

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analýza řidiče za pomocí sférických kamer

Driver Analysis Using Spherical Cameras

2020

Michal Falát

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

V Ostrave 1. apríla 2020

.....

Súhlasím so zverejnením tejto diplomovej práce podľa požiadaviek čl . 26, odst. 9 Študijného a skúšobného poriadku pre štúdium v bakalárskych programoch VŠB-TU Ostrava.

V Ostrave 1. apríla 2020

.....

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

Abstrakt

Hlavnou tému diplomovej práce je rozpoznávanie a analýza vodiča v aute pomocou sférických kamier. Táto práca je rozdelená do viacerých samostatných častí, ktoré na seba postupne nadvádzajú. Prvá časť spočíva v preskúmaní existujúcich riešení na detekciu postáv obrazoch s ich hlavnými vlastnosťami. Druhá časť je zameraná na analýzu a spracovanie obrazu pomocou sférických kamier. Posledná časť je venovaná samotnému programu na detekciu vodičov vo vozidlách a porovnanie jednotlivých metód na detekciu postáv a zhrnutie analýzy vodičovho správania za volantom počas jazdy.

Kľúčové slová: Sférická kamera, detekcia obrazu, analýza ľudskej tváre, detekcia ľudí, vodič, Openpose

Abstract

Main focus of this Diploma thesis is detection and analysis of driver in car with help of spherical cameras. This thesis is divided into few parts, which are connected to each other. The first part is about research of existing solutions for detecting pose estimation. The second part is focused on an analysis and processing video from spherical cameras. The last part is about final program for the detection of drivers in vehicles and the comparison of the various methods for the detection of person's pose and a summary of the analysis of driver behavior during drive.

Keywords: Spherical camera, image detection, analysis of human face, pedestrian detection, driver, Openpose

Obsah

Zoznam použitých skratiek a symbolov	8
Zoznam obrázkov	9
Zoznam tabuliek	10
Zoznam výpisov zdrojového kódu	11
1 Úvod	12
2 Detekcia a analýza ľudského tela v obrazoch	13
2.1 Haar	13
2.2 HOG	15
2.3 OpenPose	17
2.4 TF Pose Estimation	20
2.5 AlphaPose	22
2.6 Ostatné metódy	23
3 Detekcia vodiča vo vozidle	25
4 Využitie sférických kamier na detekciu obrazu	27
5 Vlastná implementácia	31
5.1 Popis programu	31
5.2 Požiadavky a návrh programu	32
5.3 Použité technológie	33
5.4 Vytvorenie a spracovanie videa	34
5.5 Detekcia vodiča	36
5.6 Orientácia hlavy	39
5.7 Bezpečnostný pás	40
5.8 Neurónová sieť	41
5.9 Výstup programu	42
5.10 Porovnanie výsledkov	43
5.11 Využitie zozbieraných dát	44
5.12 Používateľská príručka	45
6 Možnosti vylepšenia detektie	46
7 Záver	47

Zoznam použitých skratiek a symbolov

2D	– 2-dimensional
3D	– 3-dimensional
CNN	– Convolutional neural network
CNTK	– Microsoft cognitive toolkit
CPU	– Central processing unit
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	Výpočet integrálneho obrazu - vstupný obraz (A) , integrálny obraz (B)	13
2	Haar - dvoj-obdlžnikové príznaky (A, B), troj-obdlžnikové príznaky (C) a štvorobdlžnikové príznaky(D) [1]	14
3	Haar - detekcia tváre v slabých svetelných podmienkach	15
4	HOG - séria krokov [2]	16
5	HOG - vstupný obraz (a), normalizácia gradientu (b), orientácia gradientu (c), rozdelenie do buniek (d) vypočítané histogramy (e). [3]	16
6	OpenPose - odhad viacerých ôsob v reálnom čase pomocou polí affinity filtra. [4] .	17
7	COCO - označenie častí tela v COCO datasete. [5]	18
8	Tepelná mapa - vstupný obraz (a), tepelná mapa postavy (b)	21
9	Problém redundantnej detektie s použitím metódy bounding box [6]	22
10	WrnchAI - vstupný obraz (a), výsledok spracovania na platforme WrnchCloud(b)	24
11	BMW driving assistant - umiestnenie kamery na snímanie vodiča. [7]	26
12	Model sféricej projekcie. [8]	27
13	GoPro Omni - séria synchronizovaných kamier pre záchytenie sféricej fotografie. [9]	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)	30
16	Porovnanie výstupu z kamier (ISO 1600) - GoPro Fusion(a), Ricoh Theta V(b) .	35

Zoznam tabuliek

1 Technické parametre sférických kamier 34

Zoznam výpisov zdrojového kódu

1	Množina párov končatín v datasete COCO	19
2	Mapovanie častí ľudského tela rôznych frameworkov	37

1 Úvod

V dnešnom modernom svete sú autá takmer každodennou súčasťou života ľudí. Mnohokrát sa ani nezamýšľame nad ich bezpečnosťou, ktorá je v prípade zrážky klúčová. V súčasnosti nám pri jazde autom asistuje veľké množstvo systémov, ktoré zvyšujú bezpečnosť posádky, ale aj ostantých účastníkov cestnej premávky. Aj keď tieto systémy ešte stále nedokážu vodiča úplne nahradiť, 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 smrtelným dopravným nehodám. Výrobcovia áut sa čoraz častejšie snažia svoje systémy vylepšovať na čo najvyššiu možnú úroveň a poskytnúť tak vysoký level 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 najpríklad aj v interiéri vozidla, kde je vzdialenosť kamery od snímaného objektu menej ako 1 meter, čo nemusí byť dostatočné na zosnímanie tela celého 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 užitočných informácií, ktore sú spracované v tejto diplomovej práci. Tieto informácie boli zbierané nahrávaním videa sférickymi kamerami za rôznych svetelných podmienok a pozicií vodiča. V tejto práci sú taktiež spomenuté problémy takejto analýzy, 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 vhodné datasety z interiéru vozidla snímané sférickou kamerou.

2 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 [10] 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 sú im venované samostatné podkapitoly 2.1 a 2.2.

Každý obraz sa skladá z pixelov. Analýza obrazu však nespočíva v prehľadávaní jednodlivých pixelov, ale v hľadaní jednotlivých objektov v obraze. Tieto objekty je možné určovať do samostatných tried. Triedy nám určujú, aký druh objektu sa v obraze nachádza (Napríklad chodec, vozidlo, dopravná značka a podobne). Aby bolo možné tieto objekty (v našom prípade ľudí) nájsť, bolo potrebné nájsť spoľahlivý a rýchly spôsob detekcie.

2.1 Haar

Táto metóda bola popísaná autormi Viola a Jones [11] v roku 2001. Medzi jej hlavné výhody patrí vysoká rýchlosť a spoľahlivá detekcia a vysoká nezávislosť na intenzite osvetlenia. Vo všeobecnosti je tento detektor rozdelený do 4 samostaných časťí: Výpočet integrálneho obrazu, výpočet Haar príznakov, výber príznakov a kaskádový klasifikátor.

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

1	1	1
1	1	1
1	1	1

(A)

1	2	3
2	4	6
3	6	9

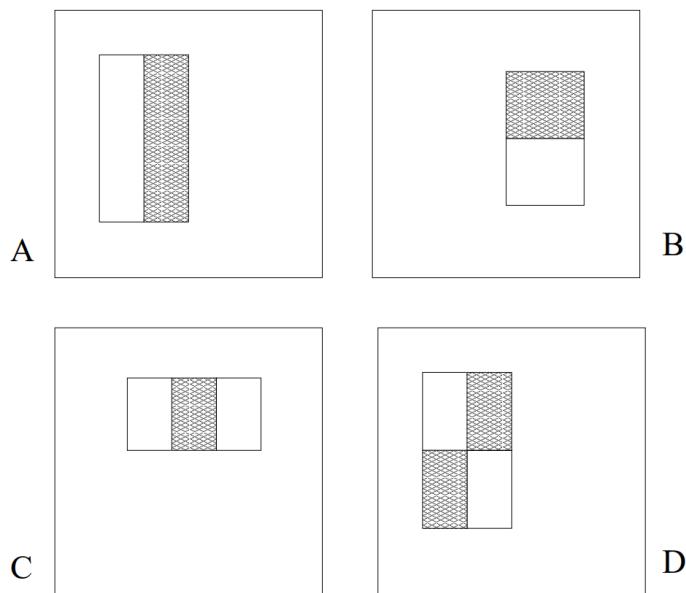
(B)

Obr. 1: 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 porovnanie 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:

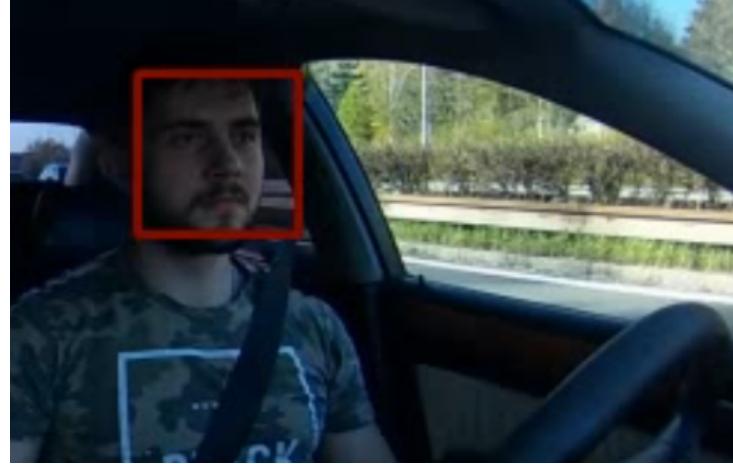
- **Dvoj-obdlížnikové** (*angl. two-rectangle*) - porovnávajú sumu pixlov v obdlížnikových oblastiach, ktoré sa nachádzajú vedľa seba, vodorovne, alebo zvislo.
- **Troj-obdlížnikové** (*angl. three-rectangle*) - porovnávajú sumu obdlížnikových oblastí, ktoré sa nachádzajú po oboch stranách aktuálnej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



Obr. 2: Haar - dvoj-obdlížnikové príznaky (A, B), troj-obdlížnikové príznaky (C) a štvor-obdlížnikové príznaky(D) [1]

Znázornenie jednotlivých typov príznakov môžeme vidieť na obrázku 2. 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 [12], ktorý vytvoril profesor Yoav Freund. Jedná sa o

klasifikačný algoritmus, ktorý je schopný vytvoriť dostatočne silný klasifikátor z kombinácií viacerých slabších klasifikátorov. Metódou Haar je možné detektovať rôzne triedy objektov. Jedným z najčastejších a jednoducho detekovateľných typov objektov je napríklad tvár.



Obr. 3: Haar - detekcia tváre v slabých svetelných podmienkach

2.2 HOG

S nápadom vylepšiť detekciu objektov použitím príznakov prišli v roku 2005 Navneet Dalal a Bill Triggs [2], 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. Hlavnou myšlienkovou ich metódy je, že objekt môže byť charakterizovaný viacerými spôsobmi. Táto metóda je rozdelená do niekoľkých samostatných krokov (Obr. 4):

- **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 $\begin{bmatrix} -1, 0, 1 \end{bmatrix}$ alebo $\begin{bmatrix} -1, 0, 1 \end{bmatrix}^T$. 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 $\theta(x,y)$:

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

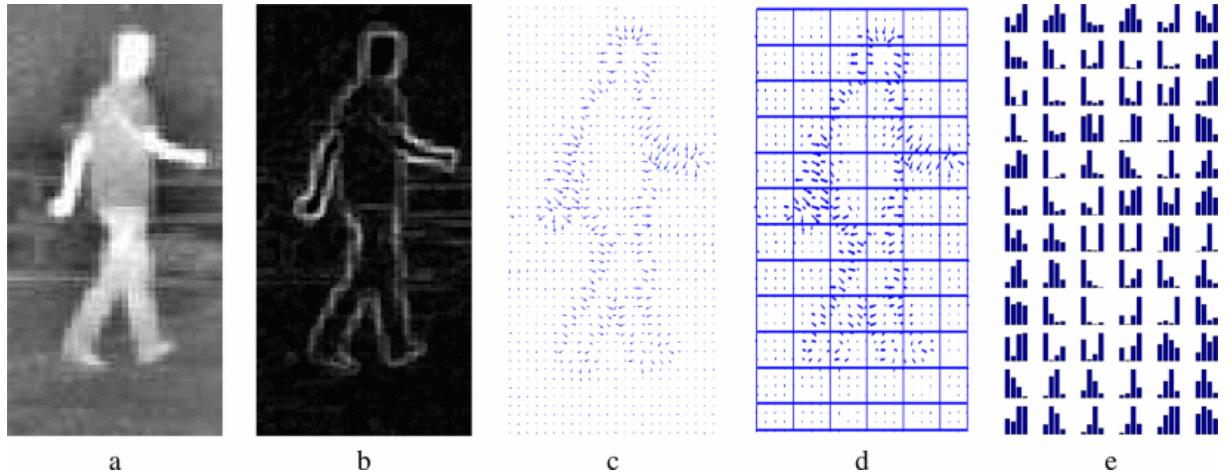
$$\theta(x,y) = \left(\frac{I_y}{I_x} \right) \quad (3)$$

- **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. [13]



Obr. 4: HOG - séria krokov [2]

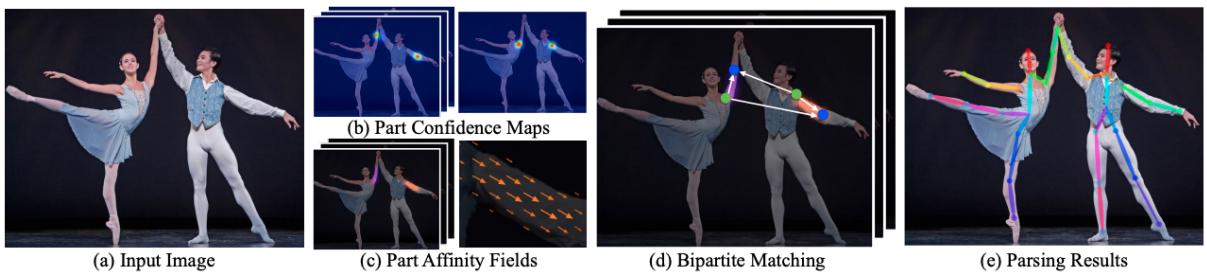
Výstupom tejto metódy je množina histogramov pre jednotlivé bloky. Histogram predstavuje grafické rozloženie intenzity jasu vstupného obrazu. Pri niektorých špecifických obrazoch je potrebné histogram vyrovnať. Tento krok je potrebný najmä pri obrazoch, ktoré sú príliš tmavé alebo príliš svetlé. Pomocou vyrovnania (*angl. equalization*) histogramu dokážeme zvýšiť kontrast obrazu. Jednotlivé kroky znázornené na vstupnom obraze chodca môžeme vidieť na obrázku 5



Obr. 5: HOG - vstupný obraz (a), normalizácia gradientu (b), orientácia gradientu (c), rozdelenie do buniek (d) vypočítané histogramy (e). [3]

2.3 OpenPose

OpenPose [4] je framework, ktorý bol prvýkrat 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 [14].



Obr. 6: OpenPose - odhad viacerých ôsob v reálnom čase pomocou polí afinity filtra. [4]

Samotný framework je veľmi detailne vysvetlený a dobre zdokumentovaný. OpenPose bol pôvodne napísaný v C++ a Caffe [15]. 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. 6a) privádza ako vstup do dvojvetvovej viacstupňovej Konvolučnej nerónovej siete (CNN). Dve vetvy znamenajú, že CNN produkuje dva rôzne výstupy z jedného vstupného obrazu. Viacstupňové znamená, že siet je v každej fáze naskladaná jedna na druh. Tento krok je analogický jednoduchému zväčšeniu hĺbky neurónovej siete s cieľom zachytiť podstatnejšie výstupy smerom k posledným stupňom.
- **Spracovanie v dvoch vetvách** - prvá vetva predpovedá mapy dôveryhodnosti (Obr. 6b) rôznych častí tela, ako je pravé oko, ľavé oko, pravé lakte a podobne. Druhá vetva zobrazená modrou farbou predpovedá afinitné polia (Obr. 6c), čo predstavuje stupeň asociácie medzi rôznymi časťami tela.
- **Viacfázové spracovanie** - v prvej fáze siet vytvorí počiatočnú sadu detekčných máp spoločnosti S a množinu polí afinity 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.

2.3.1 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.

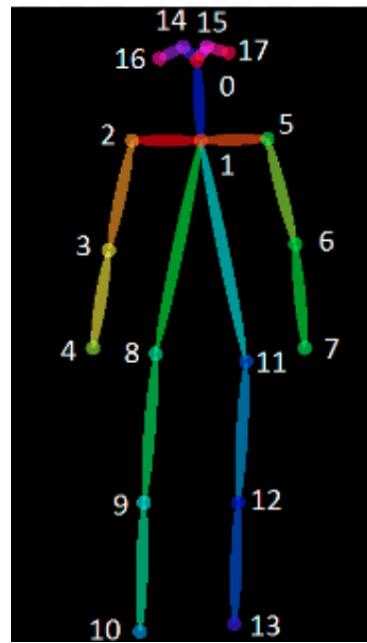
$$S = (S_1, S_2, S_3 \dots S_j) \quad (4)$$

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

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

Počet častí tela závisí od množiny datasetov, s ktorými je program OpenPose trénovaný. Pokiaľ ide napríklad o súbor datasetu COCO [16], $J = 19$, pretože existuje 18 rôznych kľúčových bodov tela + 1 pozadie. Obrázok 7 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 predpokladáme, že prvok S_1 zodpovedá mape spoľahlivosti pre kľúčový bod s číslom 0, ktorý zodpovedá nosu (Obr. 7).

Ako jednoduchý príklad môže poslužiť predstava, že celý obraz má šírku a výšku 5px, 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. 7: COCO - označenie častí tela v COCO datasete. [5]

2.3.2 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 7 je možné vidieť, že párs časť tela pozostáva z pravého ramena k pravému laktu. Schéma potom ukazuje smerový vektor, ktorý ukazuje z pravého ramena na pravý lakef. 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 skúšaný a trénovaný napríklad s datasetmi MPI [17], 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 ľudí, ktorí požadujú štruktúru datasetu MPI. Je tiež pomalší oproti BODY_25 a oveľa menej presný.

2.4 TF Pose Estimation

Tento framework pre detekciu postáv bol 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 prehladiačoch použitím knižnice tensorflow.js. Tensrflow 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 sa zamyslieť sa hneď na začiatku, koľko osôb sa reálne môže v obraze nachádzať. Hlavná myšlienka tohto algoritmu sa skladá z dvoch krokov podobne ako pri knižnici OpenPose.

- **Detekcia pozície** - na najvyšej úrovni modelu PoseNet sa vráti objekt, ktorá obsahuje zoznam klúčových 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 klúčových bodov** - odhadované časti tela osoby, ako napríklad nos, pravé ucho, ľavé koleno, pravá noha atď. Obsahuje pozíciu a spoľahlivosť klúčového bodu. PoseNet štandardne zistuje 17 klúčových bodov.
- **Poloha klúčového bodu** - pozostáva z 2D súradníc v pôvodnom vstupnom obrázku, kde bol zistený klúčový 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.

2.4.1 Výstup

Ked' PoseNet spracováva obraz, v skutočnosti vytvára tepelnú mapu (*angl. Heatmap*) spolu s ofsetovými 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. 8).

2.4.2 Tepelná mapa

Každá pozícia v tejto tepelnej mape 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 termografickej mape poskytuje klasifikáciu pravdepodobnosti, že každý kľúčový bod existuje v každom štvorci mriežky.

2.4.3 Výstupný vektor

Každý výstupný vektor je 3D vektor, ktorý má veľkosť šírka \times výška \times 34. Číslo 34 je dvojnásobok kľúčových bodov (2×17). Tepelné mapy sú iba aproximáciou toho, kde sa skutočné kľúčové body nachádzajú, pričom výstupné vektory zodpovedajú svojou polohou bodom tepelnej 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.

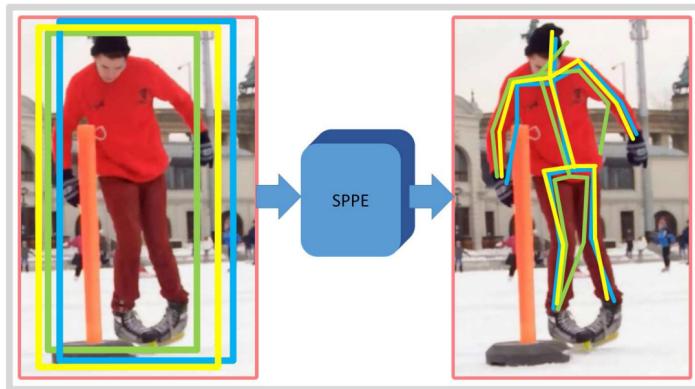


Obr. 8: Tepelná mapa - vstupný obraz (a), tepelná mapa postavy (b)

2.5 AlphaPose

AlphaPose [6] je framework, ktorý vznikol v roku 2017. Je zameraný na detekciu ľudských postáv v najpresnejšej miere. 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 [18] a model pre detekciu postáv v obraze [19], ktorý je zameraný na SPPE detekciu. Primárna myšlienka práce spočíva v riešení dvoch hlavných problémov, ktoré pri takejto detekcii vznikajú:

- **Problém lokalizačnej chyby** - skutočnosti je tento model dosť náchylný na chyby pri spočívajúce v použití ohraničovacích boxov. Aj v prípadoch, keď sú ohraničovacie rámčeky sú považované za správne a majú dostatočne vysokú spoľahlivosť ($I_{\text{OU}} > 0,5$), zistené ľudské pozície môžu byť stále nesprávne.
- **Redundantná detekcia** - model vytvára pózu pre každý ohraničovací box, čo vo výsledku vedie k duplicitnej detekcii rovnakej osoby (Obr. 9)



Obr. 9: Problém redundantnej detekcie s použitím metódy bounding box [6]

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 [20] sú verejne dostupné a určené primárne pre vedu a výskum.

2.6 Ostatné metódy

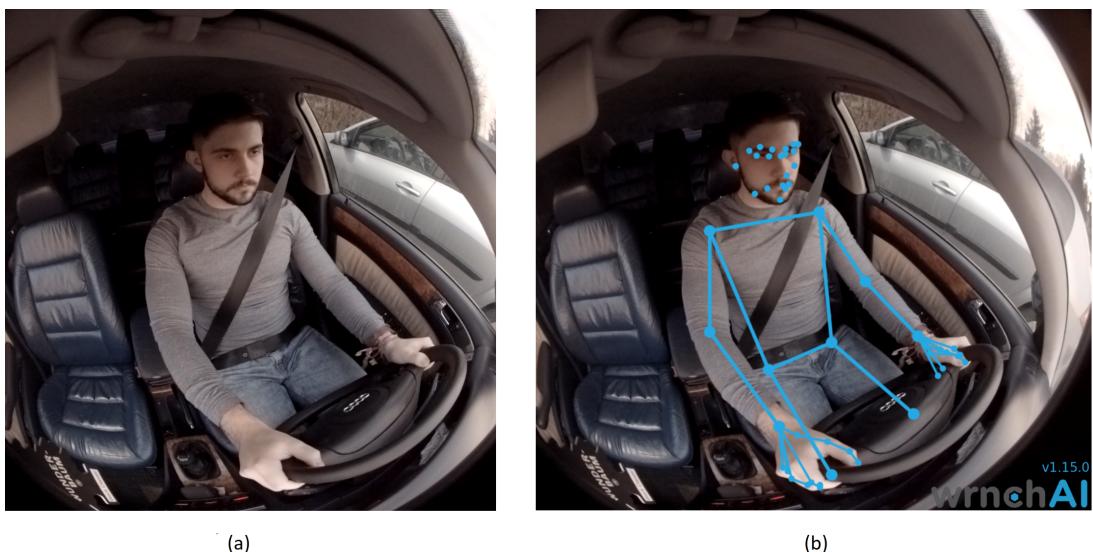
Okrem najpoužívajenších metód ako OpenPose, TF pose estiamtion, či AlphaPose existuje aj mnoho ďalších riešení. Tieto riešenia často vychádzajú zo základného princípu detekcie postáv, ktoré používajú aj tieto hlavné knižnice. Hlavný rozdiel je však použitie upravenej neurónovej siete alebo zmena rôznych parametrov. Vďaka takejto úprave je možné dosiahnúť 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.

2.6.1 WrnchAI

WrnchAI je framework, ktorý bol vytvorený už v roku 2014. S myšlienkovou využitím umelého inteli-genciou a založením projektu pre detekciu postáv prišiel Paul Kruszewski. Napriek tomu, že tento framework existuje už niekoľko rokov a v niektorých článkoch [21] dokázal dokonca získať lepší čas detekcie oproti iným metódam, nestal sa veľmi populárny medzi vývojarmi. 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 človeka sa táto cena môže zdať privysoká, avšak vzhľadom na jeho široké možnosti využitia v rôznych oblastiach sa jedná o adekvátnu cenu. WrnchAI však 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 hlbokého učenia, aby bolo zabezpečené spoloahlivé zachytenie ľudského pohybu pre všetky činnosti a v akomkoľvek prostredí.
- **wrnchCloud** - detektuje 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 základnym 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 vecami ako je napríklad rozšírená realita. S touto pomocou rozšírenej reality dokáže wrnchMobile snímanú postavu detektovať a zároveň 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. 10: WrnchAI - vstupný obraz (a), výsledok spracovania na platforme WrnchCloud(b)

Na obrázku 10 môžeme vidieť výsledok metódy cloudovej 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 klúčových bodov tváre a klúčové 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.

3 Detekcia vodiča vo vozidle

Hlavným zmyslom detektie vodiča je analyzovanie jeho správania za volantom. Pomocou takejto analýzy dokážeme 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 trahové reakcie vodičovho tela, ktoré sú spôsobené pokročilou fázou mikrospánku. Táto fáza je však už často sprevádzaná ďalšími príznakmi, ako zatvorenie očí a krátkodobé nevnímanie situácie, 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 zdetektovať nepozornosť alebo mikrospánok vodiča maximálne behom 1-2 sekúnd. Každá sekunda navyše v rýchlo idúcom vozidle dramaticky zvyšuje riziko vzniku dopravnej nehody.
- **Fungovanie v zhoršených podmienok** - systém by fungoval 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 nelíši. 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 detekcii. Autori v práci na analýze vodiča [22], 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 vo výhľade. Kedže táto práca bola zameraná primárne na detekciu očí a úst, nebolo potrebné používať kamery s vysokým uhlom záberu. Celý proces snímania však nebolo 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 spracovávať 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. Ich detekcia ospalosti vodiča je založená na algoritme, ktorý začína zaznamenávať správanie vodiča pri začatí jazdy. Výrobca si však uvedomoval všetky nedostatky použitia kamery na smínaie 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. Typickými znakmi klesajúcej koncentrácie sú fázy, počas ktorých vodič sotva riadi, 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.

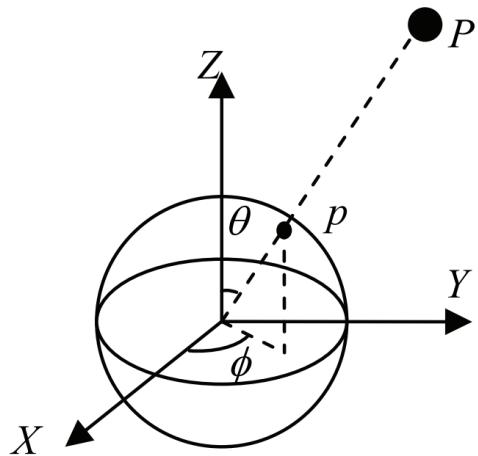
Mnoho podobných riešení zameraných na snímanie aktivity vodiča sa snažia zaviesť aj výrobcovia automobilov. Títo výrobcovia si veľmi dobre uvedomujú, že veľká časť dopravných nehôd býva spôsobená mikrospánkom, alebo len nepozornosťou vodiča. Príkladom môže byť výrobca automobilov značky BMW, ktorý vo 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 upozornený zvukovým znamením. Kamera je umiestnená v prístrojovej doske na mieste za volantom, odkiaľ je na tvár vodiča priamy výhľad (obr. 11). Okrem jej vhodného umiestnenia je navyše vybavená aj IR LED diódami, vďaka ktorým kamera funguje aj za znížených svetelných podimenok a v noci, kedy sa vyskytuje najväčší počet mikrospánkov 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.



Obr. 11: BMW driving assistant - umiestnenie kamery na snímanie vodiča. [7]

4 Využitie sférických kamier na detekciu obrazu

Výstup z kamery je reprezentovaný 2D snímkou, ktorá zachytáva obraz z celého svojho okolia. Toto zobrazenie sa dá popísť modelom sférického obrazu (Obr. 12). Predpokladajme, že existuje priestorová guľa a bod P , ktorý sa nachádza v priestore. Priesečník 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 môžeme popísť 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. [8]

Zhotovenie takejto sférickej snímky je technicky náročnejšie ako bežná fotografia. Pre dosiahnutie správneho výsledku je potrebné zachovať určité podmienky. Ideálny sférický model obrazového snímača by mal spĺňať 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 snímania z viacerých miest, rekonštrukcia výslednej sférickej sníky by sa nemusela správne vytvoriť.
- **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 rozmazený.

Možnosti snímania sférického obrazu

Aby sme dostali 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 výtvárania sférickej fotografie. Medzi najbežnejšie spôsoby zhotovenia sférických obrazov je možné zaradiť tieto spôsoby:

- **Použitie bežného fotoaparátu** - vytvorenie sférickej fotografie je možné aj pomocou bežného fotoaparátu, ktorý má obmedzený uhol záberu. Aby bolo možné zachytiť celú scénu, je potrebné vytvoriť fotografie celej scény, 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 fotografii sa môžu pohybujúce objekty objaviť viackrát. Ďalšou nevýhodou je zdĺhavé vytváranie a spájanie jednotlivých snímok, ktore nemusia vo výsledku na seba úplne nadväzovať.
- **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 ťažké, čo môže byť pre určité zamerania obmedzujúce. Zariadenie spočívajúce zo skupiny 6 kamier vytvorila napríklad firma GoPro. Skladá sa zo 6 outdoorových kamier, ktoré sú pripojené ku centrálnej jednotke. Ukážky takýchto zariadení sú znázornené na obrázku 13
- **Použitie všesmerovej kamery** - ďalšiou populárnu metódou je snímať široké scény jediným fotoaparátom s použitím zakriveného zrkadla. Tvar zrkadla môže byť napríklad parabolický, hyperbolicý 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 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 umiestnených objektívov, ktoré sa následne spracujú do výslednej sférickej fotografie. Hlavnou výhodou takého riešenia je cena, dostatočne kvalitný obraz bez väčších rozdielov medzi jednotlivými snímkami objektívov a rýchle vytvorenie snímky.

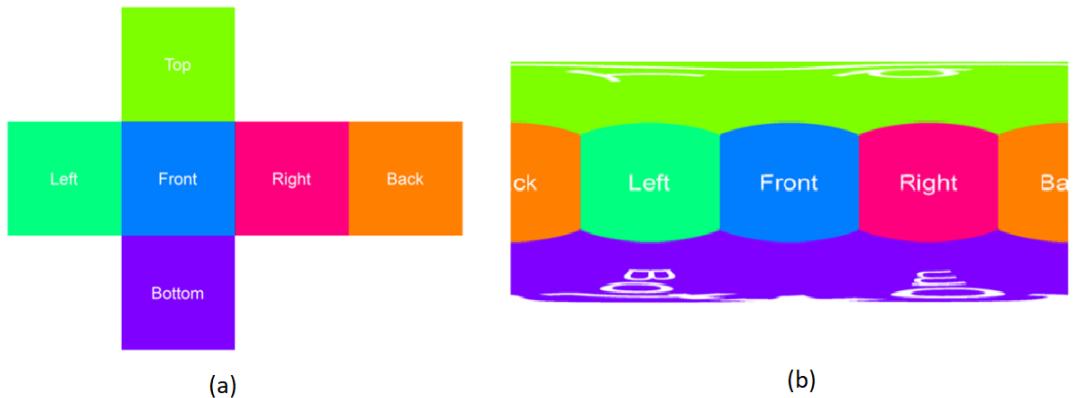


Obr. 13: GoPro Omni - séria synchronizovaných kamier pre zackytenie sférickej fotografie. [9]

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 máp chýba. Cylindrickú prjekciu je teda možné zobraziť aj ako obyčajnú fotografiu, ktorá však bude veľmi široká z dôvodu zackytenia 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 videohrách. 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 roztahnuté 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 radiálne artefakty, 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 zanedbateľné. Riešením je prepísanie na menej zdeformovanú projekciu pred zmenšením mierky, rozmazením alebo zaostrením panorámy a v prípade potreby prepísanie do sférického režimu neskôr. 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.



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

5 Vlastná implementácia

5.1 Popis programu

Táto kapitola je zameraná na popis implementácie vlastného programu. Hlavnou podstatouvýsledného programu je detekcia vodiča z videa alebo z jednotlivých obrázkov nahratých kamerou z interiéru vozidla. V kapitole sú tiež detailné popísané možnosti, problémy a riešenia nahrávania samotných videí z interiéru vozidla pri rôznych problematických 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 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 používanie metódy na detekciu postáv v súčasnosti. Okrem toho majú výborne zdokumentované zdrojové kódy a dosťatočné 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 zapnutie bezpečnostného pásu, alebo natočenie hlavy. Z týchto informácií je potom možná ďalšia analýza vodičovho správania a vyhodnotenie rizík. Program je rozdelený do jednotlivých samostatných celkov, ktoré sú na sebe nezávislé. Každý z týchto celkov spracováva konkrétnu časť:

- **Detekcia vodiča** (Kapitola 5.5) - základným prvkom programu je analýza vodičovho posedu. V kapitole je popísané využitie jednotlivých frameworkov na detekciu a analýzu ľudského tela zo zozbieraných údajov pomocou jednotlivých frameworkov.
- **Otočenie hlavy** (Kapitola 5.6) - je zameraná na rotáciu a natočenie hlavy vodiča pomocou detekcie tvárových landmarkov a knižnice dlib [23].
- **Bezpečnostný pás** (Kapitola 5.7) - táto kapitola je zameraná predovšetkým na problematiku snímania bezpečnostného pásu pri rôznych nepriaznivých podmienkach.
- **Neurónová sieť** (Kapitola 5.8) - vo výslednej časti je popísaná implementácia neurónovej siete, ktorá vyhodnocuje jednotlivé výstupy predchádzajúcich častí.

5.2 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 funkcionality 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 funkcionality, 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 obrazove. 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 obrazove.
- **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 celeho behu programu. Dôležité údaje sú čas spracovania jednej snímky, celková úspešnosť a celková doba programu.
- **Univerzálnosť** - aby bolo jednoduché použiť program na viacerých operačných systémov, je potrebné ho napísať tak, aby nevyužíval žiadnu funkcionality, ktorá je závislá na konkrétnom operačnom systéme.

5.3 Použité technológie

Programovací jazyk

Hlavný program je napísaný v jazyku Python. Tento jazyk bol použitý z dôvodu výnikajúcej 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 výhodam 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 malého 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 neuronovými sietami 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 spracovanie 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 [24] 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 neuronovými sietami.

Keras

Keras [25] 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 [26] a CNTK. V aplikácii je Keras využitý na vytvorenie modelu a jednotlivých vrstiev neurónovej siete.

5.4 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 prímärne 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. Obidve kamery obsahujú 2 objektivy tzv. fish eye, 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 zhorávajú 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

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 zhorávajú úlohu aj ostatné parametre ako napríklad svetelnosť použitých senzorov. Aj napriek tomu, že svetelnosť senzora je totožná pri obidvoch kamerách, rozdiel v kvalite výstupného videa je ľahko rozpoznateľný najmä pri znižených 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 znižených 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 autoamtické 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)

5.5 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. Preto bolo potrebné naštudovanie dokumentácie a príprava implementácie. 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 inštálácii všetkých potrebných knižníc a balíčkov je potrebné ich správne nakonfigurovať. Openpose ponúka na výber mnoho parametrov. Jedným z dôležitých parametrov pri detekcii postáv v obraze je použitý dataset.

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 nepostačujúceho výpočtového výkonu nebol v tejto práci používaný.// TODO

TF Pose

- TODO

Zjednotenie výstupov

- TODO - fasada
- graf

```

if(poseType == "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]))
    knee_left = (int(human[12][0]), int(human[12][1]))
    knee_right = (int(human[9][0]), int(human[9][1]))

elif (poseType == "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))
    knee_left = (int(human.body_parts[12].x * image_w + 0.5), int(human.body_parts[12].y * image_h + 0.5))
    knee_right = (int(human.body_parts[9].x * image_w + 0.5), int(human.body_parts[9].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 rôznych frameworkov

5.6 Orientácia hlavy

- Haar priznaky ,
- prevod 2D na 3D
- smerova priamka

5.7 Bezpečnostný pás

- bezp. pas

5.8 Neurónová sieť

- NN klasifikator
- trenovanie
- testovanie

5.9 Výstup programu

obrazky
tabulky

5.10 Porovnanie výsledkov

porovnanie

5.11 Využitie zozbieraných dát

pouzitie v buducnosti

5.12 Používateľská príručka

`python program.py –use-openPose=true`

6 Možnosti vylepšenia detekcie

Zhrnutie vysledkov

7 Záver

Zhrnutie vysledkov

Literatúra

- [1] Paul Viola, Michael Jones, et al. Robust real-time object detection. *International journal of computer vision*, 4(34-47):4, 2001.
- [2] 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.
- [3] 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.
- [4] 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.
- [5] 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/>.]
- [6] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. Rmpe - regional multi-person pose estimation. In *ICCV*, 2017.
- [7] 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>.
- [8] Shigang Li. Full-view spherical image camera. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 4, pages 386–390. IEEE, 2006.
- [9] GoPro, 2016. [Online; Citované 6.03.2020 z https://gopro.com/en/cz/news/vr_spherical_omni_rig_shipping.
- [10] Azriel Rosenfeld. Picture processing by computer. *ACM Computing Surveys*, 1(3):147–176, Jan 1969.
- [11] 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.
- [12] 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.

- [13] Yanwei Pang, Yuan Yuan, Xuelong Li, and Jing Pan. Efficient hog human detection. *Signal Processing*, 91(4):773–781, 2011.
- [14] 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>.
- [15] 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.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] 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.
- [20] 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>.
- [21] 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/>.
- [22] 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.
- [23] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.
- [24] 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.

- [25] François Chollet et al. Keras. <https://github.com/fchollet/keras>, 2015.
- [26] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.