Posun** - pohyb kamery z jej aktuálnej 3D pozície (X, Y, Z) na novú 3D pozíciu (X', Y', Z') sa nazýva posun. Tieto 3 súradnice pozície sa nazývajú aj 3 stupne voľnosti, pretože kamera sa môže pohybovať v 3 rôznych smeroch. Posun predstavuje vektor, ktorý sa rovná rozdielu starej a novej pozície: $(X' - X, Y' - Y, Z' - Z)$.
- **Rotácia** - kamera sa môže otáčať aj okolo osí X, Y a Z . Rotácia má preto tiež tri stupne voľnosti. Existuje mnoho spôsobov, ako reprezentovať rotáciu. Môže sa reprezentovať pomocou Eulerových uhlov (pretočenie, stúpanie a vybočenie), matice rotácie 3×3 alebo podľa smeru rotácie a uhla.

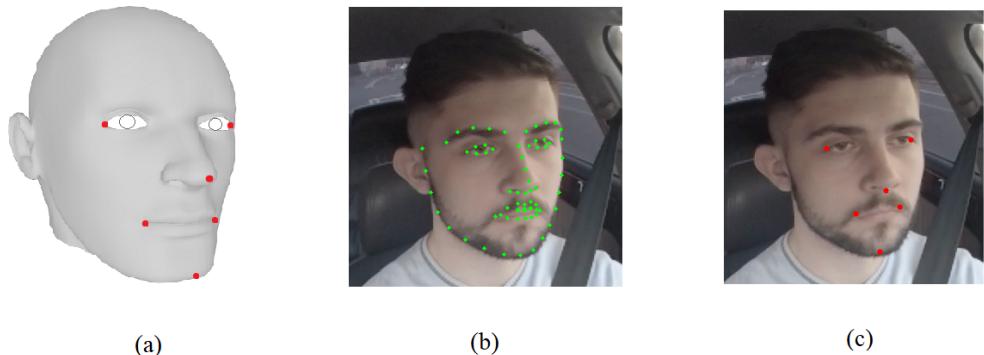
Určenie pozície 3D objektu znamená nájsť 6 hodnôt - tri pre posun a tri pre rotáciu objektu. Po získaní týchto hodnôt je ďalej možné 3D body transformovať do svetových súradníč na 3D body v súradničiach kamery. 3D body v súradničiach kamery môžu byť premietané do obrazovej roviny (súradnicový systém obrazu) pomocou vnútorných parametrov kamery (ohnisková vzdialenosť, optický stred atď.). Na výpočet 3D pozície objektu v obraze sú potrebné nasledujúce informácie:

- **2D súradnice niekoľkých bodov** - na lokalizáciu 2D bodov pozície (x, y) je potrebné vybrať niekoľko dostatočne významných bodov na obrázku. V prípade tváre je možné zvoliť

napríklad krajné rohy očí, špičku nosa, rohy úst. Tieto hodnoty nám poskytuje metóda na detekciu landmarkov, ktorá ako výsledok vráti pozíciu jednotlivých bodov.

- **3D súradnice rovnakých bodov** - aby sa dokázali body z obrazu transformovať, je potrebné nájsť správny spôsob transformácie. Táto transformácia je založená na využití reálneho 3D modelu tváre, z ktorého sa vyberú referenčné body. Tieto body musia mať k dispozícii súradnice v 3D sústave. Pre tvár sa zvolia referenčné body uvedené vyššie a sú znázornené na obrázku 28. Pre tieto body existuje niekolko vypočítaných modelov. Nasledujúca skupina bodov predstavuje pozíciu bodov, ktoré sú použité na transformáciu v 3D priestore:

1. Špička nosa $P = (0, 0, 0)$
2. Brada $P = (0, -330, -65)$
3. Lavý roh ľavého oka $P = (-225, 170, -135)$
4. Pravý roh pravého oka $P = (225, 170, -135)$
5. Lavý roh úst $P = (-150, -150, -125)$
6. Pravý roh úst $P = (150, -150, -125)$



Obr. 28: Detekcia tváre pomocou landmarkov. Znázornenie referenčných bodov tváre v 3D [12] (a), detekcia na reálnej tvári vodiča (b), výber referenčných bodov v 2D (c)

Po zistení 3D súradníc však stále ostáva neznáma matica rotácie R o veľkosti 3×3 a posun t veľkosti 3×1 vzhladom na súradnice kamery. Ak by tieto hodnoty boli známe je následne možné vypočítať umiestnenie (X, Y, Z) bodu P v súradnicovom systéme kamery. Vzťah medzi týmito hodnotami je znázornený pomocou rovnice 7. Po úprave tejto rovnice vnikajú 2 neznáme vektory (r_{ij}) a (t_x, t_y, t_z) , ktoré sa dajú zistiť použitím správnej transformácie. Do úvahy je potrebné počítať aj s faktorom mierky s , keďže nie je zaručená konštantná veľkosť tváre na všetkých vstupných obrázkoch.

$$\begin{aligned}
\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} &= R \begin{bmatrix} U \\ V \\ W \end{bmatrix} + t \\
\Rightarrow \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} &= [R \mid t] \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix} \\
\Rightarrow \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} &= \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \end{bmatrix} \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix}
\end{aligned} \tag{7}$$

Na výpočet týchto neznámych je možné použiť napríklad priamu lineárnu transformáciu (angl. DLT). Na 3D modeli poznáme viaceru referenčných bodov (U, V, W), ale nie sú známe zodpovedajúce body (X, Y, Z) referenčných 2D bodov, u ktorých poznáme súradnice iba v 2D. Ak sa zanedbá miera radiálneho skreslenia, súradnice (x, y) pre ľubovoľný bod p v 2D projekcii sú znázornené v rovnici 8, kde f_x a f_y sú ohniskové vzdialosti v smere x a y a (c_x, c_y) je optický stred.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = s \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{8}$$

Túto rovnicu je možné vyriešiť pomocou metódy nazývanej priama lineárna transformácia (DLT). DLT je možné použiť, keď vznikne problém, v ktorom je rovnica takmer lineárna, ale jej výsledok je stále závislý napríklad od faktora mierky s . Použitie DLT nie je príliš presné z viacerých dôvodov. Po prvé, rotácia R má tri stupne voľnosti, ale maticová reprezentácia použitá v riešení DLT má 9 hodnôt. V riešení DLT nie je nič, čo zaručí prevod matice 3×3 na rotačnú maticu. Ďalšou nevhodou DLT je chýbajúca minimalizácia chýb. Aby výsledok neboli príliš skreslený a výsledný vektor zodpovedal skutočnému natočeniu tváre, je potrebné minimalizovať chyby, ktoré vznikajú v 2D projekcii.

Je dôležité zamerať sa na vzťah medzi premietanými 3D bodmi a 2D bodmi tváre. Ak je odhadovaná pozícia tváre natočená priamo, 3D body premietnuté do obrazovej roviny sa takmer dokonale zarovnajú s 2D bodmi tváre. Ak je odhad pozície nesprávny, je možné vypočítať mieru chyby opäťovného premietania. Ako bolo vyššie spomenuté, pomocou riešenia DLT je

možné nájsť približný odhad matíc R a t . Najjednoduchší spôsob, ako vylepšiť riešenie DLT, by bolo mierne zmeniť pozíciu R a t a skontrolovať, či sa chyba reprojekcie neznížila. Ak sa tak stane, môže sa priať nový odhad natočenia tváre. Aj keď tento postup bude fungovať, bude veľmi pomalý a časovo náročný. Na iteratívne zlepšenia tejto chyby vznikajúcej pri transformácii existuje viacero metód. Jedna takáto metóda sa nazýva Levenberg-Marquardtova optimalizácia [35], ktorá dokáže významne ovplyvniť výsledný čas spracovania.

Implementácia

Na vyriešenie problému perspektívneho n-bodu (PNP) existuje v OpenCV viacero metód, ktoré dokážu vypočítať pozíciu objektu vzhľadom na pozíciu kamery. V programe, ktorý má vytvoriť smerový vektor orientácie tváre je použitá funkcia `solvePnP()`. Funkcia implementuje niekoľko typov algoritmov na odhadovanie pozície, ktoré je možné zvoliť pomocou parametra `flag`. Štandardne používa príznak `SOLVEPNP_ITERATIVE`, čo je v podstate obyčajná lineárna transformácia nasledovaná optimalizáciou Levenberg-Marquardt. `SOLVEPNP_P3P` používa iba 3 body na výpočet pozície. Z tohto dôvodu je výpočet rýchly, ale aj malá odchýlka v jednom z týchto 3 bodov vedie k výraznému skresleniu a nie je vhodné ju v tomto prípade používať. Ako ďalší parameter je potrebné zadať maticu kamery, ktorú je potrebné vypočítať na základe veľkosti vstupného obrazu. Výpočet tejto matice je znázornený na výpise 5. Ďalší dôležitý parameter je nastavenie koeficientu skreslenia `distCoeffs`. Tento parameter je potrebné modifikovať najmä pri významnom zakrivení kraju obrazu pri použití širokouhlej kamery. Napriek tomu, že vodič je snímaný takouto kamerou, ku skresleniu dochádza iba pri krajoch videa. V strede obrazu, kde sa tvár vodiča nachádza je skreslenie minimálne a z tohto dôvodu je vhodné parameter nechať na predvolenú hodnotu. Výpočet rotačného a pozičného vektora je znázornený vo výpise 6.

```
# Camera matrix calculation
focal_length = input_image[1]
center = (input_image[1]/2, input_image[0]/2)
camera_matrix = np.array(
    [
        [focal_length, 0, center[0]],
        [0, focal_length, center[1]],
        [0, 0, 1]
    ], dtype="double"
)
```

Výpis 5: Výpočet matice kamery

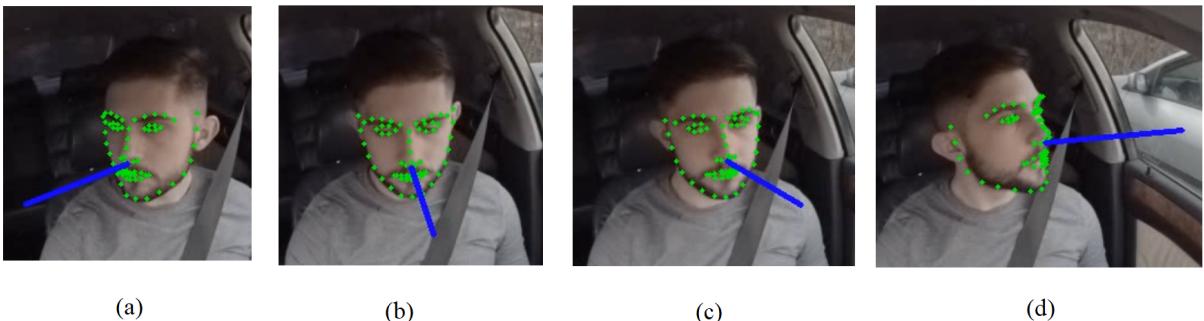
```

dist_coeffs = np.zeros((4, 1))
rotation_vector, translation_vector = cv2.solvePnP(model_points_3d,
    image_points, camera_matrix, dist_coeffs, flags=cv2.SOLVEPNP_ITERATIVE)
nose_end_point2D, jacobian = cv2.projectPoints(np.array([(0.0, 0.0, 1000.0)
]), rotation_vector, translation_vector, camera_matrix, dist_coeffs)

```

Výpis 6: rotačného a pozičného vektora

Výsledok metódy je realizovaný formou rotačného vektoru R a vektoru posunu t . Pre zobrazenie orientácie hlavy je dôležitý práve rotačný vektor. Aby bolo možné tento vektor znázorniť vo vstupnom obrázku, je potrebné určiť stred, odkiaľ sa vykreslí stredová priamka vyjadrujúca rotáciu tváre. Vhodným bodom je práve špička nosa, pretože nos sa nachádza približne v strede tváre. Na výpočet druhého bodu rotačnej priamky je použitá funkcia z OpenCV *projectPoints()*. Táto metóda berie ako prvý parameter bod v 3D priestore, voči ktorému sa rotácia určuje. Bod musí mať rovnaké súradnice (X, Y) ako bod špičky nosa. Posledná súradnica však je posunutá v priestore, aby bolo možné zostrojiť projekčný bod. Výsledkom funkcie sú súradnice bodu v 2D, ktoré reprezentujú premietnutý bod. Na záver je teda potrebné vykresliť priamku medzi bodom určujúcim špičku nosa a súradnice získaného bodu. Na testovanie a overenie funkčnosti boli použité viaceré natočenia tváre. Výsledok je znázorený na obrázku 29.



Obr. 29: Výsledok detektie orientácie tváre - pohľad vpravo (a), pohľad dolu (b), pohľad na vozovku (c), pohľad vľavo (d)

S použitím takejto detektie je možné odhadnúť úroveň natočenia tváre vodiča oproti kamere. Aby bolo možné vymedziť pohyb a rotáciu hlavy, je potrebné určiť povolený rozsah jej otáčania pri bežnej jazde. Detekciu otáčania hlavy treba samozrejme v špecifických prípadoch ignorovať. Každý vodič, ktorý parkuje sa často otáča okolo seba, aby získal prehľad okolo vozidla. V takomto prípade by detekcia nemala byť aktívna. Do úvahy tiež treba brať fakt, že kamera sa nenachádza priamo pred vodičom a vektor rotácie tváre má pri pohlade vodiča vpred určité vychýlenie od kamery (obr. 29c). Z tohto dôvodu je potrebné vymedziť možný uhol natočenia tváre. Na detekciu úrovne natočenia hlavy boli pridané pomocné čiary (obr. 30), ktoré zvierajú uhol približne 100

stupňov. Ten je dostatočný na rozhľad vodiča okolo svojho okolia. Treba brať do úvahy fakt, že vodič svojou polohou vzhľadom na volant mení aj zorný uhol pri jazde. Ak je vodič napríklad nízkeho vzrastu, je predpoklad, že bude mať sedadlo nastavené čo najbližšie ku volantu. Aby v takom prípade mal dostatočný rozhľad o svojom okolí počas jazdy, musí hlavou otáčať viac ako človek vysokého vzrastu, ktorý ma sedadlo posunuté dozadu. V takom prípade by bolo potrebné zmeniť povolený rozsah otáčania hlavy, alebo nájsť spôsob ako to urobiť automaticky napríklad detekciou vzdialenosť volantu od tela vodiča. Jedným zo spôsobov, ako takýto systém vylepšiť, je fixovať možný uhol natočenia hlavy na statické body, ako sú napríklad spätné zrkadlá. Tieto body je možné získať nejakou automatickou detekciou v obraze, alebo manuálnym vyznačením v obraze. Pri druhej možnosti však treba brať ohľad na to, že pozícia kamery nemusí byť stále rovnaká a systém by tým pádom nepracoval správne. Z dôvodu zložitosti program preto pracuje iba s jednoduchými pomocnými čiarami, ktoré základnú funkciu splňajú. Pozícia natočenia je znázornená aj do originálneho obrázku. Pri vytočení hlavy by reálny systém použitý vo vozidle mohol upozorniť vodiča napríklad zvukovým hlásením. Pri prekročení limitu otočenia hlavy, je kolmica vykreslená červenou farbou a v ľavom hornom rohu sa zobrazí upozorňovací text.



Obr. 30: Orientácia hlavy vodiča s pomocnými čiarami

5.7 Možnosti vylepšenia detekcie

Po úspešnom dokončení implementácie programu prišiel rad na detailnú analýzu výsledkov. Napriek rôznym prekážkam, ktoré sa objavovali je celková úspešnosť detekcie a analýzy pomerne vysoká. Počas tejto práce sa objavilo viacero problémov, ktoré výslednú kvalitu samotnej detekcie znížili. Ako bolo spomenuté v kapitole 5.4, frameworky na detekciu osôb dokážu telo úspešne zosnímať aj pri znížených svetelných podmienkach. Pri detekcii tváre však často dochádzalo k zlej, či dokonca chýbajúcej detekcii. Tento problém bol spôsobený tým, že knižnica *dlib* nedokázala nájsť kľúčové body tváre v zhoršených svetelných podmienkach a pri prudkom natočení hlavy do boku. Pri použití jednoduchého filtra, ktorý ukladal niekoľko predchádzajúcich výsledkov sa tento problém dokázal čiastočne eliminovať. Použitím filtra môže v takejto situácii vzniknúť malé oneskorenie, ktoré ale nepredstavuje veľký problém pri reálnej detekcii.

Ďalší problém, ktorý môže ovplyvniť výsledok je napríklad prostredie, druh vozidla, či samotná postava vodiča. Je potrebné si uvedomiť, že model trénovacej siete bol vytvorený na jedinej postave vodiča. Ak by ten istý model mal byť použitý pre vodiča, ktorý je napríklad nižší, alebo jeho posed vo vozidle sa nejakým spôsobom líší, výsledky by pravdepodobne neboli uspokojivé. Jednou z myšlienok, ako vylepšiť detekciu je vytvoriť systém, ktorý sa natrénuje pre každého vodiča zvlášt. To zabezpečí, že aj keď sa takýto systém použije s iným vodičom, nový natrénovaný model bude prispôsobený práve na tohto vodiča. Podobný problém predstavuje aj kabína vodiča a umiestnenie kamery. Aby bol takýto systém použiteľný, musí sa úspešne vyšporiadat aj so zmenou prostredia. Z tohto dôvodu by bolo vhodné vymyslieť spôsob kalibrácie, ktorý sa automaticky prispôsobí všetkým premenným faktorom ako je kabína či postava vodiča a jeho posed. Takúto kalibráciu je možné vykonať napríklad pri nástupe do auta. Systém by rozpoznal tvár vodiča a ak by neexistoval v systéme natrénovaný model, musela by sa previesť kalibrácia. Tá by spočívala v krátkom simulovaní jednotlivých jazdných úkonov, ako držanie volantu, zmena rýchlosného stupňa, alebo pozeranie sa do spätných zrkadiel. Program by sa na základe týchto informácií dokázal nakalibrovať automaticky a vodič by mohol pokračovať v jazde.

5.8 Zhrnutie výsledkov programu

V predchádzajúcich kapitolách bola rozpisana problematika a popis implementácie dvoch dôležitých častí pri skúmaní vodiča vo vozidle. Prvou časťou je detekcia a analyzovanie samotnej postavy vo vozidle. V druhej časti bola analýza smerovaná na detekciu otáčania hlavy vodiča. Tieto dve časti umožňujú čiastočne analyzovať vodičovo správanie a zistiť v akej miere sa venuje vedeniu vozidla. Vo výslednom programe sú použité obidve časti, ktoré je možné pomocou argumentov modifikovať alebo upraviť konkrétnie hodnoty programu. Celkový program, popis konfigurovatelných parametrov a dokumentácia je uvedená v repozitári diplomovej práce [36].

Pri použití obidvoch častí je možné vidieť celkovú činnosť programu, ktorý analyzuje postavu vodiča a jeho natočenie hlavy. Tieto časti sú dôležité prvky správania vodiča, od ktorých môže

závisieť samotná jazda a štýl vedenia vodiča. Napriek tomu, že program nedokáže fungovať v reálnom čase vo vozidle, môže poslúžiť ako základ alebo inšpirácia pre ďalšie práce zamerané na podobnú oblasť. Pri dlhodobom sledovaní parametrov a skúšaní rôznych kombinácií bolo zistených viacero dôležitých poznatkov. Na použitie detektie takého druhu je vhodné použiť knižnicu, ktorá dokáže fungovať aj na zariadeniach s menším výpočtovým výkonom. Je predpoklad, že ak by podobný systém bol v budúcnosti nainštalovaný vo vozidle, nebude mať k dispozícii výkonný hardware, ktorý by rýchlu detekciu umožňoval. Výrobcovia vozidiel musia bráť do úvahy aj cenu takýchto systémov a preto je potrebné nájsť rozumný kompromis ceny vzhľadom k celkovej úspešnosti systému. Z týchto dôvodov pripadá voľba na použitie framework TF Pose Estimation (kap. 5.4), ktorý dokáže fungovať aj na slabších grafických kartách s dostatočne vysokou úspešnosťou. Pri použití kvalitnej sférickej kamery, alebo kamery s dostačne širokým uhlom záberu môže výsledný systém poskytnúť nové možnosti detektie vodiča vo vozidle. Ukážka výsledného programu je znázornená na obrázku 31.



Obr. 31: Výsledná detekcia pohyby hlavy a vyhodnotenie správania vodiča

6 Záver

Vozidlá tvoria v dnešnej dobe významnú oblasť života ľudí. Dnes neslúžia už len ako dopravné prostriedky, ale mnoho ľudí vo vozidlách nachádza využitie voľného času, spoznávanie nových miest či oddych. Pri ich využívaní sa musí dbať na bezpečnosť posádky, ale aj bezpečnosť ostatných účastníkov premávky. Výrobcovia vozidiel vyvíjajú rôzne moderné systémy, ktoré dokážu asistovať a mnohokrát aj zastúpiť prácu vodiča. Tieto systémy sú každým dňom modernejšie a kvalitnejšie. Jedným zo spôsobov, ako analyzovať správanie vodiča v interiéri vozidla môže byť napríklad aj vizuálna detekcia rôznych klúčových aktivít počas vedenia vozidla. V práci boli najskôr preskúmané dostupné možnosti, ktorými je možné analyzovať aktivity posádky vozidla v súčasnosti. Takýto druh analýzy zatiaľ nie je veľmi rozšírený a poskytuje ho len veľmi úzke spektrum výrobcov vozidiel. Je to mnohokrát spôsobené vysokou cenou takýchto systémov, ale aj pomerne vysoké náklady na vývoj. V ďalšej kapitole bola popísaná problematika zachytávania obrazu v interiéri vozidla. Použitie detekcie obrazu v interiéri vozidla prináša rôzne prekážky, ktoré môžu viesť k celkovému neúspechu detekcie. Ako bolo zhrnuté v kapitole 4, je dôležité splniť mnoho dôležitých podmienok pre dosiahnutie požadovaných výsledkov detekcie. Jedným z faktorov, ktoré významne ovplyvňujú detekciu je napríklad zmena intenzity svetla či uhol záberu kamery, kvôli ktorému boli použité práve sférické kamery. Od týchto parametrov získaného obrazu závisia aj knižnice, ktoré postavu vodiča detekujú, či analyzujú. Následne bol vytvorený program, ktorý dokáže využiť výstup týchto knižníc a metód na detekciu postáv v obraze. Po úspešnom zozbieraní dát boli analyzované rôzne možnosti pohybu a správania vodiča vo vozidle. Po dôslednej analýze týchto dát a zvážení dostupných možností bola vytvorená neurónová sieť, ktorá vyhodnocuje vodičov posed a detektor natočenia hlavy, ktorý sa zameriava na to aby vodič sledoval premávku v čo najvyššej možnej miere. Tento druh použitej detekcie je dosť špecifický a môže samozrejme závisieť aj od druhu vozidla, alebo umiestením kamery. Napriek tomu výsledný program dokázal úspešne detektovať vodičovo správanie s pomerne vysokou mierou úspešnosti. Hlavným cieľom práce bolo tiež preskúmať dostupné možnosti, ktoré je možné v tejto oblasti využiť a vytvoriť ukážku toho, že použitie detekcie obrazu v automobilovej oblasti má veľký potenciál v budúcnosti.

Literatúra

- [1] Gabriel Nica. Bmw assisted driving view looks cool and useful, 2019. [Online; Citované 16.02.2020 z <https://cdn.bmwblog.com/wp-content/uploads/2019/09/bmw-assisted-driving-view.jpg>.
- [2] Paul Viola, Michael Jones, et al. Robust real-time object detection. *International journal of computer vision*, 4(34-47):4, 2001.
- [3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.
- [4] Massimo Bertozzi, Alberto Broggi, Mike Del Rose, Mirko Felisa, Alain Rakotomamonjy, and Frédéric Suard. A pedestrian detector using histograms of oriented gradients and a support vector machine classifier. In *2007 IEEE Intelligent Transportation Systems Conference*, pages 143–148. IEEE, 2007.
- [5] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1812.08008*, 2018.
- [6] Bryan Chung. Openpose in processing and opencv (dnn), 2018. [Online; Citované 23.02.2020 z <http://www.magicandlove.com/blog/2018/08/06/openpose-in-processing-and-opencv-dnn/>.
- [7] CMU-Perceptual-Computing-Lab. Deep pose estimation implemented using tensorflow with custom architectures for fast inference., 2018. [Online; Citované 10.04.2020 z <https://github.com/ildoonet/tf-pose-estimation>.
- [8] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. Rmpe - regional multi-person pose estimation. In *ICCV*, 2017.
- [9] Shigang Li. Full-view spherical image camera. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 4, pages 386–390. IEEE, 2006.
- [10] GoPro, 2016. [Online; Citované 6.03.2020 z https://gopro.com/en/cz/news/vr_spherical_omni_rig_shipping.
- [11] Lutz Roeder. Netron - visualizer for neural network, deep learning and machine learning models. Online <https://github.com/lutzroeder/netron>.
- [12] 3d male head model. Online <https://www.turbosquid.com/FullPreview/Index.cfm/ID/346686>, 2007.

- [13] Paul Smith, Mubarak Shah, and Niels da Vitoria Lobo. Determining driver visual attention with one camera. *IEEE transactions on intelligent transportation systems*, 4(4):205–218, 2003.
- [14] Azriel Rosenfeld. Picture processing by computer. *ACM Computing Surveys*, 1(3):147–176, Jan 1969.
- [15] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001.
- [16] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.
- [17] Yanwei Pang, Yuan Yuan, Xuelong Li, and Jing Pan. Efficient hog human detection. *Signal Processing*, 91(4):773–781, 2011.
- [18] CMU-Perceptual-Computing-Lab. Openpose: Real-time multi-person keypoint detection library for body, face, hands, and foot estimation, 2017. [Online; Citované 23.02.2020 z <https://github.com/CMU-Perceptual-Computing-Lab/openpose>.
- [19] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678, 2014.
- [20] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- [21] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.
- [22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [23] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.
- [24] AlphaPose. Real-time and accurate multi-person pose estimation and tracking system, 2017. [Online; Citované 26.02.2020 z <https://github.com/MVIG-SJTU/AlphaPose>.

- [25] Paul Kruszewski. Wrnch. Online <https://wrnch.ai/>.
- [26] Vikas Gupta. Pose detection comparison : wrnchai vs openpose, 2019. Online; Citované 03.03.2020 z <https://www.learnopencv.com/pose-detection-comparison-wrnchai-vs-openpose/>.
- [27] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.
- [28] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [29] François Chollet et al. Keras. <https://github.com/fchollet/keras>, 2015.
- [30] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.
- [31] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME-Journal of Basic Engineering*, 82(Series D):35–45, 1960.
- [32] Anne T McCartt, Stephen A Ribner, Allan I Pack, and Mark C Hammer. The scope and nature of the drowsy driving problem in new york state. *Accident Analysis & Prevention*, 28(4):511–517, 1996.
- [33] Davis King. Trained model files for dlib example programs. Online <https://github.com/davisking/dlib-models>.
- [34] SATYA MALLICK. Head pose estimation using opencv and dlib, 2016. Online; Citované 02.03.2020 z <https://www.learnopencv.com/head-pose-estimation-using-opencv-and-dlib/>.
- [35] MLA Lourakis and Antonis A Argyros. Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment? In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1526–1531. IEEE, 2005.
- [36] Repository for driver analysis using spherical cameras. Online https://github.com/michalfalat/SP_VSB.

Prílohy

I. Použitie programu

Program je navrhnutý tak, aby ho bolo možné spustiť cez príkazový riadok. Základné spustenie programu pre analýzu vodiča je vo výpise 7. Jednotlivé parametre programu sa zadávajú pomocou argumentov. Zoznam konfigurovatelných parametrov je v tabuľke 9.

```
python main.py --video-input testVideo.mp4
```

Výpis 7: Spustenie programu

Parameter	Popis parametru	Predvolená hodnota
--video-input	názov vstupného videa	testVideo.mp4
--framework	výber frameworku (OP_POSE, TF_POSE)	OP_POSE
--model-name	názov natrénovaného modelu	defaultModel.keras
--init-frame	nastavenie začiatočného snímku videa	0
--record-video	uloženie výsledného videa	-
--record-video-name	názov výsledného videa	output.mp4
--record-video-fps	FPS výsledného videa	30
--show-native-output	zobrazenie výsledku z frameworku počas detekcie	-
--resolution-height	výška videa pre spracovanie	800
--no-final-statistics	vypnutie výsledných štatistik	-
--no-continuos-statistics	vypnutie priebežných štatistik	-
--no-image-print-statistics	vypnutie informácií v obraze	-
--no-head-orientation-detection	vypnutie detekcie orientácie hlavy vodiča	
--no-head-orientation-limit	vypnutie rozsahu detekcie orientácie hlavy vodiča	-
--no-body-detection	vypnutie detekcie postavy vodiča	-
--no-showing-output	vypnutie priebežného výstupu programu	-
--no-nn-filtering	vypnutie filtrovania neurónovej siete	-
--save-train-image	uloženie snímky videa pre trénovanie	-

Tabuľka 9: Zoznam parametrov programu

Pre správne fungovanie programu je potrebné mať nainštalované všetky požadované nástroje a správne verzie knižníc. Počas práce boli použité nasledujúce verzie nástrojov a knižníc:

- Python 3.3
- OpenCV 3.4.3
- Tensorflow 1.15
- Keras 2.3.1
- dlib 19.17
- OpenPose 1.5.1
- TF Pose Estimation

II. Prílohy v IS EDISON

Súčasťou odovzdanej práce je aj súbor, ktorý obsahuje nasledujúce zložky:

- *sourcecodes* - zdrojové kódy vytvoreného programu
- *results* - ukážka videí z detekcie a analýzy vodiča vo vozidle.