Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Detekcia jazdných pruhov

Bakalárska práca

2020
Ján Zdarilek

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Detekcia jazdných pruhov

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	Aplikovaná informatika
Školiace pracovisko:	Katedra Aplikovanej informatiky
Školiteľ:	RNDr. Zuzana Černeková, PhD.

Bratislava, 2020
Ján Zdarilek

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Ján Zdarilek
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Detekcia jazdných pruhov
Road lane marks detection

Anotácia: Naštudovať problematiku z oblasti asistenčných systémov pre riadenie automobilu. Analyzovať predchádzajúce práce v oblasti detekcii jazdných pruhov. Urobiť prehľad existujúcich aplikácii. Implementovať jednu zvolenú metódu.

Cieľ: Naštudovať problematiku z oblasti asistenčných systémov pre riadenie automobilu. Analyzovať predchádzajúce práce v oblasti detekcii jazdných pruhov. Urobiť prehľad existujúcich aplikácii. Implementovať jednu zvolenú metódu.

Vedúci: RNDr. Zuzana Černeková, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 04.10.2019

Dátum schválenia: 07.10.2019 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Ďakujem RNDr. Zuzane Černekovej, PhD. za odborné vedenie a pomoc pri tvorbe bakalárskej práce.

Abstrakt

Cieľom tejto bakalárskej práce je venovať sa problematike detekcie jazdných pruhov, analýze existujúcich riešení a implementácii jednej z nich. V prvej kapitole je opísaný úvod k práci, rôzne štatistické údaje, zákony a opísanie problematiky. V druhej kapitole sú popísané existujúce riešenia a ich analýza. Tretia kapitola sa venuje technológiám a algoritmom, s ktorými sme sa stretli pri tvorbe programu. Ďalšie kapitoly sú venované výslednému programu. Najskôr návrhu, potom samotnej implementácii a posledná kapitola sa venuje výsledkom testovania aplikácie a ich analýze.

Kľúčové slová: Spracovanie obrazu, jazdné pruhy, video, pohľad z vtáčej perspektívy

Abstract

The aim of this bachelor thesis is to devote the issue of lane detection, analysis of existing solutions and implementation of one of them. The first chapter describes the introdution to the work, various statistics, laws and description of the issue. The second chapter describes the existing solutions and their analysis. The third chapter devoted with technologies and algorithms that we encountered in creating the program. The next chapters are devoted to the final program. First the design, then the implementation itself, and the last chapter deals with the results of application testing and analysis.

Keywords: image processing, road lane, video, bird's eye view

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Šoférovanie je pre väčšinu ľudí bežná vec, nad ktorou sa nemusia špeciálne zamýšľať a mnohé veci robia automaticky. Úlohou šoféra je prejsť bezpečne do svojho cieľa bez kolízií. Na ceste je dôležité sledovať prvky, ktoré pomáhajú udržovať dopravu bez nehôd a pri ich dodržiavaní sa znižuje práve riziko zrážok. Sú to značky, semafory, ale aj jazdné pruhy. Napriek týmto pomôckam pre šoféra sa stávajú mnohé kolízie a niektoré z nich aj s následkami nezlučiteľných so životom. Sú spôsobené nepozornosťou vodiča, rýchlou jazdou, mikrospánkom, resp. zvýšeným množstvom alkoholu v krvi. Práve mikrospánok má za následok veľké množstvo dopravných nehôd, keď vodič až s oneskorením reaguje na situáciu na vozovke pred ním.

Mikrospánok je krátkodobý spánkový stav, ktorý trvá niekoľko sekúnd. Slúži na regeneráciu psychických a fyzických síl. Človek vie tomuto stavu predísť tým, že ide na cestu odpočinutý, počas dlhšej cesty si dáva krátke prestávky, dodržiava pitný režim. Avšak, ak nedodrží týchto pár základných rád, mikrospánok na neho môže prísť hocikedy. Sú určité príznaky, ktoré ukazujú možné prejdenie organizmu do stavu mikrospánku. Sú to zívanie, zahmlievanie pred očami, hladenie do diaľky. V prípade, že vodič ani na tieto príznaky nereaguje, vystavuje seba, ale aj okolo idúcich riziku dopravnej nehody. Je dôležité hľadať riešenia na tento problém a vďaka nemu zachrániť veľa ľudských životov. Je to výzva pre každého z nás.

Vývoj bezpečnostných systémov je kľúčový, čo sa týka budúcnosti. Najdôležitejšia je samozrejme záchrana ľudských životov, ale majú budúcnosť aj pri zľahčovaní života ľudom, napríklad pomocou autonómnych áut. Tieto body sú kľúčové a od nich sa má odvýjať náš postoj pri tvorbe takýchto systémov. Každý pokus o zlepšenie aj už existujúcich systémov môže priniesť obrovskú zmenu. Každá dostupná aplikácia na detekciu potencionálneho nebezpečenstva dáva možnosť ľuďom chrániť seba, ale aj druhých.

Práve v tejto práci sa budeme venovať tomu, ako pomôcť predchádzať za pomoci počítačového videnia takýmto nehodám. Cieľom tejto práce je detekovať jazdný pruh a pozrieť sa na existujúce riešenia. Ich analýzou zistiť

možnosti pre ďalší vývoj a nové pokusy o zlepšenie presnosti. Naším cieľom je tiež vytvoriť aplikáciu, ktorá na videu dokáže správne rozpoznať jazdný pruh.

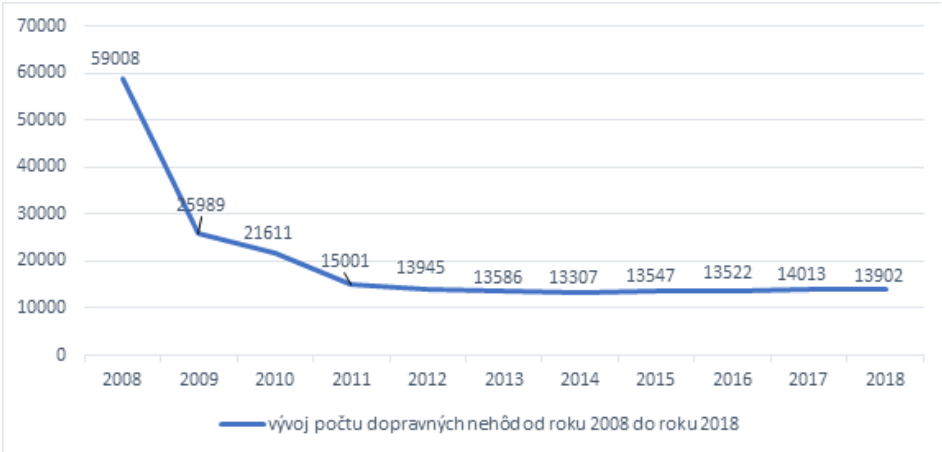
Kapitola 1

Úvod do problematiky

Táto časť je venovaná úvodu do problematiky a získaniu všeobecných informácií k práci a samotnej detekcii čiar na ceste. Taktiež k oboznámeniu sa s pojmami a zákonmi, ktoré súvisia s premávkou.

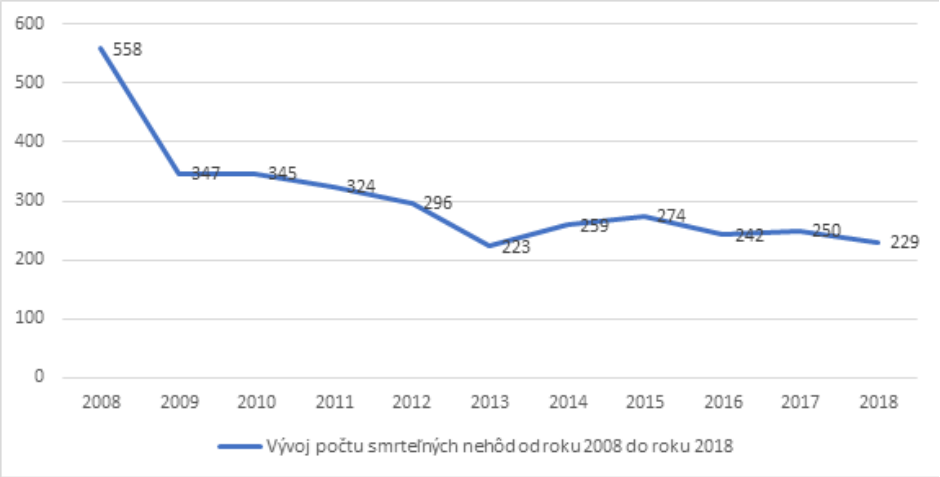
1.1 Štatistické údaje

V tejto časti sa pozrieme na štatistiky dopravných nehôd na Slovensku[21]. Tieto údaje sú zbierané a tvorené políciou Slovenskej republiky pre Ministerstvo vnútra Slovenskej republiky. Vďaka nim si môžeme spraviť predstavu o tom, ako je to s nehodami na slovenských cestách. Podľa štatistík z každého roka vieme povedať, že sa nehodovosť od roku 2008 znížila o viac ako štvrtinu. Od roku 2012 sa počet nehôd ustálil medzi 13 až 14 tisíc ročne. V roku 2018 bolo na Slovensku 13902 dopravných nehôd, čo v prepočte vychádza, že sa v Slovenskej republike priemerne stane 38 havárií denne. Klesajúci trend počtu havárií je pozitívnou správou. Tento pokles je spôsobený prísnejšími dopravnými zákonmi, ale aj vďaka vývojom bezpečnostných prvkov pre automobily.



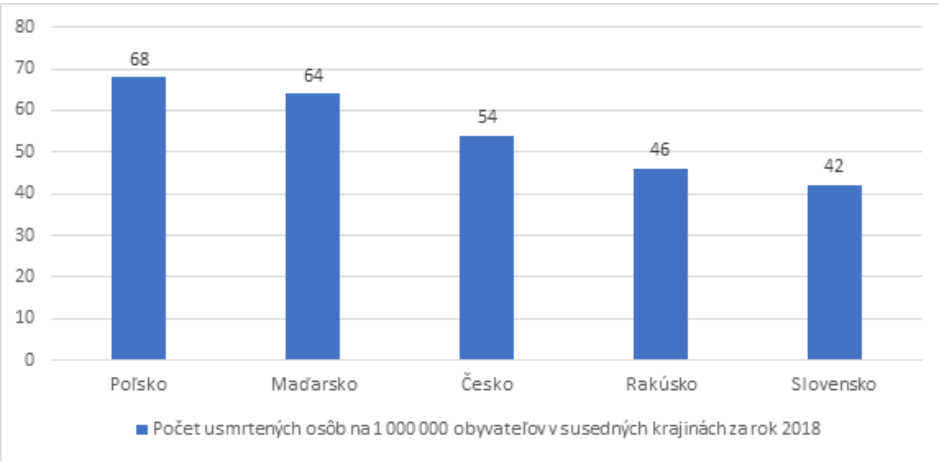
Obr. 1.1: Vývoj počtu dopravných nehôd od roku 2008-2018

Počet smrteľných nehôd sa tiež od roku 2008 znížil, a to približne o polovicu. Keď porovnáme štatistiky vývoja počtu dopravných nehôd a počtu smrteľných nehôd, vyplýva, že aj znížením počtu nehôd je stále úmrtnosť na cestách veľmi vysoká. Od roku 2008 sa znížil počet nehôd násobne, ale väčšina z nich sú práve nehody, ktoré nemajú smrteľné následky. V roku 2018 bolo na slovenských cestách usmrtených 229 ľudí. V prepočte na obyvateľov to vychádza 42 usmrtených na jeden milión ľudí.



Obr. 1.2: Vývoj počtu smrteľných nehôd od roku 2008 do roku 2018

V porovnaní s okolitými krajinami máme podľa štatistík najnižšiu úmrtnosť na cestách v prepočte na milión obyvateľov. Avšak táto štatistika je skreslená, keďže v Slovenskej republike sa za smrť pri nehode ráta, len ak človek zomrie priamo pri nehode alebo na následky do 24 hodín, kým v medzinárodných štatistikách sa ráta úmrtie na následky nehody do jedného mesiaca. [21]. Každá krajina má vlastné štatistické metódy, takže tieto štatistiky môžu byť len orietačné a nemusia byť úplne presné.



Obr. 1.3: Počet usmrtených osôb na 1000000 obyvateľov v susedných krajinách za rok 2018

1.2 Zákony o cestnej premávke

V tejto časti si ujasníme niektoré pojmy zo zákonov Slovenskej republiky ohľadom cestnej premávky. Vychádzam pritom zo zákona č. 8/2009 Z. z. [22]. Prvou dôležitou časťou je vysvetlenie samotného pojmu cestná premávka. „Cestnou premávkou na účely tohto zákona sa rozumie užívanie diaľnic, ciest, miestnych komunikácií a účelových komunikácií1) (ďalej len „cesta“) vodičmi vozidiel a chodcami.“ Pri sledovaní vozovky je dôležitá aj okrajová časť, krajnica. „Krajnicou sa myslí časť cesty od kraja vozovky po kraj cesty.“ A posledným pojmom, ktorému sa budeme venovať v tejto časti je vozovka. „Vozovkou sa myslí spevnená časť cesty určená predovšetkým na premávku motorových vozidiel.“

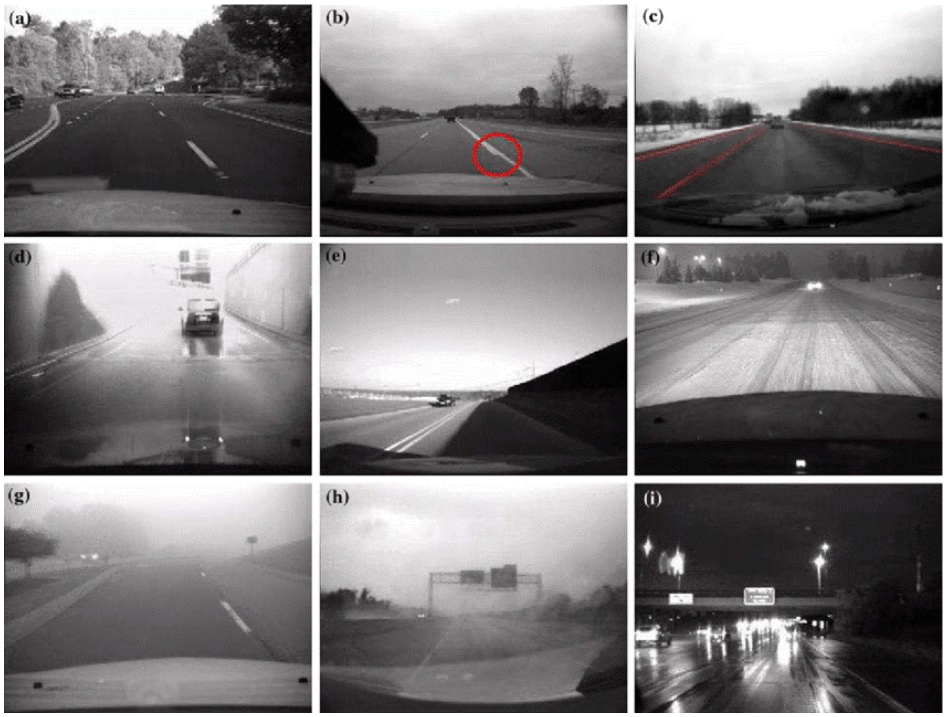
1.3 Problematika

Kvôli zníženiu počtu havárií a úmrtí na ceste sa začali vyvíjať rôzne bezpečnostné systémy pre autá. Tieto systémy nielen, že dávajú väčší komfort a bezpečnosť pre vodiča, ale pomáhajú chrániť aj ostatných účastníkov premávky. Nehovoríme tu len o druhých vodičoch áut, ale aj o chodcoch a cyklistoch. Tieto systémy pomáhajú šoférovi v nepredvídateľných situáciách, ale aj keď zlyhá ľudský faktor. Často signalizujú rôzne prekážky na ceste, povolenú rýchlosť na danom úseku cesty, či daný vodič jazdí podľa čiar na ceste a mnoho ďalších. Pri niektorých prípadoch dokonca autopilot preberie riadenie a bezpečne odstaví vozidlo na krajnicu. Tieto systémy by s najväčšou pravdepodobnosťou neexistovali nebyť práve pokroku v oblasti počítačového videnia. Jazdné pruhy sú pre šoféra jeden zo základných orientačných prvkov na ceste. Sú vyznačené rôznymi typmi čiar a farieb. Typy čiar informujú šoféra, či môže napríklad predbiehať druhé vozidlo, či na danom mieste môže odstaviť vozidlo. Na Slovensku sa ako farba pruhov používa biela, a v prípade, že je cesta v štádiu opravy a pruhy sú zmenené, dočasné pruhy majú oranžovú farbu. Avšak, nie každá krajina má biele jazdné pruhy. Napríklad, ak sa pozrieme na Spojené štáty americké, zistíme, že používajú namiesto bielej žltú farbu.

1.3.1 Problémy pri detekcii čiar na vozovke

Vnímanie vozovky zahŕňa zisťovanie počtu a polohy pruhov, zlučovanie pruhov, ale aj to, či je cesta v dedine, meste alebo mimo mesta. Pri samotnej detekcii sa stretávame s množstvom problémov, ktoré je potrebné riešiť. Jedným z nich je rôzny vzhľad čiar. Líšia sa vo svojej šírke, dĺžke, či typom. Čiary sa môžu rozširovať, zužovať, môžu byť prerušované, spojité. Druhým je jasnosť obrazu. Problém nastáva napríklad pri výjazde auta z tunela. Vtedy môže byť obraz príliš presvetlený. A tretím priblémom je,

ak je slabá viditeľnosť. Môže byť spôsobená hmlou, silným dažďom, snehom na ceste, tieňmi na ceste, alebo odrazom od mokrej vozovky [1]. Keďže potrebujeme, aby bezpečnostný systém fungoval v reálnom čase, a zároveň jeho omylnosť má byť čo najnižšia, je potrebné dané prípady problémov vyriešiť tak, aby čo najmenej spôsobovali chyby systému.



Obr. 1.4: Problémy pri detekcii čiar na ceste: a) rôzne typy čiar b) zmena šírky čiar c) rôzna šírka pruhov d) presvetlená časť snímky po východe z tunela e) tieň na ceste f) cesta pokrytá snehom g) nízka viditeľnosť kvôli hmle h) nízka viditeľnosť kvôli hustému dažďu i) odraz na mokrej vozovke v noci [1]

1.3.2 Systémy používané na snímanie cesty

V tejto časti sa pozrieme na systémy, ktoré je možné využiť pri sledovaní jazdného pruhu [1]. Niektoré z nich sú veľmi dobré a presné, avšak ich náklady na zaobstaranie sú častokrát veľmi vysoké, a niektoré nie sú drahé, ale zase presnosť je tam obmedzená. Väčšinou sa používa kombinácia týchto systémov, keď sa skĺbi presnosť s čo najnižšou výrobnou cenou.

Videnie pomocou jednej kamery: alebo inak monokulárne videnie, funguje za pomoci jednej kamery namontovanej v strede čelného skla auta. Veľmi často býva za spätným zrkadlom. Kopíruje ľudský pohľad na vozovku.

LIDAR: Jeho nevýhoda je jeho vysoká cena, avšak má veľmi veľa dobrých vlastností použiteľných pri detekcii. Dokáže zmerať 3D štruktúru vozidla a určiť vzdialenosť za pomoci odrazu lúča podľa času odrazu od objektu. Nemá problém so svetelnými

podmienkami. Napríklad mu nevadia tiene, pretože využíva aktívny zdroj svetla. Lidar nám dáva 3D informácie o stúpaní vozovky, rozpoznávaniu obrubníka a okrajov ciest a odhad drsnosti vozovky, ktoré slúži na zistenie okraju vozovky v prípade, že sa tam nenachádza značenie.

Stereo zobrazenie: Pozostáva z dvoch kamier. Je to krok medzi monokulárnym videním a Lidarom. Stereo je lacnejšia alternatíva k Lidaru. Vie nám poskytnúť rovnaké údaje ako lidar, ale s menšou presnosťou.

Geografický informačný systém (skr. GIS), Globálny lokalizačný systém (skr. GPS) a Inerciálna meracia jednotka (skr. IMU): GIS je počítačový systém pracujúci s geografickou informáciou. GPS slúži na zistenie presnej pozície pomocou satelitov. Geografické dáta spolu s polohou vozidla sa využívajú v komerčných navigačných systémoch. Tieto GPS prijímače zvyčajne dosahujú presnosť na 5 až 10 metrov, ktoré možno zlepšiť použitím presného IMU. Kombináciou týchto troch prvkou by sme dokázali spraviť fungujúce autonómne auto. Avšak, závisí od presnosti navrhnutého systému a spoľahlivosť dát. Napríklad nepresné údaje na digitálnej mape, výpadok GPS signálu.

Dynamika vozidla: Meria rýchlosť, zrýchlenie a uhlovú rýchlosť vozidla pomocou rýchlosti kolesa. Tieto informácie bývajú využívané na dosiahnutie lepších výsledkov počas sledovania jazdných pruhov. Avšak, presnosť meraní dynamiky vozidla je obmedzená.

Radar: Pre účely detekcie čiar na ceste nie je veľmi užitočný, avšak je využívaný v iných systémoch. Dokáže, napríklad, detekovať prekážky.

Kapitola 2

Existujúce riešenia

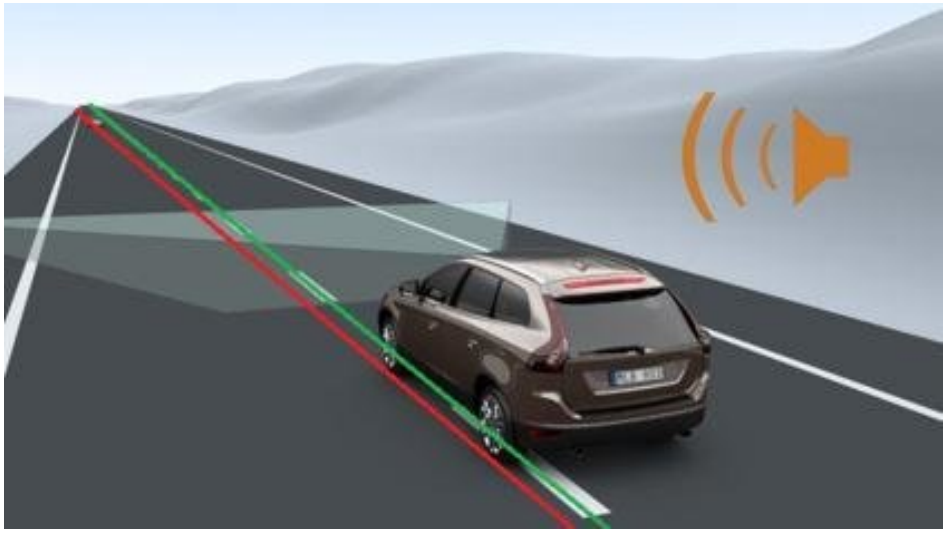
Táto kapitola sa bude venovať už existujúcim asistenčným systémom, ktoré sú založené na algoritmoch počítačového videnia. Sú to rôzne mobilné aplikácie, ale aj samotné systémy pre autonómne autá.

2.1 Systémy pre autá

Bezpečnostné systémy pre autá sú v dnešnej dobe samozrejmosť. Mnohí ľudia sa ani nezamýšlajú nad tým, ako tieto systémy fungujú. Ale v mnohých prípadoch zachránia život a vodič alebo iný účastník premávky si ani neuvedomí, že práve takýto systém pomohol pri prevencii pred kolíziou.

2.1.1 LKA – Lane Keep Assistant

V novších vozidlách zväčša nájdeme aj systém na udržiavanie v jazdnom pruhu (LKA – Lane Keep Assistant). Novšie systémy aktívne zasahujú do riadenia a udržujú auto v strede jazdného pruhu. Systém využíva na svoju činnosť kameru umiestnenú za predným sklom a deteguje vodiace čiary v obraze. V prípade, že sú čiary dobre viditeľné, začne vozidlo navádzať. Úroveň zásahu systému je v závislosti od výrobcu či modelu rôzna. Niektoré vytvárajú iba moment na volante, ale samostatne vozidlo neriadia, niektoré zasahujú do riadenia dosť razantne. Žiadny zo systémov, však nie je v takom štádiu, že by sme sa naň mohli vždy spoľahnúť. Sú to iba asistenčné systémy, ktoré uľahčujú jazdu najmä na diaľniciach, ale aj vzhľadom na legislatívu musíte vždy aspoň pridŕžať volant a byť pripravený kedykoľvek korigovať chyby systému. Vzhľadom na to, že štandardne sa tieto systémy orientujú iba podľa kamery, majú svoje obmedzenia. Tými sú predovšetkým znížená viditeľnosť, hmla, silný dážď, sneženie a podobne.[14]



Obr. 2.1: Systém LDWS[8]

2.1.2 LDWS - lane departure warning system

LDWS je systém na signalizáciu vybočenia z jazdného pruhu.Tieto systémy sú navrhnuté tak, aby minimalizovali nehody spôsobené, napríklad rozptýlením vodiča alebo ospalosťou. Tento systém využíva princíp Houghovej transformácie a detekcie okrajov pomocou Canny edge detektora. [8]

2.1.3 Značky Tesla a Volvo

Automobilky Tesla, Volvo sa podieľajú na tom, že vývoj bezpečnostných systémov áut ide veľmi rýchlym tempom dopredu. Samozrejme, aj ostatné automobilky majú vplyv, avšak, tieto dve sú na momentálnom trhu jednotky vo vývoje technológií. Firma Volvo v roku 1959 vymyslela revolučný vynález. Dovolíme si tvrdiť, že je to jeden z najzásadnejších bezpečnostných systémov. Je to bezpečnostný pás. Keďže vedeli, že daný vynález môže zachrániť milióny životov, ponúkli ho verejne pre všetky automobilky bezplatne. [7] Tento bod v histórii bol zásadným. Práve tento vynález ukázal výrobcom áut, že nie je dôležitá iba rýchlosť auta, výdrž, spotreba, ale že práve bezpečnosť má byť pre všetkých prvoradá.

V roku 2016 uviedla firma Tesla na trh nový asistenčný systém. Jeho úlohou je, aby auto držalo v plánovanom jazdnom pruhu. Systém využíva kameru, ktorá je zabudovaná v aute, ultrazvukové a radarové senzory. Práve ony udržujú auto v pruhu, sledujú ostatné vozidlá v okolí. V prípade, že by vodič chcel zmeniť jazdný pruh, ale daná zmena pohybu auta by mohla spôsobiť kolíziu alebo ohroziť niekoho, systém zasiahne. Ďalej využíva senzory na volante. V prípade, že sa vodič nedrží dlhšiu dobu volantu, auto začne spomaľovať a spustí signalizačné svetlá pre okoloidúce autá. [9]

2.2 Mobilné aplikácie

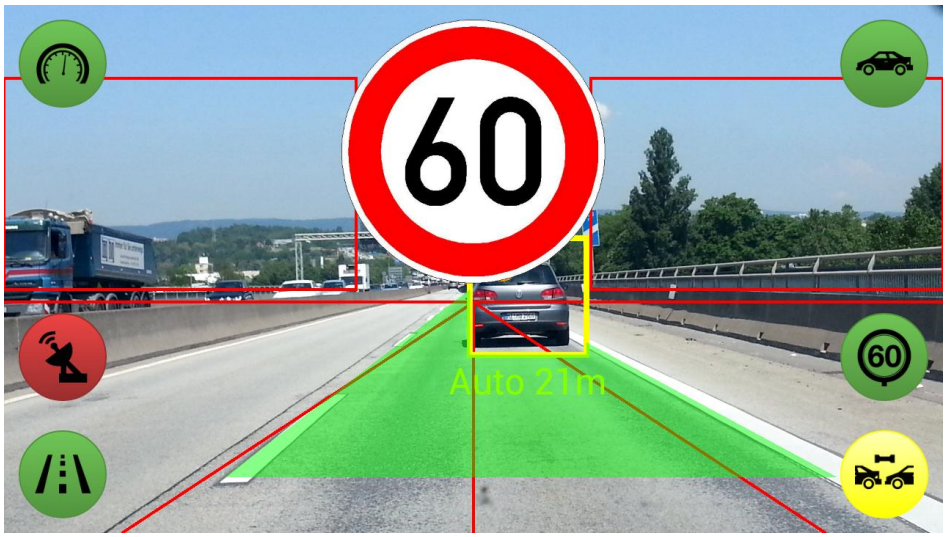
Pri našej práci budeme vychádzať hlavne z mobilných aplikácií, pretože majú rovnaké možnosti na snímanie cesty, aké plánujeme využiť. Tiež využívajú len jednu kameru, za pomoci ktorej sa snažia detekovať jazdné pruhy, okoloidúce autá a rôzne ďalšie. Nie sú zďaleka také presné ako aplikácie zabudované v autách, ktoré sú tvorené viacerými spôsobmi snímania. Ale napriek tomu sú schopné pomôcť vodičovi a v niektorých prípadoch aj zachrániť život.

2.2.1 iOnRoad

Táto mobilná aplikácia ukazuje vodičovi jazdnú dráhu pruhu a vzdialenosť áut nachádzajúcich sa pred vozidlom. Dáva hlasový aj vizuálny signál používateľovi, že auto vybáča z pruhu, resp. auto idúce pred jeho vozidlom je príliš blízko. Aplikácia využíva natívny fotoaparát zo smartfónu, GPS a senzory na detekciu vozidiel pred používateľovým vozidlom. Vypočítava aj aktuálnu rýchlosť vodiča pomocou natívnych snímačov, vďaka čomu vie určiť bezpečnú vzdialenosť medzi vozidlami. Špeciálnou vymoženosťou tejto aplikácie je systém na vyhľadanie svojho zaparkovaného auta. V prípade, že ho neviete nájsť, aplikácia spraví fotku miesta, kde sa nachádza. Ak ani toto nestačí, spustí GPS navigáciu k vozidlu. Aplikácia ponúka aj automatickú detekciu jazdy [15]. Ak pohyb mobilného zariadenia zodpovedá jazde autom, aplikácia začne snímať cestu. Naopak, ak stojíme v kolóne alebo na križovatke, aplikácia nás zbytočne neupozorňuje. Výsledky aplikácie závisia od viacerých faktorov: optiky fotoaparátu, schopnosti rýchleho a presného zaostrovania, procesora, ale aj od okolitých podmienok. Ideálne podmienku sú za jasného počasia, a zase naopak pri snežení, hmle aplikácia nemusí byť úplne presná. Aplikácia sa snaží nahradiť drahé systémy zabudované v nových autách. Určite nie je taká presná ako systémy priamo zabudované do auta od výrobcu, ale je to veľmi dobrá alternatíva. Aplikácia ihneď po zverejnení získala niekoľko ocenení. Jedným z nich bola cena za najlepšiu Android aplikáciu s rozšírenou realitou.

2.2.2 aCoDriver 5

Je veľmi podobná k aplikácii iOnRoad, o ktorej sme písali vyššie. Tiež sa zameriava na snímanie jazdných pruhov, ale aj na vzdialenosť vozidiel. Signalizuje príliš krátku vzdialenosť od vozidla a aj vybočenie z jazdného pruhu. Avšak, táto aplikácia je použiteľná len pri obmedzenej rýchlosti. Pri presiahnutí maximálnej rýchlosti, pri ktorej dokáže detekovať čiary a objekty pred vozidlom, prestáva byť aplikácia presná a použiteľná. Vodičovi dá signál o dočasnej nepoužiteľnosti aplikácie [4].



Obr. 2.2: Aplikácia aCoDriver 5 [4]

2.3 Projekt – Počítačové videnie pre nájdenie čiar

V tejto podkapitole sa budeme venovať projektu, ktorý nás najviac inšpiroval a niektoré postupy sme použili aj v našej práci. Tento projekt popísal a vytvoril Moataz Elmasry [11]. Na sledovanie vozovky použil kameru namontovanú na prednom skle auta. Samotný program vytvoril v Pythone za pomoci Opencv. Projekt pozostáva z nasledujúcich častí:

1. Kalibrácia kamery na odstránenie skreslenia. Výstupom z kamery je video, ktoré predstavuje sériu obrázkov. V dôsledku povahy šošoviek sú obrázky náchylné na skreslenie, ktoré vedie k nekonzistentnému zväčšeniu v závislosti od vzdialenosti objektu od optickej osi [11]. Na odstránenie skreslenia sa použije šachovnica, kde sa identifikujú rohy šachovnice, zistia sa odchýlky, a tie sa potom použijú pri snímkach z kamery.

2.Predspracovanie obrazu. Na detekciu pruhov na ceste sa použije b kanál z lab a l kanál z luv. Lab predstavuje farbu ako tri kanály: jasová zložka, zeleno-červená zložka a modro-žltá zložka [11]. Luv predstavuje transformáciu farebného priestoru XYZ, ktorý sa pokúša o percepčnú jednotnosť [11].

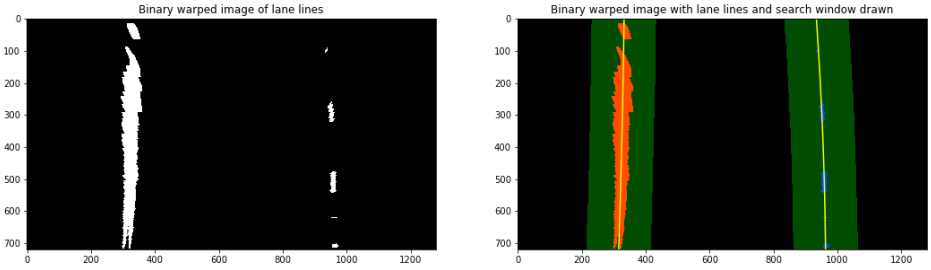
3.Perspektívna transformácia. V štandardnej perspektíve kamery sa objekty vzdialenejšie od fotoaparátu javia menšie a čiary jazdných pruhov sa zbiehajú, čím ďalej sú od auta. Jedným spôsobom, ako opraviť takéto skreslenie, je transformácia perspektívy obrazu tak, že sa pozeráme na obrázok zhora, tiež známe ako pohľad z vtáčej perspektívy. Najprv sa vyberie oblasť v obrázku, na ktorú chceme použiť transformáciu. Potom sa vyberú body predstavujúce cieľový priestor, do ktorého chceme segment transformovať [11].

4. Detekcia čiar na ceste. Najprv sa aplikuje binárne prahovanie [11]. Tým sa získa obraz s iba dvoma farbami. Biele pixely predstavujú čiaru z pruhu, ktorú sa snažíme

identifikovať. Dôležité je nájsť začiatočný bod ľavého a pravého pruhu. Jedným z prístupov je vygenerovanie histogramu pixelov. Histogram by mal obsahovať dva vrcholy, ktoré predstavujú ľavú a pravú líniu pruhu. Poloha dvoch vrcholov sa použije ako východiskový bod. Na vyhľadávanie ďalších bodov, ktoré označujú čiaru, sa využije technika vyhľadávania v posuvnom okne (sliding window). Začína zdola a iteratívne skenuje celú cestu až k vrchu obrázka a do zoznamu pridáva detekované pixely. Ak je v okne zistených dostatočný počet pixelov, vykreslí sa ďalšie okno okolo svojej strednej polohy. Podľa nájdených bodov sa spraví hladká čiara, ktorá sa chová ako najlepšia aproximácia toho, kde je čiara pruhu [11].

5. Poloha vozidla a jazdného pruhu. Nakoniec sa v projekte vypočíta poloha vozidla v jazdnom pruhu. Počíta sa s tým, že je kamera umiestnená v strede obrázka. Prevod z pixelov na metre bol počítaný pomocou rozlíšenia obrázku.

Výsledkom tohto projektu je veľmi presná aplikácia. Dokáže oveľa presnejšie zachytiť jazdný pruh. Má síce nevýhodu, že pri slabej viditeľnosti čiar môže mať problém s detekciou, ale napriek tomu má táto metóda budúcnosť. Pravdepodobne nie ako samostatná aplikácia, ale kombináciou iných systémov môže dokonale a presne označovať cestu.



Obr. 2.3: technika sliding window v porovnaní s binárnym obrázkom [11]

Obr. 2.4: Snímka z projektu so zakrivením jazdného pruhu a stred vozidla [11]

2.4 Analýza existujúcich riešení

V predchádzajúcich podkapitolách sme rozobrali fungovanie rôznych mobilných aplikácií, systémov pre autonómne autá.

LDWS a LKA sú profesionálne systémy, na ktorých sa podielalo množstvo vývojarov. Tieto systémy sú prepojené s rôznymi senzormi a ich presnosť je veľmi vysoká.

Čo sa týka analýzy mobilných aplikácií, tie sa spoliehajú na detekciu jednou kamerou. Tu je problém, že nám chýba určitý aspoň pomocný systém, ktorý by dokázal detekovať jazdný pruh v prípade zlých viditeľnostných podmienok. Avšak, tu mnohé aplikácie využívajú určitú predikciu čiar. Čiže v prípade, že na videu nie je možné rozonať čiary, tak podľa predchádzajúcich vlastností cesty určí jej pokračovanie. Toto samozrejme spôsobuje mierne odchýlky, ale je to dobrá alternatíva, ako sa s týmto problémom popasovať.

Projekt detekcie jazdných pruhov pomocou histogramu dáva nový pohľad na výzor, presnejšiu vizuálnu detekciu ako ponúka Houghova transformácia. Tento postup má veľa mínusov, avšak, v prípade kombinácie s iným systémom vie byť veľmi dobrý. Tieto mínusy sú spojené hlavne s dobrou viditeľnosťou a nastavením kamery. Samozrejme aj kalibrácia kamery je určitým problémom. Kamera sa musí vždy nakalibrovať a pre ideálne fungovanie je potrebné ju mať pevne upevnenú. Avšak, výhodou tohto postupu je aj to, že je jednoduchšie spraviť rôzne merania vzdialeností. Tieto informácie môžu zaujať vodiča a pomôcť mu.

Kapitola 3

Technológie

V tejto časti sa budeme venovať technológiám a algoritmom, ktoré sa využívajú pri tvorbe systému na detekciu čiar na ceste, ale aj všeobecné veci ohľadom programovacieho jazyku a knižníc. Spomenieme tu aj algoritmy, ktoré síce nie sú použité v našej výslednej práci, ale pri skúšaní rôznych variantov postupov detekcie sme sa s nimi stretli a sú veľmi využívané.

3.1 HP Pavilion Gaming 15-dk0010nc

Keďže sa jedná o aplikáciu, ktorá beží v programovacom jazyku Python, a nie je to mobilná aplikácia, tak v tejto časti opíšem parametre notebooku použitého na tvorbu a neskôr aj testovania programu. Jedná sa o notebook od firmy HP. Aj názov prezrádza, že ide o hráčsky počítač. Výhoda hráčskych počítačov je práve vysoký výkon, ale aj dobrá grafika, ktorú poskytuje. Tento notebook poskytuje 16GB RAM. Nachádza sa v ňom procesor Intel Core i7-8750H a čo sa týka grafickej karty, tak má v sebe NVIDIA GeForce GTX 1050/3GB, čo zvládne momentálne väčšinu grafickej náročnosti.

3.2 Xiaomi Redmi 4X

Tento mobilný telefón bol použitý na natáčanie videí pre túto prácu. Využíva osem jadrový procesor sprevádzaný grafickým akcelerátorom Adreno 50. Rozlíšenie obrazovky je 1280x720. Obsahuje zadnú kameru, ktorá má rozlíšenie fotoaparátu 13 Mpix a maximálne rozlíšenie videa, ktoré je možné natáčať, je full HD 1080p.

3.3 Python

Python je moderný programovací jazyk, ktorého popularita stále rastie. Jeho autorom je Guido van Rossum. Je používaný vo firmách ako Google, Youtube, Mozilla. Na

rozdiel od mnohých iných jazykov, ktoré sú kompilačné (napríklad Pascal, C/C++) je Python interpreter [5].

Porovnanie jazyku Python a Java:

Programy v Pythone bežia oproti Jave 3 až 5-krát pomalšie, ale zase ich vývoj trvá menej času. Tento rozdiel možno pripísať zabudovaným dátovým typom vysokej úrovne Pythonu a ich dynamickému písaniu. Napríklad python programátor nestráca čas deklarovaním typov premenných. Ale kvôli tomu sa čas spustenia predlžuje. Napríklad, keď si zoberieme výraz a+b, tak Python najskôr musí zistiť typ oboch daných premenných a až potom určiť výsledok, pričom Java tým, že to má zadefinované, tieto veci nemusí pri spustení riešiť [17]. Hlavné vlastnosti jazyka Python:

• veľmi jednoduchá a dobre čitateľná syntax, a keďže Python je aj vysoko interaktívny, je veľmi vhodný aj pre vyučovanie programovania [5].

• na rozdiel od staticky typovaných jazykov, pri ktorých je treba dopredu deklarovať typy všetkých dát, je Python dynamicky typovaný, čo znamená, že neex-istujú žiadne deklarácie [5].

• Python obsahuje pokročilé črty moderných programovacích jazykov, napríklad podpora práce s dátovými štruktúrami, objektovo-orientovaná tvorba softvéru a rôzne ďalšie [5].

• je to univerzálny programovací jazyk, ktorý poskytuje prostriedky na tvorbu moderných aplikácií, takých ako analýza dát, spracovanie médií, sieťové aplikácie a podobne [5].

• Python má obrovskú komunitu programátorov a expertov [5]. V tejto práci budem využívať verziu pythonu 3.8.1.

3.4 OpenCV

OpenCV je voľne dostupná knižnica pre počítačové videnie a strojové učenie. Knižnica obsahuje viac ako 2500 optimalizovaných algoritmov. Tieto algoritmy môžu byť použité na detekciu tvárí, objektov, rozoznávanie pohybu človeka na videu, extrahovať 3D model objektu a množstvo ďalších. Odhadovaný počet stiahnutí prekročil 18 miliónov. Je používaný od firiem cez vedecké skupiny až po bežných programátorov, či študentov. OpenCV je vytvorená pre programovacie jazyky C++, Python, Java a Matlab a je podporovaný operačným systémom Windows, Linux, Android a Mac OS. OpenCV – Python využíva Numpy, čo je vysoko optimalizovaná knižnica pre numerické operácie so syntaxou podobnú Matlabu. Všetky štruktúry matíc Opencv sa prevádzajú do polí Numpy [20].

3.5 Knižnice

V našej práci sa musíme zamyslieť nad tým, aké knižnice by sa nám mohli pri tvorbe výslednej aplikácie hodiť. V tejto časti si spomenieme na tie, ktoré budú potrebné. Prvou je knižnica OpenCV. Je opísaná v časti 3.4.

Ďalšou knižnicou, ktorú budeme potrebovať, je NumPy. NumPy poskytuje infraštruktúru na prácu s vektormi, maticami, poliami. Ponúka aj veľa matematických funkcií. My ju využijeme najmä pri práce s poliami a maticami.

Veľmi dôležitou knižnicou pre grafiku je tkinter. V nej budeme robiť celý výzor aplikácie s užívateľským prostredím. Tkinter slúži na vytváranie grafických aplikácií, umožňuje v Pythone vytvárať nové okná, pridávať tlačidlá, texty a rôzne ďalšie.

Ďalšou knižnicou, ktorá má potenciál na to byť využitá, je knižnica statistics. Jej potenciál vidíme hlavne na počítanie, napríklad najčastejšej hodnoty polohy okienok za určitý úsek alebo mediánu z pola. Statistics je knižnica na počítanie matematickej štatistiky. Jej funkcionalita má simulovať počítanie na vedeckej kalkulačke.

A poslednou knižnicou je time. Túto využijeme najmä pri testovaní rýchlosti našej aplikácie. Time poskytuje služby na prácu s časom, jeho merania, ale aj získania aktuálneho času.

Kapitola 4

Návrh aplikácie na detekciu pruhov

V tejto kapitole sa budeme venovať detailnému návrhu aplikácie na detekciu jazdných pruhov na ceste, ktorá bude slúžiť na zvýšenie bezpečnosti vodičov, ale aj ostatných účastníkov premávky. Možných postupov, ako takýto systém spraviť, je viacero. Vybral som si sledovanie pruhov mocou jednej kamery. Je to, aj čo sa týka nákladov, malá investícia. Aplikácia teda bude využívať video z kamery umiestnenej na čelnom skle auta. Môže sa použiť ľubovoľná kamera alebo aj kamera na mobile. Dôležitým prvkom je, aby bolo danú cestu vidieť. Ideálne nastavenie kamery je v strede čelného skla bez toho, aby niečo bránilo vo výhľade a aby nebránila vodičovi vo výhľade. Odporúčajú sa držiaky, ktoré dokážu zmierniť otrasy na videu. Tým sa dokáže predísť prípadným malým nepresnostiam v detekcii. Dnešné auto kamery alebo aj mobilné fotoaparáty dokážu zachytávať snímky pri rýchlosti minimálne 30fps. Výslednou aplikáciou bude počítačová aplikácia, ktorá bude po spustení detekovať jazdný pruh z videa. Mnohé aplikácie a systémy potrebujú nakalibrovať kameru a bez toho nie je možné úspešné používanie systému. Výhodou mojej aplikácie bude to, že kalibrácia kamery nie je potrebná.

4.1 Predspracovanie

Táto časť bude vykonávať základné operácie na vstupnej snímke. Na to, aby sme vedeli detekovať jazdné pruhy, nepotrebujeme vysoké rozlíšenie. Ľudia chcú mať z videa čo najlepší zážitok, preto je bežné rozlíšenie nastavené oveľa väčšie ako stačí nám. Preto sa najskôr zmenší video na rozlíšenie veľkosti 640x480 pixelov. Takýto obrázok má stále dostatočnú kvalitu na to, aby sa z neho dali vyčítať potrebné údaje na detekciu. Potom zmeníme obrázok z farebného na šedo úrovňový. Oblasť záujmu sa nachádza v dolnej časti snímky videa, kde sa nachádza povrch vozovky.

4.1.1 Gaussov filter

Obraz, ktorý dostane aplikácia na vstupe, môže obsahovať šum. Ten môže zabraňovať lepším výsledkom a lepšej detekcii. Preto sa použije Gaussov filter s konvolučným jadrom 5x5. Je vhodný na vyhladenie obrazu a pri odstránení Gaussovho šumu. „Gaussov filter má filtračné jadro odvodené od Gaussovej dvojrozmernej funkcie definované nasledovne“[10] :

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Obr. 4.1: Gaussovský filter veľkosti 5

4.1.2 Segmentácia

Hlavným cieľom segmentácie je rozdeliť obraz na časti, ktoré majú silnú koreláciu s objektami alebo oblasťami reálneho sveta zobrazenými v obraze. Pri čiastočnej segmentácii je cieľom rozdeliť obraz na časti, ktoré sú homogénne z hľadiska vybranej vlastnosti, napríklad jasu, farby, odrazivosti, textúry a podobne. Pri segmentácii šedoúrovňových obrazov využívame dve základné vlastnosti týchto obrazov: diskontinuitu a podobnosť. Segmentačné metódy možno rozdeliť na tri skupiny: prahovanie, segmentácia založená na hranách (diskontinuita), segmentácia založená na oblastiach (podobnosť). Nejednoznačnosť obrazových dát je hlavným segmentačným problémom, často sprevádzaným informačným šumom. Čím viac apriórnej informácie je k dispozícii pri segmentačnom procese, tým lepšie výsledky pri segmentácii možno dosiahnuť [13].

4.2 Prahovanie

Najjednoduchšia segmentačná technika, je výpočtovo nenáročná a rýchla. Na segmentovanie objektov od pozadia sa používa jasová konštanta, ktorá sa nazýva prah. Prahovanie je transformácia, ktorá zobrazuje vstupný obraz f (i, j) na výstupný obraz g (i, j) nasledovne:

$$g(i,j) = 1 \leftarrow f(i,j) >= T$$

$$g(i,j) = 0 \leftarrow f(i,j) < T$$

Niektoré metódy na určenie prahu sa snažia určiť prah automaticky. Na určenie prahu sa používajú metóda p-podielu, metóda analýzy tvaru histogramu a optimálne prahovanie [13].

4.3 Testovaný návrh podľa DP od Gabriela Mrvu

4.3.1 Canny edge detector

Vytvoril ho John F. Canny. Je tiež známy ako optimálny detektor a jeho cieľom je splniť tri hlavné kritériá. Prvé je nízka chybovosť. Tým sa myslí význam správnej detekcie iba existujúcich hrán. Druhou je dobrá lokalizácia. Ide o vzdialenosť medzi pixelmi okrajov detekovaných objektov a reálnych bodov objektu musí byť minimalizovaná. Poslednou je minimalizovanie odozvy teda očakáva sa iba jedna odozva detektora na hranu.Na začiatku použijeme filter na odstránenie šumu. Potom po získaní veľkosti a smeru gradientu sa vykoná úplné skenovanie obrázka, aby sa odstránili všetky nežiadúce pixely, ktoré nemusia byť okrajom. Výsledkom je binárny obrázok. Posledným krokom je prahovanie. Tento pojem je vysvetlený v časti 3.2.1. článku [3].

4.3.2 Houghova transformácia

Houghova transformácia je segmentačná technika použiteľná vtedy, keď treba detekovať objekty so známym tvarom hranice. Houghova transformácia môže detekovať rovné čiary aj krivky (hranice objektu), ak sú známe ich analytické vyjadrenia. Je robustná pri identifikácii zakrytých a zašumených objektov.

Houghova transformácia je výpočtovo náročná operácia. Počet priamiek, ktoré transformácia skúša, je obrovské množstvo. Budeme vychádzať z toho, že priamky nemôžu byť vodorovné, keďže jazdné pruhy také nie sú. [2] Houghova transformácia je implementovaná v OpenCV funkciou HoughLines(). Má viacero vstupných parametrov, ako binárny obrázok, prah a parametre presnosti p, Théta. Funkcia vracia pole (p, Théta) hodnôt. p sa meria v pixeloch a Théta v radiánoch.[16]

4.4 Výsledný návrh

4.4.1 Threshold Binary: binárne prahovanie

Na zistenie toho, kde sa nachádza pruh, si potrebujem snímku upraviť. Potrebujeme zvýrazniť a hlavne rozlíšiť čiaru na ceste od ostatného. Spravíme to pomocou binárneho prahu. Pre každý pixel sa použije rovnaká prahová hodnota. Ak je hodnota pixla menšia ako prahová hodnota, je nastavená na 0. Inak je nastavená na maximálnu hodnotu. Na aplikovanie prahovania sa používa funkcia cv.threshold. Prvým argumentom je zdrojový obrázok, ktorým by mal byť šedo úrovňový obrázok. Druhým argumentom je prahová hodnota, ktorá sa používa na klasifikáciu hodnôt pixlov. Tretím argumentom je maximálna hodnota, ktorá je priradená hodnotám pixelov prekračujúcich prahovú hodnotu. OpenCV poskytuje rôzne typy prahovania, ktoré sú dané štvrtým paramet-

rom funkcie. Základné prahovanie sa vykonáva pomocou typu cv.THRESH BINARY [19].

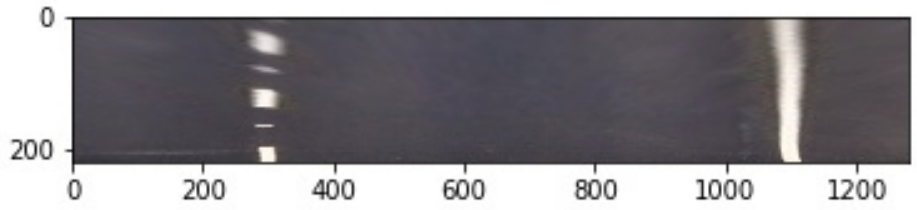
4.4.2 Pohľad z vtáčej perspektívy

V tejto časti sa pozrieme na funkciu, ktorá sa začína čím viac používať v počítačovom videní. Má obrovský potenciál do budúcna vo viacerých smeroch, ale aj čo sa týka samotného sledovania čiar na ceste. Vďaka nej bude možné merať vzdialenosti objektov od seba, ale aj samotná detekcia môže mať lepšie výsledky. Pri tejto funkcii sa zvyčajne používa kalibrácia kamery. Je to kvôli tomu, aby bola výsledná snímka čo najpresnejšia, lebo mnohé kamery spôsobujú významné skreslenie[18]. Na kalibráciu kamery sa veľmi často využiva šachovnica, ktorá sa odfotí z rôznych uhlov. Práve na týchto snímkach sa daná kalibrácia vykoná. Predpoklady na to, aby mohla byť kalibrácia použitá je stabilné nastavenie kamery. To znamená, že kamera by musela byť pevne upevnená na aute a podľa tohto pohľadu sa kamera kalibrovala. Avšak, po jej posunutí by sa presnosť znížila. Takže kalibrácia sa oplatí robiť len v prípade, ak by išlo o kameru zabudovanú priamo v aute. Avšak, v našom prípade, kde chceme, aby bol systém čo najviac univerzálny, to nie je úplne potrebné. Pri detekcii čiar nebude potrebné mať všetky údaje úplne presné, ale využijeme ich blízkosť k realite. Ale čo sa týka budúceho vývoja tohto systému, bude určite dobré rozmýšlať aj nad pridaním kalibrácie kamery.

Z pohľadu vodiča na cestu je vidieť, že sa rovnobežné pruhy zbližujú. Tento jav vieme vyriešiť pomocou funkcie ktorá pretvorí pohľad na pohľad z vtáčej perspektívy. Najskôr sa vyberá oblasť záujmu. Teda štvoruholníková časť zo snímky, ktorú určíme bodmi jeho rohov. Snímku je potom možné transformovať do pohľadu z vtáčej perspektívy dvoma spôsobmi[12]:

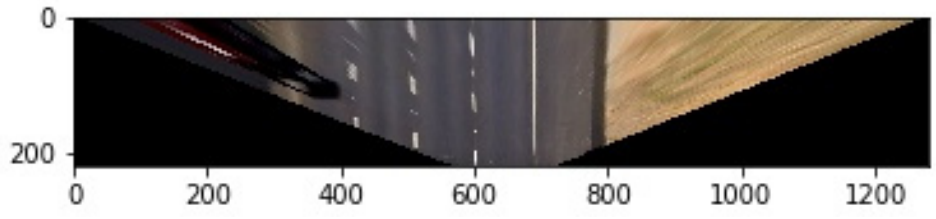
a) Roztiahnutím hornej časti snímky a spodná časť ostáva nezmenená, ako je možné vidieť na Obr.4.2. [12].

b) Zmenšením spodnej časti snímky a horná časť ostáva nezmenená, ako je možné vidieť na Obr.4.2. [12].



Obr. 4.2: Roztiahnutie hornej časti obrázka. [12]

Na výpočet bodov na výslednom obrázku sa používa metóda násobenia matíc. Každý z bodov, ktoré sa určili na pôvodnom obrázku, sa vynásobí maticami, podľa



Obr. 4.3: Zmenšenie dolnej časti obrázka. [12]

toho ako chceme daný bod transformovať. Využívajú sa tri matice: Translácia, škálovanie a rotácia. c = cos ϕ , s = sin ϕ uhol rotácie počítaný proti smeru hodinových ručičiek.S_u, S_v a x_u, x_v sú faktory mierky a hodnoty translácie horizontálny a vertikálny smer

$$H_s = \begin{bmatrix} S_u & 0 & 0 \\ 0 & S_v & 0 \\ 0 & 0 & 1 \end{bmatrix} H_r = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} H_t = \begin{bmatrix} 1 & 0 & x_u \\ 0 & 1 & x_v \\ 0 & 0 & 1 \end{bmatrix}$$

Obr. 4.4: Matice translácie, rotácie a škálovania.

4.4.3 Detekcia jazdných pruhov

Predchádzajúce kroky sú predprípravou na to, aby sme boli schopný, čo najlepšie detekovať čiary na ceste. V tejto časti sa budeme venovať idei, ako hľadať jazdné pruhy. Každý obrázok sa skladá z pixelov a každý má pozíciu súradníc a okrem toho aj informáciu o farbe. Tým, že v tejto fáze je snímka čierno-biela, tak viem, že farebné zložky budú len dve. Buď 0 alebo 255. Na zistenie pozície začiatku čiary použijem postup súčtu hodnôt farebných zložiek pixlov podľa x-ovej súradnice. V bode, kde bude súčet najväčší, môžeme predpokladať, že práve tam má začiatok čiara definujúca jazdný pruh. Neskôr budem od tohto bodu hľadať pixely, ktoré nemajú nulovú farebnú zložku. Budem postupovať po štvorcoch, takzvané “sliding windows”, a vždy od posledného vyššie hľadať nenulové farebné zložky pixelov. Týmto vykreslením všetkých štvorčekov získam vyznačený jazdný pruh. Na detekciu pruhu v prípade slabej viditeľnosti čiar sa pri použití Houghovej transformácie používa napríklad Kalmanov filter. V našom prípade využijeme získané predchádzajúce dáta. V prípade, že v niektorom momente stratíme čiary na ceste, vieme predpokladať, ako má daná cesta približne vyzerať. Základom bude udržiavať určitú množinu údajov na dopočítanie pruhu v prípade takýchto problémov.

4.5 Testované návrhy aplikácie

Prvý postup, ktorý sme sa rozhodli spraviť, bol inšpirovaný prácou od Gabriela Mrvu s názvom Rozšírenie riadenia vozidla počítačovým videním, ale aj rôznymi mobilnými amatérskymi aplikáciami na mobil. Práve návrh, ktorý opíšeme, v tejto časti je momentálne najpopulárnejší postup, čo sa týka detekcie čiar.

Ako vstup sme použili video s rozlíšením 640x480 pixelov. Takéto rozlíšenie bolo dostatočne malé a zároveň ponúkalo dostatočné informácie na detekciu čiar. Prvým krokom bola zmena obrázku na šedo úrovňový. Tento krok je pri väčšine prác rovnaký, lebo farby snímky nás úplne nezaujímajú, ale prakticky iba odtiene. Ďalším krokom je nastavenie oblasti záujmu, teda miesta na snímke, kde predpokladáme, že sa bude nachádzať jazdný pruh. Týmto získame menšiu oblasť a pri vyhodnocovaní a hľadaní výsledkov nám to dá menej výsledkov, a samotné vyhodnocovanie je jednoduchšie. Cannyho hranový detektor je nasledujúcim použitým algoritmom. V našom prípade chceme detegovať hrany, ktoré sa nachádzajú na snímke. Predtým, než začneme detegovať čiary, je potrebné zbaviť snímky šumu. Každý takýto šum by mohol znamenať určitú nepresnosť pri detekcii a hľadaní hrán. Na typ šumu vytvorený kamerou použijeme Gaussov filter, ktorý je popísaný v časti 3.4. Po zbavení šumu sme spustili Cannyho hranový detektor. Tým, že máme určenú oblasť záujmu, tak sa nájdené hrany nachádzali len v nej. Potom sme použili Houghovu transformáciu. Ňou sme na snímke našli všetky čiary, ktoré kopírovali hrany. Pri prerušovanej čiare sme spojili nájdené čiary a vytvorili jednu dlhú čiaru. Problémom pri detekcii Houghovou transformáciou je, že v prípade zákruty je detekcia v hornej časti pomerne nepresná.

Obr. 4.5: Zlá detekcia pri zákrute. [6]

Aj z tohto dôvodu sme sa rozhodli si vybrať nový postup, ktorý by mohol potencionálne priniesť o niečo lepší a krajší výsledok. Napriek tomu je tento postup veľmi dobrý a prináša dobré výsledky. Napríklad v diplomovej práci Gabriela Mrvu s názvom Rozšírenie riadenia vozidla počítačovým videním je úspešnosť na jeho záberoch 90,80 percent. Ale u iných projektov bola úspešnosť nižšia. Na rovnej ceste bola detekcia veľmi dobrá, ale v prípade, že bola zákruta, sa presnosť znížila.

Nový postup, s ktorým sme sa rozhodli zaoberať, využíva technológiu na sledovanie čiar pomocou hodnôt z histogramu a vykreslovaním okienok na pozíciach, kde bol zistený jazdný pruh. Detail je opísaný v návrhu v časti 4.4. Nami vybraný nový postup sa zdá podľa zdrojov na výzor veľmi dobrý s potenciálom do budúcna. Avšak, z dát, ktoré máme k dispozícii, nevieme povedať, ako by daný systém reagoval v prípade zhoršenia podmienok, zlej viditeľnosti, rôznych zákrutách. Ale aj napriek tomu je dobré daný postup vyskúšať a zistiť na koľko môže byť využiteľný pri detekcii čiar.

Kapitola 5

Riešenie problematiky a implementácia

5.1 Získanie obrazu

Spôsob, akým sme získali obraz, je popísaný v úvode kapitoly 4. Obraz bol natočený na mobilné zariadenie Xiaomi Redmi 4X. Získané videá sú uložené vo formáte MP4. Samozrejme, je možné spúšťať aj iné typy videí s rôznym formátom. Získané videá boli natočené za dobrých svetelných podmienok počas dňa, ale aj pri horších v podvečerných hodinách.

Obr. 5.1: Svetelné podmienky-šero

5.2 Postup pri implementácií

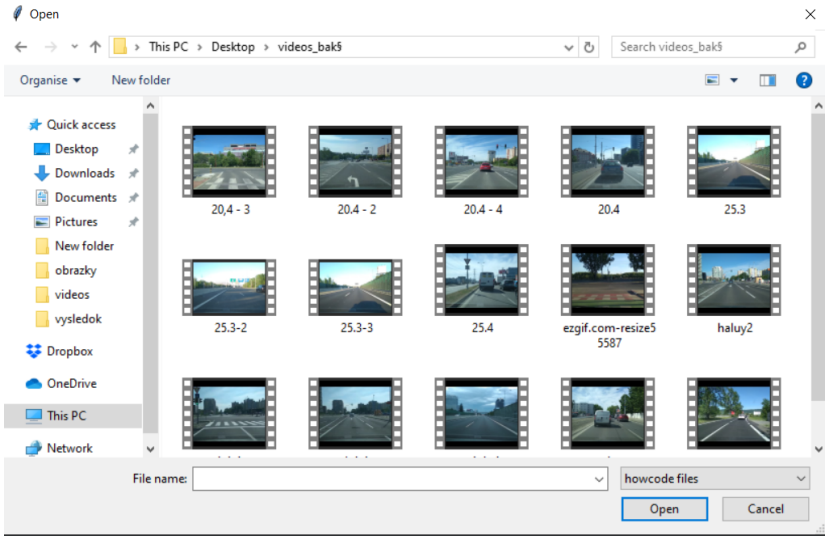
V riešení problematiky sme postupovali podľa návrhu aplikácie opísanej v kapitole 4. Pri samotnej tvorbe programu a jeho testovaní sme zistili, že náš postup nie je úplne ideálny a výsledky nebudú také, ako sme na začiatku predpokladali. Avšak aj napriek

Obr. 5.2: Svetelné podmienky-slnečno

tomu malo veľký význam vyskúšať tento postup. Problém bol v časovej náročnosti algoritmu na detekciu, kedže program má bežať v reálnom čase. Program sme rozdelili na dve časti. Prvou je GUI používateľské rozhranie. V tejto časti sa venujeme hlavne práci s Tkinterom. Druhá časť vykonáva prácu s videom.

5.3 Používateľské rozhranie

Používateľské rozhranie sme spravili jednoduché a prehľadné. Používateľ má možnosť prehrať video z jazdy pomocou tlačidla "načítaj video". Po stlačení sa ukáže dialógové okno, pomocou ktorého si môže užívateľ vybrať video. Obr.5.3 ukazuje dialógové okno otvorené užívateľom. Ihneď po spustení videa začne detekcia jazdného pruhu na ceste.



Obr. 5.3: Dialógové okno na výber videa

Video sa spustí v novom okne. Ďalej je k dispozícii možnosť zastaviť/pokračovať. Užívateľ môže hocikedy zastaviť alebo pustiť video pomocou tlačidla "p"na klávesnici. V

Canvase sa pred spustením programu bude nachádzať manuál pre užívateľa. Bude tam v skratke vysvetlený postup na prácu s programom.

Obr. 5.4: GUI-používateľské rozhranie

5.4 Predspracovanie

Po načítaní videa sme upravili každú snímku a pripravili ju na detekciu. Ako prvé sme nastavili snímku na šedo úrovňovú volaním funkcie self.gray video().

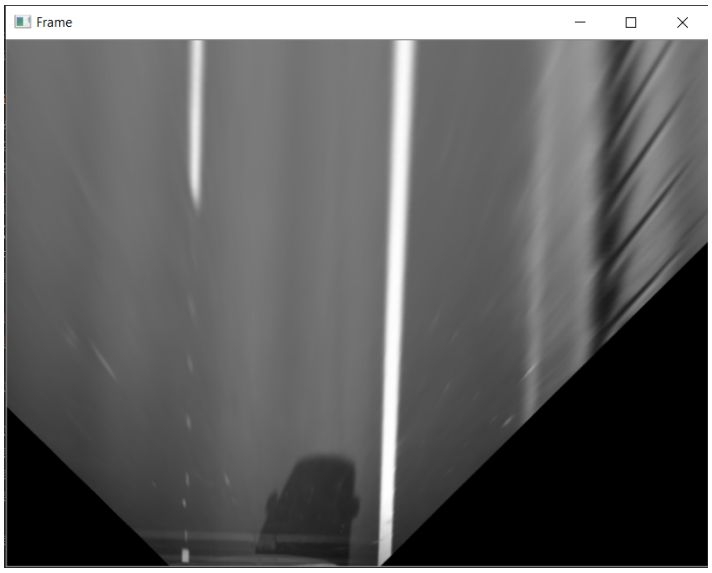
cv2.cvtColor(self.frame, cv2.COLOR BGR2 GRAY)

Ako ďalšie sme použili Gaussov filter na odstránenie šumu na snímke o veľkosti 5.

self.frame = cv2.GaussianBlur(self.frame,(5,5),0)

Chceme dostať čierno-biely obrázok. Preto sme použili binárne prahovanie, ktoré nám nastaví čiary na maximálnu bielu a zvyšok na čiernu. Hodnoty parametrov boli určené pri testovaní na každé video samostatne. Avšak, ešte predtým, ako aplikujeme binárne prahovanie, spravíme transformáciu obrázku. Tá je popísaná v časti 5.5. Pomocou nej dostaneme oblasť, ktorá je pre nás zaujímavá a potencionálne sa na nej majú nachádzať čiary.

cv2.threshold(self.frame, 205, 255, cv2.THRESH BINARY)



Obr. 5.5: Obrázok pred použitím binárneho prahovania.

Obr. 5.6: Obrázok po použití binárneho prahovania.

Tabuľka 5.1: Hodnoty parametrov binárneho prahovania pre videá

video1	video2	video3	video4
<180,255>	<200,255>	<130,255>	<155,255>

5.5 Transformácia

Ďalším krokom je transformácia snímky vo videu. Použijeme funkciu self.bird eye view(), ktorá transformuje obrázok do pohľadu z vtáčej perspektívy. V tejto funkcii sú určené štyri body, ktoré chceme transformovať. V našom prípade to máme určené stabilne a nastavené tak, aby zobrazili pruh pred autom a aj okolie pruhu kvôli zákru-

tám. Potom sú určené aj štyri body na novom obrázku, kde sa má daná transformácia ukázať.Tieto dva parametre sa vložia do funkcie cv2.getPerspectiveTransform(src, dst). Ňou dostaneme transformačnú maticu 3x3. Potom použijeme rovnakú funkciu, ale vstupné parametre vymeníme a z toho dostaneme inverznú maticu. Po získaní týchto matíc po zavolaní funkcie,

cv2.warpPerspective(self.frame, M, imgSize, cv2.INTER LINEAR)

kde parameter self.frame obsahuje obrázok, z ktorého má byť transformácia vykonaná. M predstavuje transformačnú maticu. A ďalšími sú veľkosť obrázka a kombinácia interpolačných metód. Pomocou týchto funkcií dosiahneme obrázok, ktorý nám dá vo výsledku pohľad z vtáčej perspektívy. Výsledok našej transformácie je vidieť na obrázku: Obr.5.5.

5.6 Detekcia jazdného pruhu

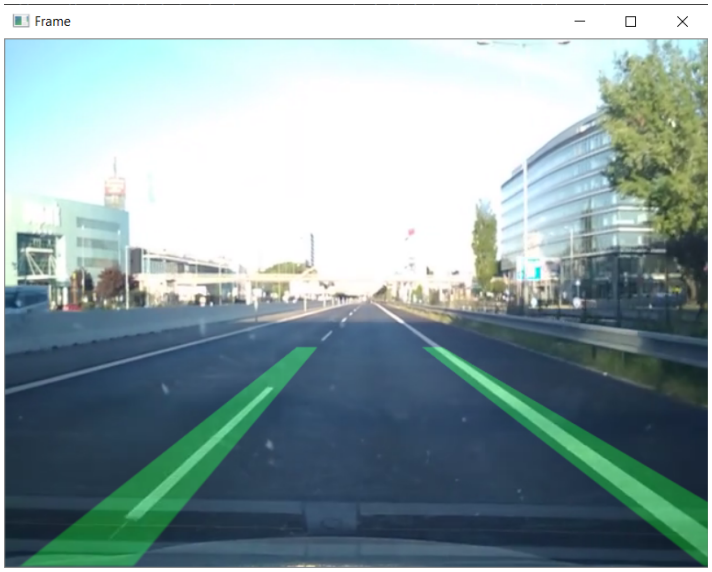
Na detekciu čiar na ceste využívame metódu sliding windows. Tá z videa, ktoré je čierno-biele pomocou binárneho prahovania, získa pozície, kde sa daná čiara má nachádzať. Ako vstupné parametre funkcie self.sliding windows() sú obrázok, ktorý je binárny, farba, akou chceme zvýrazňovať čiary na ceste. Ďalším je počet okienok nwindows, ktoré chceme vykresliť na dĺžke detekcie pruhu. Parameter margin určuje okraje vykreslovaného okienka a posledný parameter minpix určuje koľko minimálne pixlov musí byť nájdených, aby sa daná stopa mohla rátať ako čiara. Tri parametre sú nastavené, a to nwindows,margin a minpix. Tieto parametre môžeme zmeniť v prípade, že by sme chceli pozmeniť zobrazovanie veľkosti okienok. Parameter nwindows je predvolene nastavený na 30. Táto hodnota nám vyšla ako najlepšia, čo sa týka výzoru vykreslenia detekovanej čiary a zároveň presnosť detekcie je tiež veľmi dobrá. Margin je nastavený defaultne na hodnotu 15 a to z vizualizačného dôvodu. Pri testovaní aplikácie nám vyšlo, že takýto okraj je dostatočný na detekciu čiar a zároveň nie je príliš veľký a rušivý. A ako posledná predvolená hodnota je hodnota parametru minpix. Táto je nastavená na 1. Najmä pri horších podmienkach je dôležité mať tento parameter čo najmenší, a to z dôvodu horšej viditeľnosti celej čiary.

self.sliding windows(self, img, color, nwindows=30, margin=15, minpix=1)

Vo funkcii sliding windows sa na začiatku zavolá funkcia self.get hist(img), ktorej vstupný parameter je snímka z videa. Prechádzame hodnoty farby každého pixela na x-ovej osi. Sčítame všetky hodnoty na danej X-vej súradnici. Na konci získame pole takýchto hodnôt, ktoré nám daná funkcia vráti. Vrcholy, teda body, kde začína čiara, získame z pola sčítaných hodnôt. Hľadáme maximum na ľavej strane snímky a potom

na pravej. K pravej strane rátame aj stredný bod kvôli tomu, aby sme predišli prípadnej chybe detekcie vyplývajúcej z tohto problému.

Ďalšou funkciou je self.nonzero img(img), ktorá vracia pole nenulových pixelov. V nej ich však nespočítavame, a to preto, lebo z nich vieme zistiť polohu pokračovania čiary. A posledným krokom je nastavenie výšky okna, v ktorom chceme vykreslovať a detekovať jazdné pruhy.Tento parameter sa pri kreslení delí počtom okienok, ktoré chceme vykresliť. Týmto si aj určíme, v ktorom rozmedzí na y súradnici budeme hľadať biele hodnoty pixelov. Potom už prechádzame na samotné vykreslovanie okienok. Na nájdené miesto, kde sa má čiara nachádzať, sa vykreslí okienko o šírke parametru margin, v našom prípade 15, a výška závisí od našej zadanej výšky, na ktorej chceme detekovať čiaru. Týmto dosiahneme to, že aj pri miernej odchýlke dokážeme zachytiť čiaru pomocu okienka, kedže mierne presahuje hrúbku čiary. Po vykreslení štvorčekov na ľavej aj pravej strane sa pustíme do hľadania pokračovania čiary na ceste. Hľadáme hodnoty po riadkoch výšky štvorčeka. V prípade, že nie je nájdený žiaden pixel v danom riadku alebo je príliš vzdialený, použijeme funkciu self.checkifcond(), ktorá skontroluje, či môže byť daný pixel čiara, teda či odpovedá približne možnému pokračovaniu pruhu alebo nie. V prípade, že neodpovedá, resp. nie je nájdený pixel, vypočítame pomocou poslednej vzdialenosti čiar odpovedajúcu pozíciu. V prípade, že nevieme nájsť ani jeden jazdný pruh, vykreslujeme okienka podľa x súradnice predchádzajúceho štvorčeka. V tejto funkcii je samozrejme viacero kontrolných podmienok, aby sa predišlo zbytočným nepresnostiam pri detekcii jazdného pruhu.



Obr. 5.7: Detekcia pruhu

Obr. 5.8: Zmena jazdného pruhu

5.7 Signalizácia vybočenia z pruhu

Funkcia vybočenia z pruhu kontroluje pozíciu čiar na ceste. V prípade, že sa pozícia veľmi zmení, program ukáže symbol ' <—> ', ktorý symbolizuje zmenu pruhu. Túto funkcionalitu nastavuje funkcia self.lane change().

5.8 Spätná transformácia a výsledná snímka

V tejto časti programu máme snímku transformovanú na pohľad z vtáčej perspektívy.Sú na nej detekované čiary určujúce jazdný pruh. Avšak, ako výsledok nám toto nestačí. Ďalším krokom je inverzná transformácia obrázka. Na to použijeme inverznú maticu transformácie k pôvodnej. Tú sme získali pomocou obrátenia vstupných parametrov funkcie getPerspectiveTransform:

cv2.getPerspectiveTransform(dst, src)

Pomocou nej môžeme získať obraz v pôvodnej perspektíve. Využijeme na to rovnakú funkciu ako pri transformácii do pohľadu z vtáčej perspektívy:

cv2.warpPerspective(self.frame, Minv, img_size, cv2.INTER_LINEAR)

Vstupné paramtre sú: snímka transformovaná do pohľadu z vtáčej perspektívy, inverzná matica, veľkosť snímky a posledným je interpolačná metóda. Keď máme snímku v pôvodnej perspektíve ostáva nám len spojiť túto snímku s detekovanými čiarami s pôvodnou snímkou. Na to sme využili funkciu addWeighted(), ktorá vypočíta vážený súčet dvoch polí. Tieto polia u nás predstavujú snímku.

```
cv2.addWeighted(img1, 1, self.frame, 0.3, 0)
```

Ako naše vstupné parametre funkcie sú: pôvodný obrázok, jeho váha pri súčte, obrázok s detekovanými čiarami a jeho váha pri súčte. Parameter 0.3 pre obrázok, na ktorom sú detekované čiary, nám pri testovaní vyšiel ako veľmi dobrý a výsledok dobre viditeľný.

Kapitola 6

Vyhodnotenie a výsledky

Pomocou rôznych videí natočených v iných jasových podmienkach a na rôznych miestach sme testovali náš program. Testovanie sme rozdelili na tieto oblasti:

- čas rýchlosti metód,
- testovanie úspešnosti nájdenia čiary,
- celková úspešnosť detekcie.

Vo výsledných tabuľkách sme úspešnosť počítali vzťahom: úspešnosť = správne/všetky

6.1 Čas rýchlosti metód

V tejto časti sme otestovali rýchlosť jednotlivých metód. Na testovanie bol použitý rovnaký počítač ako pri tvorbe. Obsahuje v sebe procesor Intel i7-8750H s frekvenciou 2.2GHz.

Tabuľka 6.1: Čas rýchlosti metód

Metóda	čas (milisek.)
Predspracovanie	0.8079
Transformácia	0.9488
Detekcia pruhu	7.2572
Inverzná transformácia a výsledná snímka	1.4801

V tabuľke 6.1 sú zobrazené hodnoty trvania jednotlivých častí programu v milisekundách. Tieto hodnoty boli namerané pomocou knižnice time na jednom videu. Výsledné časy sú priemer zo všetkých snímok z videa. Najdlhšie trvá metóda detekcie pruhu.Priemerná dĺžka bola nameraná na úrovni 7.2572 milisekundy. Pri tejto metóde sme s tým, samozrejme, rátali, pretože je výpočtovo najnáročnejšia. Ako druhý zaujímavým časom je čas metódy Inverznej transformácie a výslednej snímky. Táto časť

programu je druhá najdlhšia. Jej čas 1.4801 milisekundy je vyšší ako u samotnej transformácie. A je to spôsobené najmä spajaním transformovaného obrázka s pôvodným.

6.2 Testovanie úspešnosti nájdenia čiary

Tabuľka 6.2: Úspešnosť nájdenia čiar podľa okienok

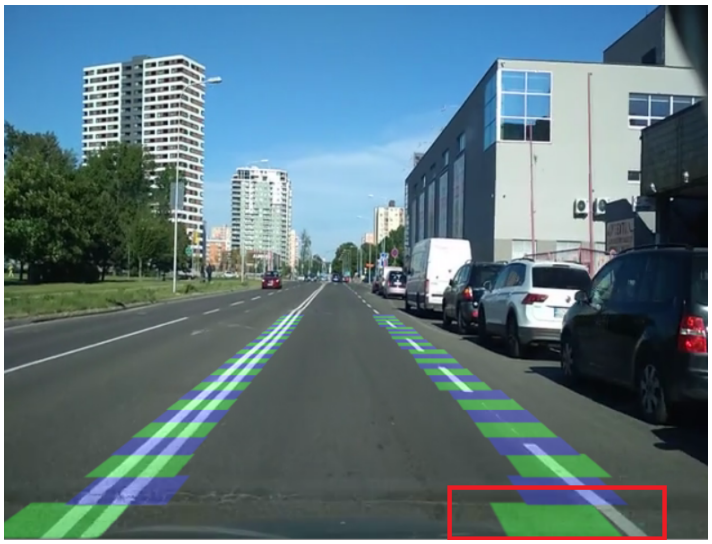
podmienky	správne	nesprávne	úspešnosť v percentách(%)
deň	30009	4251	87,6
večer	17920	3500	83,7
zložité situácie	8023	2657	75,1

Pri tomto teste sme zisťovali presnosť sledovania čiar na ceste. Prešli sme veľké množstvo snímok, na ktorých sme ručne zisťovali počet okienok, ktoré sú na správnej pozícii. Okienko bolo na správnej pozícií vtedy, keď okienko prekrývalo celú čiaru. V prípade, že čiaru nezakrývalo, brali sme to ako nesprávne nájdenie. Detekciu som rozdelil na tri oblasti. Prvou boli ideálne podmienky počas dňa za dobrého svetla. Vtedy dosiahol program presnosť 87,6 percenta. Druhou boli ideálne podmienky vo večerných hodinách. Tam sme získali presnosť 83,7 percenta. A poslednou oblasťou testovania bola detekcia jazdného pruhu v zložitejších situáciach. V našom prípade tam boli zahrnuté tiene na ceste, odbočka na križovatke a preraďovanie z jedného pruhu do druhého. Pri tomto testovaní sme získali presnosť 75,1 percenta. Ako dobré stavy sme brali, ak program detegoval pri preraďovaní buď obe krajné čiary s tým, že strednú ignoroval, alebo ak nedetegoval ani jednu čiaru a vytvoril si pruh pred autom. Posledná možnosť je, ak detegoval pruh, do ktorého auto prešlo. V prípade, že nastal jeden z týchto stavov, spočítali sme počet správne umiestnených okienok. Ak nenastal ani jeden z vyššie uvedených stavov, detekciu okienok sme brali ako nesprávnu. V prípade, že program zistí, že je prudká zákruta a pri detekcii sa zistia detegované pixely mimo cesty, tak sa najskôr zmenší počet okienok, a ak to nestačí, tak sa detekcia na chvíľu vypne. V prípade vypnutia sme tento stav nezarátavali ani ako správny, a ani ako nesprávny.

6.3 Celková úspešnosť detekcie

V tejto časti sme zfinalizovali celkovú úspešnosť daného programu. Vychádzali sme z údajov z časti 6.2. Z tých údajov nám vyšli percentá úspešnosti.

V týchto dvoch tabuľkách sa nachádzajú výsledky z celkového pohľadu na presnosť detekcie čiar. Dostali sme dva výsledky, ktoré odzrkadľujú celkovú presnosť programu.



Obr. 6.1: Chyba pri detekcii čiary na ceste

Tabuľka 6.3: Celková úspešnosť nájdenia čiar podľa počtu správnych

Celkom	správne	nesprávne	úspešnosť podľa počtu správnych(%)
66360	55952	10408	84,3

Tabuľka 6.4: Celková úspešnosť nájdenia čiar podľa percent z jednotlivých oblastí

deň(%)	večer(%)	špeciálne situácie(%)	úspešnosť podľa percent(%)
87,6	83,7	75,1	82,1

Prvá tabuľka obsahuje celkový počet kontrolovaných okienok. Jej druhý údaj obsahuje počet okienok, ktoré dobre detegovali čiaru a tretí údaj je počet okienok so zlou pozíciou. Výsledok, ktorý sme dostali, dosiahol 84,3 percenta. Tento výsledok nám dáva môžnosť sa pozrieť na úspešnosť na testovacích dátach. Avšak, všeobecná výpovedná hodnota pre celkovú úspešnosť je menšia, keďže počet kontrolovaných štvorčekov pre všetky tri oblasti nie je rovnaký počet. Týmto zvyšujeme mierne percentá, pretože najviac kontrolovaných údajov máme počas ideálnych podmienok.

V druhej tabuľke sme vychádzali z percent dosiahnutých v jednotlivých oblastiach testovania. Týmto sme chceli dosiahnuť väčšiu presnosť pri celkovom výsledku. Výsledkom je vypočítaná úspešnosť na úrovni 82,1 percenta.

Ako môžeme vidieť, v prvej tabuľke nám vyšiel výsledok, ktorý je mierne vyšší ako v druhej tabuľke, čo je spôsobené práve vyšším počtom otestovaných okienok počas ideálnych podmienok počas dňa.

Záver

Cieľom tejto bakalárskej práce bola implementácia aplikácie na detekciu jazdných pruhov a naštudovania problematiky v tejto oblasti. Táto skúsenosť je veľmi pozitívna, lebo nám dala možnosť nahliadnuť do oblasti počitačového videnia a zorientovania sa v tejto problematike. Pri sledovaní štatistiky úmrtnosti a nehodovosti na Slovensku, ale aj v zahraničí, sme si uvedomili, že každý takýto bezpečnostný systém môže zachrániť ľudský život. Preto vylepšovanie už existujúcih aplikácií je veľmi dôležité, aby mali aj bežní ľudia prístup k bezpečnostým systémom, aj keď si nechcú alebo nemôžu kúpiť drahé auto. Sledovanie a analýza týchto programov nám ukázala ich výhody, ale i nevýhody. Presnosť je napriek mnohým problémom veľmi vysoká a v mnohých prípadoch môže zachrániť život.

V práci sme predstavili mnohé takéto programy od mobilných aplikácií až po bežné profesionálne systémy pre autá. Tiež sme sa venovali detailne návrhu jedného systému, ktorý sme si rozobrali do detailov. Na základe neho sme sa inšpirovali v písaní vlastného návrhu. V ňom sú vysvetlené postupy a algoritmy použité neskôr v práci. V časti technológie a algoritmy sme si vysvetlili niektoré pojmy použité v bakalárskej práci. Niektoré z nich neboli použité priamo v implemetácii, ale sú často používané pri tejto problematike.

Na základe nášho návrhu sme prešli na implementáciu. V nej sme sa sústredili na použitie vhodných algoritmov a ich vylepšení. Po samotnej implementácii sme testovali rýchlosť jednotlivých metód a presnosť detekcie čiar. Počas vývoja aplikácie nám napadli rôzne veci a vylepšenia, ktoré by našu aplikáciu spravili oveľa používanejšou. Jedným z nich je vytvorenie verzie pre Android, kedže knižnica OpenCV je kompatibilná s operačným systémom Android.

Nami vybraný postup je na konci použiteľný samostatne menej, aj keď dáva pomerne dobré výsledky. Očakávané výsledky boli na začiatku oveľa vyššie, ako sme na konci dostali. Bolo to spôsobené najmä veľkou náchylnosťou na chyby pri detekcii. Ale čo sa týka budúceho vývoja, tam vidím pomerne veľký potenciál.V kombinácii s iným systémom by to mohol byť veľmi dobrý program na detekciu čiar, vzdialenosti či rýchlosti. Aj toto je pre mňa výzva, aby som sa ďalej sústredil a objavoval nové veci v tejto oblasti a neskôr vylepšiť moju bakalársku prácu na profesionálny systém, ktorý dokáže zachrániť ľudí.

Literatúra

[1] D. Levi A. B. Hillel, R. Lerner and G. Raz. Recent progress in road and lane detection: a survey, machine vision and applications, vol. 25, no. 3. 2014.

[2] Abid K. A. Mordvintsev. Hough line transform. *https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html*, 2013.

[3] Abid K. A. Mordvintsev. Open cv tutorials - canny edge detection. *https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html*, 2013.

[4] apkpure. acodriver 5, 5.0.1. *https://apkpure.com/acodriver-5/com.evotegra.aCoDriver*, 27.5.2017.

[5] A. Blaho. Programovanie v pythone, 3.vydanie. *http://input.sk/pdf/python.pdf*, 2018.

[6] CampusHippo. Learn detect highway lane lines with opencv and python. *https://campushippo.com/lessons/detect-highway-lane-lines-with-opencv-and-python-21438a3e2.*

[7] Volvo car UK. A million lives saved since volvo invented the three-point safety belt. *https://www.media.volvocars.com/uk/en-gb/media/pressreleases/20505*, 13.8.2009.

[8] Phooi Yee Lau Chee, Winserng. A framework for lane departure warning system for various lane markings, in proceedings of international workshop on advanced image technology, penang, malaysia. 2017.

[9] D. Dorian. Tesla adds lane-keeping assist through a software update. *Car and Driver máj 2019*, 3.5.2019.

[10] W. Benešová Z. Haladová J. Kučerová E. Šikudová, Z. Černeková. Počítačové videnie detekcia a rozpoznávanie objektov. *Praha: vydavateľstvo Wikina.*, 2011.

[11] M. Elmasry. Computer vision for lane finding. *http://www.moatazelmasry.com/projects/computer-vision-lane-finding/*, 27.5.2017.

[12] N. Falaleev. Bird's eye view transformation. *https://nikolasent.github.io/opencv/2017/05/07/Bird's-Eye-View-Transformation.html*.

[13] M. Ftáčnik. Segmentácia obrazu, predmet: Základy počítačovej grafiky a spracovania obrazu. *http://sccg.sk/~ftacnik/IP-5.pdf*, 2019.

[14] R. Hubinský. Asistenčné systémy vozidiel, 1.časť. *PC REVUE 12/2018*, 28.12.2018.

[15] ionroad. ionroad augmented driving pro. *https://ionroad-pro.en.aptoide.com/app*, 2013.

[16] G. Mrva. Rozšírenie riadenia vozidla počítačovým videním : Bakalárska práca, univerzita komenského v bratislave fakulta matematiky, fyziky a informatiky. 2015.

[17] Python official. Comparing python to other languages. *https://www.python.org/doc/essays/comparisons/*.

[18] Opencv. Camera calibration. *https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html*.

[19] Opencv. Image thresholding. *https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html*.

[20] Opencv. Introduction to opencv-python tutorials. *https://docs.opencv.org/master/d0/de3/tutorial_py_intro.html*.

[21] Polícia Slovenskej Republiky. Vyhodnotenie dopravno-bezpečnostnej situácie. *http://www.minv.sk/swift_data/source/policia/dopravna_policia/dn/prezentacie_dbs/2018/Vyhodnotenie%20DBS%20za%20rok%202018%20def..pdf* ,*minv*, 19.2.2019.

[22] Ministerstvo spravodlivosti SR. Zbierka zákonov sr, zákon č. 8/2009 z. z., o cestnej premávke a o zmene a doplnení niektorých zákonov (k 1.1.2017). *Ministerstvo spravodlivosti SR*, 3.12.2008.

Príloha A: obsah elektronickej prílohy

V elektronickej prílohe priloženej k práci sa nachádza zdrojový kód programu, natáčané videá a videá s výsledkami experimentov. Zdrojový kód je zverejnený aj na stránke `https://zdarilek2.github.io/bak_praca/`.

Príloha B: Používateľská príručka

V tejto prílohe uvádzame používateľskú príručku k nášmu softvéru. Na spustenie programu je potrebné mať stiahnutý Python ideálne verzie 3.8 a vyššej. Po otvorení Pythonu so zdrojovým kódom je potrebné nastaviť hodnotu binárneho prahovania. Do premennej "ThreshBinValue"je potrebné nastaviť hodnotu z tabuľky 5.1 podľa vybraného videa. Potom môžeme kliknúť na tlačidlo Run. Samotný program sa spustí. Na úvodnej obrazovke sú úvodné informácie k ovládaniu. Kliknutím na tlačidlo "načítať video"je možné si vybrať video, na ktorom chceme spustiť detekciu. Tlačidlo "zastaviť/pokračovať"slúži na zastavenie resp. spustenie videa. Tlačidlo na klávesnici "q"slúži na vypnutie videa počas jeho behu.