VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Autonomní robot řízený mobilním telefonem s OS Android

Autonomous Robot Based on Android Phone

2016

Adam Vůjtek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Adam Vůjtek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Autonomní robot řízený mobilním telefonem s OS Android
Autonomous Robot Based on Android Phone

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvořit mobilní aplikaci pro řízení autonomního robota. Mobilní telefon, připevněný k robotickému podvozku, by měl plně nahradil řídící počítač. Aplikace by využívala zabudovaných senzorů a kamer; řízení podvozku by probíhalo pomocí Bluetooth spojení. Na základě zpracování obrazových dat a údajů ze senzorů by aplikace řídila pohyb robota a umožňovala automaticky provádět operace typu mapování prostoru, sledování objektu a jiné.

1. Rešerše řídících systémů autonomních robotů.
2. Základní ovládání a získávání dat z podvozku pomocí Bluetooth komunikace.
3. Zpracování obrazu z vestavěné kamery, detekce překážek.
4. Implementace algoritmu pro autonomní řízení na základě obrazových dat a senzorů.
5. Testování chování robota v různých podmínkách a aplikacích.

Seznam doporučené odborné literatury:

[1] Reto Meier, Professional Android 4 Application Development, Wrox, 2012, ISBN-13: 978-1118102275
[2] Sayed Hashimi, Pro Android 2, Apress, 2010, ISBN-13: 978-1430226598
[3] Howie Choset et al., Principles of Robot Motion: Theory, Algorithms, and Implementations, A Bradford Book, 2005, ISBN-13: 978-0262033275

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

Vedoucí diplomové práce: **Mgr. Ing. Michal Krumnikl, Ph.D.**

Datum zadání: 01.09.2014
Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 24. dubna 2016 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla. Hlavní díky patří Ing. Mgr. Michalovi Krumniklovi, Ph.D. za jeho pevné nervy a klidnou mysl při komunikaci se mnou.

Abstrakt

Cílem této práce bylo naprogramovat aplikaci pro systém Android, která se bude snažit autonomně ovládat automobil. K tomuto ovládání aplikace využívá komunikaci přes technologii Bluetooth a dokáže pomocí lidarového senzoru snímat a využívat detekci obrazu přes kameru zařízení. Tyto technologie by měly pomoci k lepší detekci překážek a měly by být využity k lepšímu řízení automobilu. Zaměřil jsem se na ovládání pomocí GPS bodů a aktuální pozice automobilu. Tento problém práce s body, směrem a kompasem jsem vyřešil s pomocí knihoven Android. Pomocí OpenCV detekuji v reálném čase různé druhy překážek. V reálné simulaci se však nepodařilo, aby se automobil pohyboval úplně sám. Problém nastal při ovládání volantu, jelikož volant se nedokáže vrátit na původní pozici, je velmi těžké odhadovat jízdu směru. Také jsem se setkával s velkou prodlevou GPS lokátoru v mobilním zařízením a s velmi pomalou detekcí objektů v reálném čase (někdy až 0.5 snímku za sekundu) z důvodu vysoké náročnosti na výkon mobilního zařízení. Toto testování proběhlo na mobilním zařízení Google Galaxy Nexus. Hlavním zjištěním této práce bylo, že detekce objektů v reálném čase je možná a navigování vozidla také, pokud máme dostatečně výkonný počítač a větší možnosti ovládání vozidla. Výsledky této práce umožňují pokračovat v tomto výzkumu a zlepšovat metody řízení autonomního vozidla, popřípadě využít jinou technologii než je Android pro řízení autonomních vozidel.

Klíčová slova: robot, autonomní řízení,detekce objektů, android, java, diplomová práce

Abstract

The aim of my thesis was to make an Android application which will be able to drive a car autonomously. For this purpose, the application uses a Bluetooth technology, a lidar sensor (which detects obstacles) and also an object detection in a real time. These technologies should be helpful for better detection of obstacles and should be used for better control while driving a car. I focused on controlling a car by a gps sensor and current car's position .I solved the problem with GPS points, direction and compass by Android libraries.I can detect various obstacles in real time by using OpenCV. In a real simulation the car was not able to drive itself autonomously. The problem began when manipulating the steering wheel because it could not have got back to the original position. It's very difficult to predict the car's direction. I have also found out that

the GPS locator delays in the mobile device and the detection of objects is very slow in a real time (sometimes about 0,5 frames per second) due to the high memory consumption. It was tested on the mobile Google Galaxy Nexus. The finding of my thesis is that detection of objects in a real time and navigation of a car is possible but there has to be a high-performance PC and better car driving. Nevertheless, my thesis' results enable another researches of this topic and it is possible to improve the methods of driving an autonomous car or to use other technology than Android.

Key Words: robot, autonomous driving, object detection, android, java, master thesis

Obsah

Seznam použitých zkratek a symbolů

GPS	–	Global Position system
OPENCV	–	Open Source Computer Vision Library
BSD	–	Berkeley Software Distribution
LSD	–	Line Segments Detector

Seznam obrázků

1 Úvod

V současnosti se lidé snaží co nejvíce zjednodušit si život různými roboty, ať už kuchyňskými nebo chytrými mobilními zařízeními, počítači a nebo třeba tablety. Doba už je tak daleko, že se postupně na světlo světa dostávají nejen autonomní vysavače, ale také autonomní automobily a drony. Otázka je taková, mohou být autonomní systémy natolik inteligentní, aby řídily samy sebe? Co je potřeba k tomu, a kolik času nám zabere sestrojení tak dokonalého robota, který by neohrožoval svým chodem lidstvo a byl plně autonomní. Některé firmy se tímto problémem zabývají již několik let. Jedná se kupříkladu o firmy jako Google, Nissan, Tesla, Mercedes, aj. Na těchto projektech pracují stovky odborníků a využívají nejlepší možné vybavení. Nové technologie, které se postupně zdokonalují, jsou především vytvářeny na nejlepších univerzitách světa. Na přelomu roku 2015/2016 se již poprvé setkáváme s poloautonomními automobily, které buď brázdí dálnice (zatím pouze některé společnosti zkouší chování v reálném provozu a to hlavně v USA a Japonsku), nebo přímo na prestižních výstavách ukazují automobilky ty nejluxusnější modely aut, které nejen dokážou samy zaparkovat, vyjet z parkoviště, ale také vyhnout se překážce, dodržovat vzdálenost, ale také zvládají autonomní řízení na dálnici. Mezi hlavní propagátory těchto řídících asistentů patří hlavně firmy Mercedes, Tesla, Volvo. Věřím tomu, že v následujících letech se vývoj techniky posune tak daleko, že tyto vymoženosti budou dostupné nejen v prémiových řadách, ale také u běžných aut, u kterých se dnes běžně nacházejí kupříkladu senzory vzdálenosti od překážky.

V neposlední řadě bychom neměli zapomínat, že do povědomí o plně autonomním robotu se dostalo vozítko *Curiosity*, které v roce 2012 přistálo na Marsu [1] . Toto vozítko se pohybovalo průměrnou rychlostí 30 m za hodinu a dokázalo přejet překážky vysoké až 75 cm. Časová prodleva v komunikaci mezi vozítkem a zemí byla cca 14 minut a tak řízení a získávání informací bylo velmi složité a nákladné.

Prvotní myšlenkou této diplomové práce bylo vytvořit vozítko, které bude podobné vozítku *Curiosity*. Nyní v mé diplomové práci ovládám autíčko určené pro děti, na které je připevněno Android zařízení.

V mé diplomové práci se budu zabývati autonomním robotem, který bude řízený přes chytrý telefon Android. Robot bude reprezentován jako elektrický automobil pro děti, který bude obsahovat lidarové čidlo, které by mělo zachytit překážky na určitou vzdálenost. Robot dokáže jet dopředu, dozadu, vlevo, vpravo. Mobilní telefon bude snímat kamerou dění před sebou, zachytávat přes detekční algoritmy různé objekty. Také bude obsahovat snímač polohy a kompas, který nám bude pomáhat s pohybem v prostoru. Tato mobilní aplikace by měla býti napsána v jazyce *Java*.

2 Rešerže autonomních systémů

Autonomní systémy řízení jsou v největším rozvoji. Lidé totiž touží po co největším pohodlí a po co největším komfortu, aby při jízdě automobilem mohli řešit pracovní záležitosti a přitom se nebát o svou bezpečnost. Lékařům v práci by jistě pomohli dokonalí roboti, kteří v noci budou kontrolovat pacienty a popřípadě zvládnou vyřešit lehčí situace, jako je změření tlaku nebo teploty. Letadla, která budou nejen sama dodržovat výšku letu, jak to je doposud, ale také zvládnou přistát a vznést se a hlavně kontrolovat letecký koridor a komunikovat s ostatními letadly bez pilotovy pomoci.
Autonomní technologie již nejsou hudbou budoucnosti. V dnešní době se setkáváme s těmito technologiemi téměř všude, ať už v domácnosti (autonomní vysavače, autonomní sekačky trávy), tak v automobilismu (dodržování vzdálenosti od dalšího automobilu, držení automobilu v jednom pruhu, předcházení kolizí při předjíždění, autonomní brždění, chytré převodovky), ale také v medicínském prostředí (autonomní robot, který dokáže udělat běžné vyšetření a výsledky poslat vašemu praktickému lékaři).
Traduje se, že první automatizované řídící roboty (autonomní automobily) si budete moct koupit již tento rok (2016) a okolo roku 2020 budou tyto automobily vysoce automatizované (bez potřeby zásahu člověka). Nyní se budu věnovat rešeržím společností, které se zabývají touto problematikou a vyvíjejí autonomní řídící systémy.

2.1 Google car

Vozítko společnosti Google na obrázku 1 vypadá skoro jako dětská hračka. Zdání však klame a tento prototyp zvládne velkou část řízení za řidiče. Společnost Google se začala věnovat vzniku

Obrázek 1: Google automobil [4]

autonomního automobilu již v roce 2008. S prvním automobilem najeli 700 000 mil a z toho zaznamenali dvě autonehody, které však nezpůsobil stroj, nýbrž člověk. V prvních sériích byly

automobily polo autonomní, i když sice řídily samy, uvnitř byl člověk, který dával pozor, kdyby nastala kolize nebo něco s čím by si počítač neporadil, člověk by musel tento problém vyřešit za počítač . Tyto první zkoušky s poloautonomním řízením byly zkoušeny na automobilech Toyota Prius, Audi TT a Lexus RX450h. Nicméně nyní Google vyvinul plně autonomní automobil, který dokáže jezdit bez lidské pomoci.[3]

2.1.1 Senzory

Google k řízení automobilu používá mimo jiné lidarové čidlo. Čidlo snímá okolí okolo celého automobilu a dokáže vytvořit 3D mapu prostředí a z tohoto poté pomocí algoritmu najít překážky nebo potenciální srážky s jinými vozidly. Tato mapa nejen, že obsahuje 3D body okolí, ale také vzdálenosti k objektům na mapě od automobilu.
I když lidar je skvělý co se týče 3D modelování, k určení rychlosti ostatních aut a tím pádem rozestupu [3] mezi auty se nedá použít. Proto automobil využívá radary, které jsou zapouzdřené v náraznících . Dva jsou vepředu a dva jsou vzadu. Radary posílají informace o vzdálenosti od okolních automobilů, které jsou před/za ním a automobil podle toho bezpečně přidává/brzdí. Tato technologie funguje společně s řídící jednotkou automobilu, gyroskopem, ale také otáčkami kol, aby se předešlo smykům, popřípadě kolizi z důvodu vysoké rychlosti.
Další z věcí, které automobil používá, jsou vysokofrekvenční kamery. Ty sice jsou v každém autonomním automobilu jiné, ale obecně tyto kamery jsou umístěny vedle sebe uvnitř automobilu. Tyto kamery snímají okolí vozu a pak z těchto dvou obrazů vytvoří stereo projekci. Touto technologií se docílí lepší detekce objektů a je jednodušší zjistit periferní pohyby, rozměry objektů nebo také pohyb objektu. Každá z kamer snímá v úhlu 50 stupňů s přesností na 9 metrů.
I když sonarové sondy mají dosah pouze 6 metrů, odezva tohoto systému je velmi rychlá a velmi přesná. Systémy, které jsou v automobilu umožňují vozu přebírat data ze všech řídících jednotek a následně je aplikovat na brzdový systém nebo bezpečnostní systém jako jsou pásy nebo ESP (elektronizační stabilizační systém). [3] [4]

2.1.2 Navigování

Google využívá svůj vlastní systém map, také vlastní inerciální měřící jednotky a speciální čidlo na kolech, které dokáže zjistit skutečnou rychlost vozidla a zkorigovat ji s naměřenou rychlostí. Systém pracuje společně s kamerami, které jsou na palubě a zpracovává reálná data a pomocí kterých dokáže určit přesnou polohu automobilů, které jsou okolo Google automobilu, až s přesností na několik centimetrů. Google tím dokáže zjistit s předstihem, kde se nachází problém na silnici nebo uzavřená silnice.

2.1.3 Řídící systémy

Řídící jednotky v automobilu dokážou v reálném čase modelovat dynamiku jízdy řidičů, jak se chovají chodci na přechodech, nedefinované objekty kolem nás (kužely, přenosné značky). Ně-

které informace pro řídící jednotku jsou již implicitně nastaveny, jako zastavení na přechodu nebo na červenou na semaforu, některé se učí přímo za jízdy. Automobil se učí jízdou a dalo by se říci, že čím více ujede kilometrů, tím má naučené zvládat více problémových situací a lépe si s nimi poradí.
Samoučící se algoritmus zpracovává údaje nejen když s automobilem jedete, ale také když stojíte a ostatní automobily jedou kolem vás. Dynamika a chování řidičů je mapováno a posléze tímto Google automobil předchází různým kolizím, stejně jako klasický lidský řidič. Po takových jízdách jsou automobily dostatečně chytré aby se přizpůsobily různým situacím:
Kupříkladu pomalu pohybující se automobil v pravém pruhu, naznačuje pravděpodobnost, že automobil za ním se ho pokusí předjet.
Díra nebo nějaký objekt na silnici pravděpodobnostně vykazuje, že se mu řidič bude chtít vyhnout. Pokud je zácpa v levém pruhu, je vyšší pravděpodobnost, že bude průjezdnost v pravém pruhu (pouze v USA).

[3] [4]

2.2 Delphi

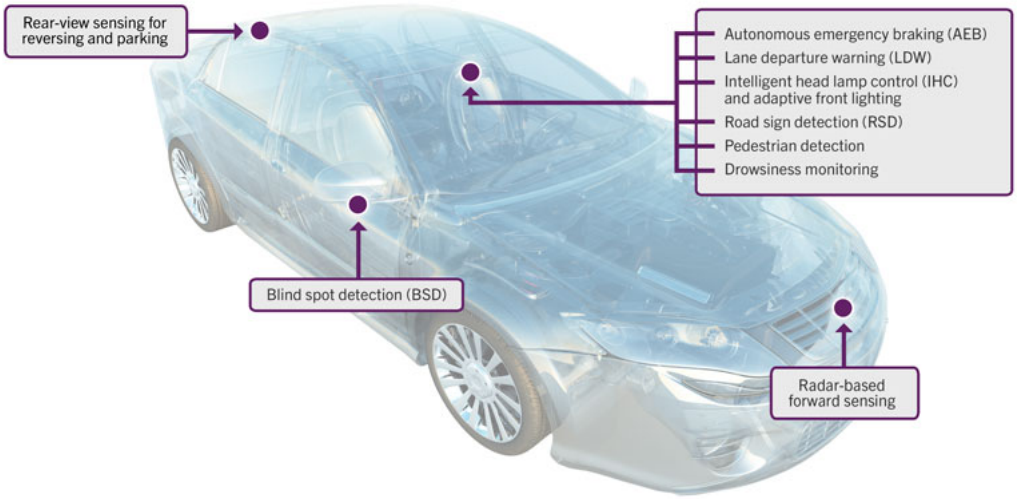
Delphi patří na trhu mezi největší firmy, které se nachází v automobilovém průmyslu. Ve vývoji autonomního automobilů se firma spojila s *Audi* a jejím modelem *Audi SQ5*. Na začátku roku 2016 tento automobil urazil vzdálenost 3400 mil z San Franciska do New Yorku a zvládl 99 % řízení sám. Pouze když bylo potřeba sjet z dálnice do městského provozu nebo naopak, zasáhl lidský faktor. *Audi SQ5* má na čelním skle kameru, která rozpoznává jízdní pruh, dopravní značky a barvu semaforu. Obsahuje také středové radary, které mají dosah až 80 metrů. Každý takový radar je umístěn v rozích automobilu. Další radar se objevuje ve přední části automobilu a dalších 6 jich je vzadu + dvě nízko rozsahové jednotky. Jedna se nachází vepředu a vzadu. Lidarové senzory se nacházejí vepředu automobilu, tam jsou umístěné tři, a další dva v zadní části zapouzdřené v nárazníku. Na obrázku 2 si můžete povšimnout, že automobil je velmi těžko rozeznatelný s normálním modelem *Audi*. Výhoda tohoto směru vývoje je v tom, že řídící systémy nejsou rozpoznatelné při běžném provozu a automobil se tváří jako klasický automobil. Všechna data jsou samozřejmě zasílána v real timu na server, kde se pak zpracovávají a zjišťuje se, jak efektivní vyhodnocení nastalo. Tímto se automobil dokáže učit ze svých vlastních chyb.

[5]

2.3 QNX

Firma QNX společně s BlackBerry se zabývá čistě adaptací svého řídícího softwaru do automobilů. V praxi to znamená, že v budoucnu, pokud budete chtít mít automobil, který bude autonomní, koupíte si packet od firmy QNX a její partneři Vám upraví Vaše vozidlo dle požadavků.[7] Na obrázku 3 jsou ukázány a vysvětleny, jaké technologie firma QNX bude nabízet.

Obrázek 2: Delphi automobil [6]



Obrázek 3: QNX technologie [7]

2.4 Volvo

Značka Volvo se již delší dobu zabývá autonomním řízením, a tak na trhu zastává přední příčky mezi výrobci autonomních systémů. Jelikož plně autonomní řízení je zatím legislativní problém téměř pro všechny země, Volvo sází na poloautomatické řídící systémy a některé systémy již přechází až v plně automatické. V systému *Pilot Assist* je kupříkladu zrychlení automobilu při detekování volné cesty. Při rychlostech do 50 km/h dokáže Volvo měnit směr jízdy, brzdit, přidávat lehce plyn a udržovat si nastavenou vzdálenost od automobilu, který je vepředu. Systém *City Safety* detekuje vozidla, které se pohybují v blízkosti vašeho vozidla. Dokáže také detekovat cyklisty a podle toho navrhnout úhybný manévr (použití *Pilot Assist*). Také dokáže rozpoznat

chodce, zvířata a jiné objekty a to jak ve tmě tak při denním světle. [8]

2.4.1 Používané technologie

Volvo jako jedna z prvních firem začala používat *chytré svícení*. Čtyři kamery na přední části dokážou na krátkou vzdálenost detekovat značení jízdních pruhů a detekci objektů. Pokud je kupříkladu vyžadována změna síly světel, ať už v tunelu nebo vozovka před vámi je prázdná, Volvo dokáže změnit tuto sílu vnějšího osvětlení.
Vozidlo má perfektní přehled o dění na silnici pomocí speciálních čtyř senzorů, které dohromady dávají přehled v úhlu 360 stupňů ze středu automobilu. Také obsahuje dvojici vysílačů směřujících dozadu, které pomáhají když automobil chce změnit směr jízdy. Tyto dva radary zjišťují, zdali není v blízkosti automobil, který by znamenal potenciální nebezpečí srážky.
Dvanáct ultrazvukových senzorů se při nízké rychlosti starají o detekci objektů na silnici. Tyto senzory jsou schopny nalézt objekt i ve tmě, a následně předat řídící jednotce informace o objektu a ta dokáže vyhodnotit, zdali hrozí srážka s objektem, a pokud hrozí srážka, jaký další krok učinit. [8]

Pomocí laserového skeneru v přední části vozidla dokáže automobil až na 150 metrů rozlišovat objekty a tím předcházet kolizím.
Volvo využívá vlastní navigační systém. Velmi detailní 3D mapy a vysoce výkonný *GPS* systém byl navržen tak, aby dokonale spolupracoval s mapami a řídící jednotkou. Systém ví, kde se přesně nacházíte a co se kolem vás děje (pomocí radaru). Také systém dokáže upravovat maximální povolenou rychlost a číst ji přímo ze značek na silnici. Také u spalovacích motorů pomocí *GPS* a map dokáže velmi efektivně řadit v závislosti na terénu a trase, kterou jsme zvolili. [8]

2.4.2 IntelliSafe autopilot

Mezi nejlepší technologie automobilky patří *IntelliSafe autopilot*. Tato technologie využívá trifokální kamery, to jsou kamery (viz. obrázek 4), které dokážou zpracovávat obraz na všechny vzdálenosti. Tyto kamery jsou umístěny v horní částí čelního skla a systém dokáže identifikovat chodce nebo jiné objekty na vozovce v reálném čase a dát signál jinému systému.

Autonomní systém od Volva je přímo spojen s cloudem, takže v reálném čase dostává informace o zácpách, nehodách nebo o dopravní situaci, přes kterou se chystá jet.Toto však ale platí i naopak, takže pokud automobil zaznamená zácpu a tato zácpa není v cloudu uložená, odešle i s přesným popisem situace. Pokud to dopravní situace bude požadovat a autonomní řízení si nebude vědět rady, řidič převezme řízení. [8]

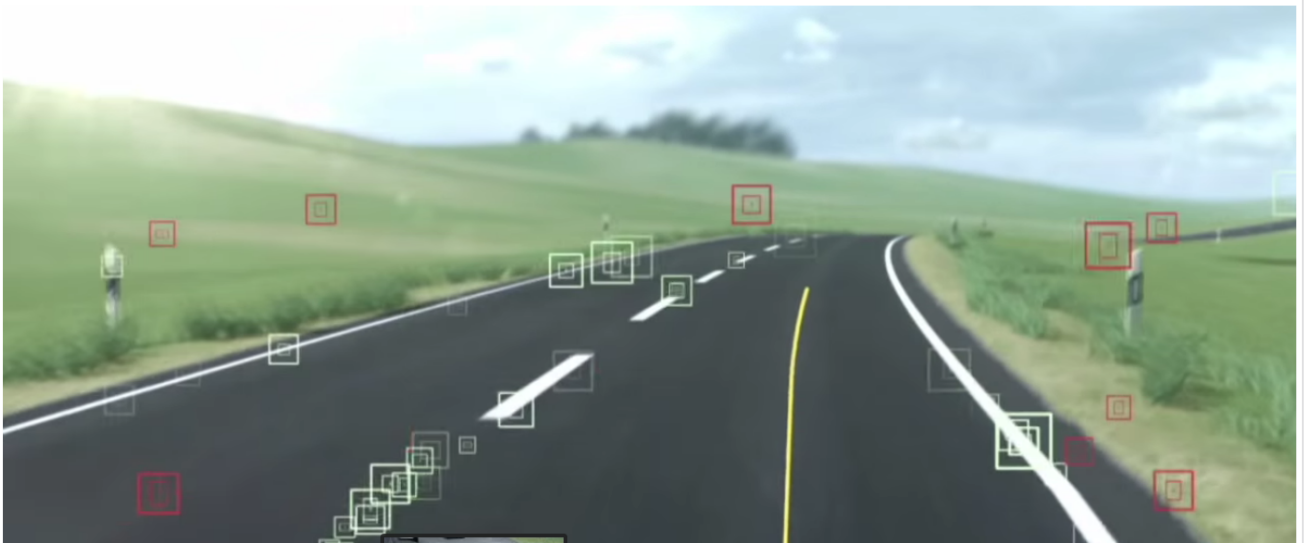
Obrázek 4: Volvo stereokamera [8]

2.5 Mercedes-Benz

Mercedes Benz patří mezi průkopníky poloautonomního/autonomního řízení svých automobilů a tuto technologii již nabízí ve svých nejluxusnějších modelech. Tento balíček se nazývá *Intelligent drive* a nabízí ho u svých čtyř tříd,C,G, E, S-class. Tyto technologie jsou velmi dobře popsány na oficiálních stránkách Mercedesu.

2.5.1 Active Lane keeping Assist

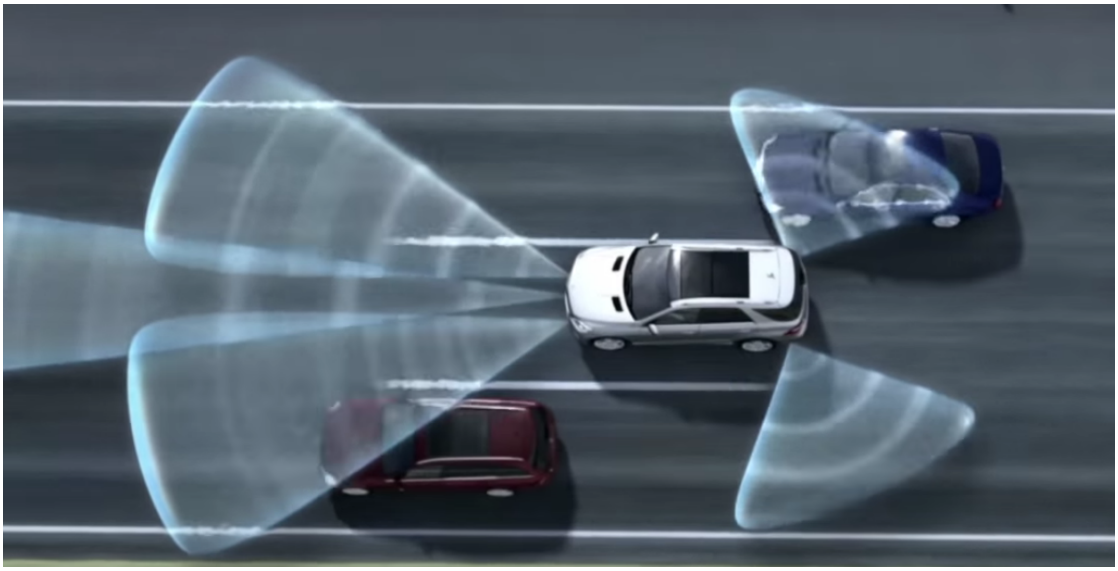
Tato technologie detekuje v reálném čase vozovku a hledá na ní bílou čáru. Poté vypočítá dokonalou stopu automobilu. Pokud se automobil blíží vnější čáře, tzn. odchylka od dokonalé stopy je vyšší než obvykle, automobil pošle signál pomocí vibrací do volantu, aby upozornil řidiče na potenciální problémy. Pokud řidič nezareaguje, automobil dokáže přibrzdit vnější kola tak, aby bezpečně automobil vrátil do vozovky. Na obrázku 5 si můžeme povšimnout, jak kamera detekuje čáry podél silnice (bílá barva). Pokud je hledání negativní, vyznačí se oblast červenou barvou (může se stát, že původně systém vyhodnotil, že na určitém místě se nachází čára, ale pak vyhodnotí, že se jednalo o negativní vyhledání). [9]



Obrázek 5: Detekce silnice pomocí stereokamery [9]

2.5.2 Active Blind Spot Assist

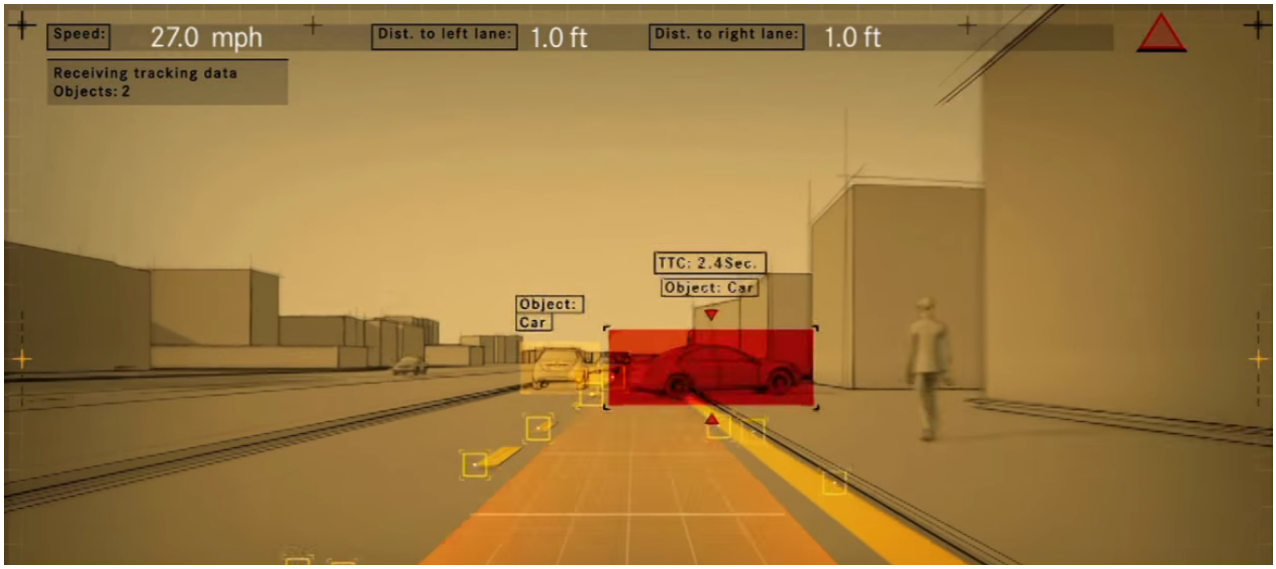
Systém radarových senzorů, které jsou umístěny v přední i zadní části automobilu kontrolují okolí, zdali se nenachází v přímé blízkosti jiné vozidlo (kupříkladu v mrtvém úhlu). Pokud chceme změnit pruh, ve kterém jedeme a v naší blízkosti se nachází vozidlo, které nevidíme a hrozila by srážka,systém dokáže přibrzdit vozidlo a tím ho dostat zpět do původního pruhu a vyhnout se tak srážce.Na obrázku 6 si můžete povšimnout, jak senzory jsou umístěny na každém rohu automobilu. Dosah senzorů zasahuje až do dalšího jízdního pruhu. [9]



Obrázek 6: Detekce okolí automobilu pomocí radaru Active Blind Spot Assist [9]

2.5.3 Bas plus

Tento systém pracuje v reálném čase pomocí tří stereo kamer, a radarů, které mají krátkodobý, střednědobý a dlouhodobý dosah.Systém dokáže identifikovat objekty, ať už automobily nebo motocykly a nebo cyklisty, a tím předejít srážce, dokáže totiž v bezpečném čase upozornit řidiče, a nebo pokud je to nutné, zastavit vozidlo. Velmi pěkně je tato technologie animována na obrázku 7 [9].



Obrázek 7: Analyzování okolí pomocí systému Bas plus [9]

2.5.4 PRE-SAFE with Pedestrian Recognition

Systém funguje na podobném principu jako Bas plus, tzn. že používá také stereokamery a radary, ale slouží primárně na chodce na vozovce.Systém snímá horizont automobilu v jeho šíři a detekuje, zdali se v tomto pásu nenachází člověk. Pokud se opravdu nachází a systém vypočítá, že by mohlo dojít ke kolizi, upozorní řidiče blikáním kontrolky a posléze umí sám zastavit. Tento systém dokáže bezpečně předejít srážce s chodcem do rychlosti 41 km/h. Pokud Však se člověk pohybuje v blízkosti jiného automobilu, kupříkladu vystupuje z automobilu a nevšiml si blížícího se vozidla, systém dokáže přibrzdit až z rychlosti 65 km/h a tím předejít nebo zmírnit srážku. [9]

2.6 Nissan

Nissan patří mezi největší automobilky, které se zabývají autonomním řízením. Do roku 2020 plánuje vyrobit deset automobilů, které budou schopny řídit autonomně. V dnešní době svou technologii zkouší na prototypu Nissan Leaf, který má elektrický pohon. Bohužel zatím i tato technologie není plně autonomní, pouze dokáže řidiči ulehčit řízení. Koncept má zabudován

velmi mnoho senzorů, dvanáct kamer, pět radarů, čtyři laserové senzory, ultrasonické lasery. Jak tyto systému fungují testuje Nissan přímo v Japonsku v městském provozu. [10] [11]

2.7 Continental

Continental patří v Evropě a obecně ve světě mezi největší dodavatele autodílů na světě. V roce 2012 firma oznámila, že v budoucnu se bude velmi intenzivně věnovat vývoji plně automatizovaného systému v automobilech. V jejím cíli je, že do tohoto roku 2016 chce vyvinout částečnou automatizaci řízení, která bude obsahovat *"stop and go"* do rychlosti 30 km/h a monitorování okolí řidiče, nicméně řidič musí být v automobilu a dávat pozor na vozovce. Do roku 2020 firma Continental plánuje vyvinout "stop and go" v režimu dálnice, tj. aby bylo vozidlo schopno zastavit při kolizi z vyšší rychlosti. Také chce zajistit, aby byl režim řízení více autonomní a řidič nemusel věnovat pozornost řízení. V roce 2025 plánuje plně automatické řízení, kdy automobil bude moct jet až rychlostí 130 km/h a to plně automatizovaně. Avšak jeden z hlavních cílů Continentalu je propojit automobily bezdrátovou sítí, kde by si automobily samy přeposílaly informace o situaci na cestách, posléze až na takovou úroveň, že by pomocí těchto informací dokázaly samy řídit bez nutnosti zásahu řidiče. Continental již spolupracuje s firmou Cisco, IBM a s jinými na V2X komunikačních systémech, které mnozí odborníci vidí jako klíč k úspěchu v autonomních vozidlech. [12]

2.7.1 Používané technologie

K testování autonomních systémů firma Continental používá 2014MY Chrysler 300. V roce 2014 firma umístila do automobilu dvě kamery, které snímají pohyby řidiče a reakce, a zpracovává je a tyto data pak pomůžou k vývoji bezpečného zástupce řidiče. Z těchto surových dat se poté udělá analýza, která se přenese do testování, kdy kamery budou snímat řidiče a pokud systém vyhodnotí, že řidič nedává pozor nebo se věnuje něčemu jinému než řízení, systém se bude snažit převzít řízení za řidiče. Automobil obsahuje přes celou šířku palubní desky počítač, který ukazuje různé informace o silnici a také v jakém režimu se řidič zrovna nachází. Pokud je zapnut adaptivní tempomat, tj. systém, který sleduje vzdálenost před vozidlem a také rychlost vozidla vepředu a podle toho umí přizpůsobit svou rychlost, buďto zvýšit rychlost nebo rychlost snížit, a také samozřejmě udržuje bezpečnou vzdálenost před vozidlem. Tento režim je indikovaný zelenou barvou. Pokud je indikace modrá, systém je přepnutý do vysoce automatizovaného režimu. Tento režim dokáže sjet a najet na dálnici, dodržovat bezpečnou vzdálenost a předjíždět. Oranžová barva je použita, pokud vozidlo řídí řidič. Continental k bezpečnému řízení vozidla využil nových HD map, které detailně reprezentují silniční prostředí. Tyto mapy obsahují informace typu kolik má silnice pruhů, v jakých vzdálenostech tyto pruhy jsou, jaké je zakřivení silnice, detailní umístění povolených sjezdů z dálnice. Tento mapový soubor je samozřejmě doplňován a sdílen přes bezdrátovou síť s jinými auty, aby se tím dokonaleji doplnil o různé výjimky, které mohou nastat na silnici. Automobil také využívá řadu snímacích technologií. Firma vsadila na radar

s dlouhým dosahem, a to 250 metrů s radiusem 120 stupňů. Tento radar se nachází uprostřed vozidla. Další čtyři radary krátkého dosahu jsou umístěny na každé straně automobilu. Tyto radary dosahují délky 90 m a radiusu 120 stupňů. Vozidlo také obsahuje lidarové čidlo, které rotuje na střeše a tím detekuje okolí okolo sama sebe. V čelním skle můžeme nalézt stereo kameru, které slouží k detekci dráhy vozidla a také k rozpoznání různých objektů na silnici, ať už automobilů nebo chodců nebo dopravního značení. Tato stereokamera dokáže také rozpoznat zdali protijedoucí vozidlo má zapnutá dálková světla. Stereo kamera využívá k rozpoznávání jízdního pruhu dva algoritmy a výsledky, které obdrží, analyzuje , odstraní redudantní data a složí je do jednoho správného výstupu. Každý z těchto systémů je napojen na jiný zdroj napájení. Tím by se mělo zabránit výpadku více bezpečnostních systémů najednou. Nicméně systém jako takový se zatím testuje aby mohl jít vůbec do aktivního testování na silnici v reálném režimu. [12]

2.8 Tesla

V roce 2014 Tesla představila své plně elektrické auto s mnoha senzory, které se nacházely okolo celého automobilu. Za příplatek si mohl člověk pořídit technologický balíček, který obsahoval speciální senzory, stereo kameru, přední radary, digitálně řízené brzdy. Touto metodou firma sbírala data o řidičích a stylu jízdy a za rok zaslala aktualizaci softwaru šedesáti tisícům automobilů, které si tento balíček připlatili. Software nesl jméno Tesla 7.0 autopilot. Řídící pult automobilu velmi připomíná velký tablet, viz. 8. [13]

Obrázek 8: Řídící pult automobilu Tesla [14]

Po tomto softwarovém balíčku vůz dokázal měnit rychlost v závislosti dopravních předpisech, měnit jízdní pruhy, držet si odstup a detekovat jiné vozidla v slepém úhlu. Vůz nelze při startování ihned pustit v režimu autopilot. Je to zapříčiněno tím, že Tesla software musí mít o místě kde se nacházíte dost dat, aby dokázala bezpečně řídit. Pokud automobil má tyto data a navíc kamery dokážou rozpoznat silniční pruhy a automobil se pohybuje konstantní rychlostí, pak je možné zapnout autopilota. Tesla také využívá ultrazvukové senzory k identifikaci objektů. Tyto vysokofrekvenční vlny dokážou detekovat objekt na vzdálenost 4.8 metru. Tento senzor se používá při zácpě na dálnici nebo na silnici, kdy se lidé snaží přejíždět z jednoho pruhu do druhého a hrozí srážka, kterou může zapříčinit nepozornost nebo špatný odhad vzdálenosti. Tímto automobil dokáže dokonaleji zrychlovat a zpomalovat v závislosti na tomto senzoru. [13]

2.8.1 Použité technologie

V přední části automobilu se nachází stereokamera a také radarová jednotka na detekování překážek. Vše spolupracuje s ultrazvukovými senzory, které jsou v každém rohu automobilu a mají dosah 4,8 metru a tím detekují objekty, které se pohybují kolem automobilu. Na displeji vevnitř se zobrazují různé překážky , třeba obrubníky. Display také ukazuje čáry kolem automobilu, které obsahují bezpečný prostor okolo. Displej nejen, že ukazuje detekci okolí (obrázek 9), ale také i spotřebu, aktuální rychlost a nejvyšší povolenou rychlost. [13]

Obrázek 9: Detekce vozovky pomocí high autonomous system firmy Tesla [15]

I když vám automobil hlásí, že máte své ruce udržovat stále na volantu, jde o čistě zákonnou povinnost, na řízení to nemá žádný vliv. Na dálnici tyto senzory fungují velmi dobře, bohužel však, pokud je čára tečkovaná nebo má jinou barvu, Tesla pak má problém vyhodnotit oč se jedná.

3 Teorie autonomního řízení

V nejjednodušší představě jak řídit robota si můžeme představit šachovnici, po které se robot může pohybovat. V určitém bodě se nachází cílový bod. Na jiných místech se mohou nacházet překážky. Robot by měl dojít k bodu tak, aby nenastala kolize s překážkami. Robot by se měl přibližovat v směrnici síly F, která směřuje na koncový bod. Na robota by také měla působit opačná síla, nazývejme ji zjednodušeně K, která by měla robota směrovat pryč od překážek. Zjednodušeně řečeno, součin těchto sil by měl zapříčinit, aby se robot vyhnul všem překážkám. Tuto myšlenku algoritmu využívám ve své praktické části. Robot je vždy nasměrován na koncový bod. Mezi robotem a koncovým bodem jsou mezi body, které zjednodušují pohyb robotovi. Pokud robot narazí na překážku, robot se vrátí opačným směrem než k překážce došel a poté se vydá ve směrnici na další mezi bod, tak aby se vyhl nalezené překážce. [16]

3.0.1 Potenciální pole a směrová funkce

Robot je prezentován jako částice, který je závislá na potenciálu pole U definovaného jako

$$U = U_{att} + U_{rep} \tag{1}$$

kde U_{att}, U_{rep} jsou atraktivní a odporové (překážkové) potenciály (tj. všechny cesty).
Atraktivní potenciál má tendenci robota směřovat směrem k cílové pozici, kdežto odporový potenciál má tendenci vyhýbat se překážkám na trase. [16] Vektorové pole umělé síly $F(q)$ je definováno gradientem U

$$F(q) = -\Delta U_{att} + \Delta U_{rep} \tag{2}$$

kde ΔU je gradientem vektoru U na pozici $q(x, y)$ v 2D mapě , na které se nachází robot. V tomto případě je F definované jako [16] suma dvou vektorů $F_{att}(q) = -\Delta U_{att}$ a $F_{rep}(q) = \Delta U_{rep}$ kde konečná funkce má tvar:

$$F(q) = F_{att}(q) + F_{rep}(q). \tag{3}$$

Obecná forma rovnice, která popisuje možné atraktivní cesty polem byla popsána a navržena [16] takto:

$$U_{att} = \frac{1}{2}\tau d^2, \tag{4}$$

kde $d = |q - q_a|$, q je aktuální pozice robota, q_a je pozice atraktivního bodu (tj. bod, který se blíží k cílovému bodu), a τ je libovolná konstanta. Obecná forma rovnice, která popisuje možné odporové (překážkové) cesty (tj. cesty, jak se vyhnout překážkám) byla navržena jako:

$$U_{rep} = \begin{cases} \frac{1}{2}\eta(\frac{1}{d} - \frac{1}{d_0})^2, & \text{if } d \leq d_0, \\ 0, & \text{if } \ d > d_0, \end{cases} \tag{5}$$

kde $d = |q - q_0|$, q je aktuální pozice robota, q_0 je bod překážky, d_0 je vzdálenostní vliv a η je vhodná konstanta.
Odpovídající vyjádření síly přitažlivé (atraktivní) síly F_{att} je tedy:

$$F_{att}(q) = -\Delta U_{att} = -\tau(q - q_0), \tag{6}$$

kde q je aktuální pozice robota a q_0 je pozice atraktivního bodu a η je volitelná konstanta.
Odporová (překážková) síla F_{rep} je dána jako:

$$F_{rep}(q) = \Delta U_{rep} = \begin{cases} \eta(\frac{1}{d} - \frac{1}{d_0})\frac{(q-q_0)}{d^3}, & \text{if } d \leq d_0, \\ 0, & \text{if } \ d > d_0, \end{cases} \tag{7}$$

kde q je pozice robota, q_0 je bod překážky, $d = |q - q_0|$, d_0 je vzdálenostní vliv a η je volitelná konstanta.
Cílem úpravy rovnic bylo zlepšení výkonu vyhledávání. Nicméně následující síla F_{att} slouží k nalezení konečného a čtyř pomocných atraktivních bodů, tj. atraktivní body, které jsou nejvhodněji zvoleny vůči konečnému bodu. [16]

$$F_{att}(q) = -\Delta U_{att} = -\eta(q - q_a)\frac{1}{|q - q_a|}, \tag{8}$$

kde q je aktuální pozice robota, q_a je pozice atraktivního bodu, η je kladná hodnota dána z genetického algoritmu.
[16]Původní síla F_{att} (6) byla normalizovaná jako síla, která je nezávislá na vzdálenosti mezi robotem a cílovým bodem $F_{att}(8)$. Umělá odporová síla F_{rep} je definována jako :

$$F_{rep}(q) = \Delta U_{rep} = \begin{cases} \eta\sqrt{(\frac{1}{d} - \frac{1}{d_0})}\frac{(q-q_0)}{d^3}, & \text{if } d \leq d_0, \\ 0, & \text{if } \ d > d_0, \end{cases} \tag{9}$$

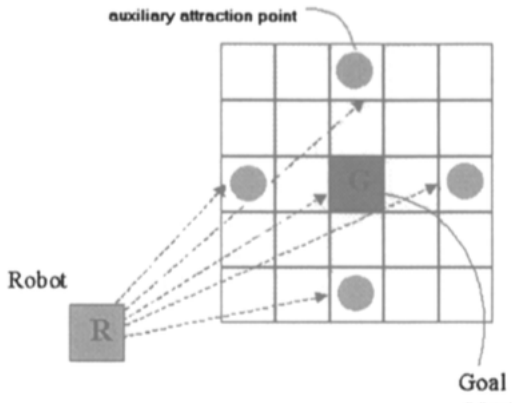
kde d_0 je vzdálenostní vliv, η je kladná hodnota dána z genetického algoritmu, $d = |q - q_0|$, q je pozice robota a q_0 je bod překážky .
Pokud se robot dostane blíže k překážce, odporová síla F_{rep} roste v opačném směru, než je trajektorie robota (tj.robot totiž směřuje k atraktivnímu bodu, ale jelikož tam je překážka, tak musí jít v jiném úhlu a k tomuto mu pomůže F_{rep}, která působí v opačném směru). Pokud je vzdálenost robota od překážky velká, tato překážka a síla nemá žádný vliv na chování robota. Vyšší odporová síla je důležitá k průchodu robota v úzkých uličkách, tj. směrnice vektoru bude strmější. Nicméně z výzkumu [16] bylo vypozorováno, že pokud pokud zvolíme $\sqrt{\frac{1}{d} - \frac{1}{d_0}}$, můžeme si všimnout mírné zvýšení odporové síly F_{rep} na střední vzdálenost. Toto nám umožňuje s předstihem se vyhýbat překážkám. [16]

3.1 Genetické algoritmy pro navigování robota

Je velmi těžké optimalizovat potenciály pole k navigování robotů, ať už kvůli velkému počtu proměnných a také složitosti výpočtu. Proto se v této oblasti využívají Genetické algoritmy. Genetické algoritmy dokážou optimalizovat velké, často složené, komplexní funkce. Dokážou postupně nacházet lepší cesty a tím se vyhnout nákladnému přístupu do paměti, které je náročné na běh počítače. [16]

3.1.1 Navigování

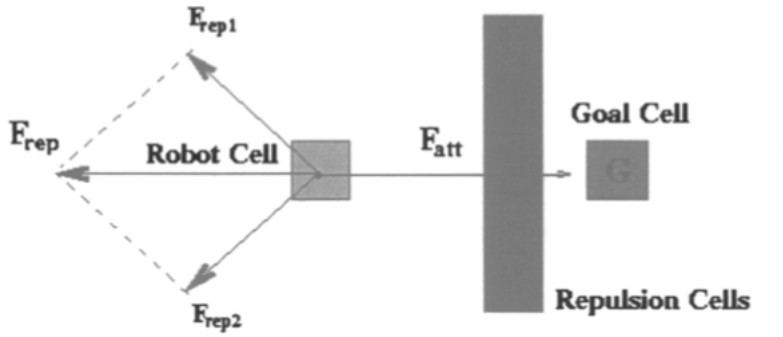
Robot je reprezentován jako částice R, která se pohybuje v dvoudimenzionálním prostoru C. Každá buňka může být obsazena robotem, nebo cílovým bodem a nebo překážkou. Ve stejné velikosti jako prostor C je vytvořena mapa překážek M. Tato mapa je zprvu prázdná a je postupně naplňována v závislosti na nalezení robotem. Cílový bod a čtyři pomocné body jsou určeny pomocí rovnice (8), a každá ze zjištěných překážek mění odporovou sílu F_{rep} (9). Tyto pomocné body jsou rozmístěny okolo cílového bodu a pomáhají k nalezení cílové cesty. Ilustrace tohoto problému je zobrazena na obrázku 10 . [16]
Pokud robot svými senzorem (senzory si musíme představit na každé buňce okolo robota) narazí na překážku, nastaví vzdálenost na $d_{min} = 1$. Pokud však nedetekoval žádnou překážku, nastaví $d_{min} = Size(R)$. Pokud tedy se pohybujeme na poli o velikosti 5×5 , pak toto číslo bude 5. Vektor síly ukazuje směrem k atraktivním bodům a robot by se měl stáčet tímto směrem. Pokud však narazí na překážku, směr by se měl změnit v závislosti na silách F_{rep} a F_{att}. (Obrázek 12). [16]



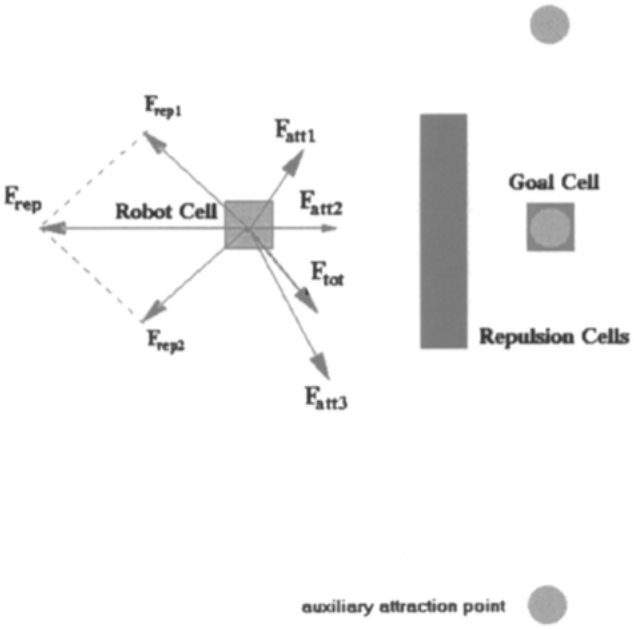
Obrázek 10: Rozmístění atraktivních bodů, cílového bodu a sil F_{att} [16]

3.1.2 Cílová funkce pro navigování robota

Cílová funkce byla vytvořena tak, aby výsledek síly pole byl plně konfigurovatelný a reagující na optimální polohu robota (takovou pozici, aby byl co nejblíže konečnému bodu a také co



Obrázek 11: Velkou překážku nelze překonat pouze s jedním atraktivním bodem [16]



Obrázek 12: Při využití vícero atraktivních bodů výsledná síla dokáže navádět robota okolo překážky [16]

nejdále od překážky). Cílová funkce je výslednicí každého uspořádání silového pole u kterého se hodnotí dvě kritéria. Minimalizace chyby na vzdálenost E mezi robotem a cílovou buňkou. A také maximalizace vzdálenosti d_{min}, tj. dosáhnutí co největší vzdálenosti od překážky. Následující rovnice ukazuje optimalizaci pro minimální vzdálenost E a maximální d_{min}. [16]

$$f = \begin{cases} \sqrt{\frac{E}{2}} e^{-d_{min}}, & \text{if } d_{min} > 0, \\ 2000, & \text{if } d_{min} = 0, \end{cases} \tag{10}$$

kde d_{min} je vzdálenost k nejbližší překážce, $E = |q_r - q_p|$, q_r je kandidát na novou pozici robota, q_g je cílový bod.

3.1.3 Genetický algoritmus

Genetické algoritmy jsou velmi účinná technika pro optimalizaci složité funkce ve velkém datové oblasti (prostoru vyhledávání). Výhodou Genetického algoritmu je poměrná jednoduchost. GA se řídí vývojem jako můžeme nalézt v přírodě. Tam totiž existují populace jednotlivých druhů, které jsou složené z různých jedinců a různých vlastností. Pro jedince se používá označení fenotyp a pro jeho reprezentaci se používá termín genotyp, genom nebo chromozom [17]
V našem případě je algoritmus použit z důvodu zjednodušení operací, protože pomocí GA se přesnost zlepšuje každou iterací cyklu. Takže za určitou dobu je schopen projít stejnou mapu nejrychlejší možnou metodou. [16]

4 Ovládání automobilu

Ovládání automobilu je řízeno přes Bluetooth spojení mezi řídící jednotkou automobilu a mobilním zařízením. Mobilní zařízení zasílá určité znaky a řídící jednotka automobilu tyto znaky detekuje a poté je vyhodnotí a následně provede úkon dle příchozího znaku.

4.1 Řízení motorků pohybu v automobilu

Řízení všech pohybů je umožněno smyčkou, která běží po celou dobu na řídící jednotce automobilu. Automobil obsahuje 3 motorky. Jeden motorek pohybuje levým kolem, druhý motorek kolem pravým. Třetí motorek zatáčí volantem. Vozidlo má klasickou koncepci a tedy je poháněna zadní náprava.

Obrázek 13: Reálný vzhled automobilu

4.1.1 Pohyb do stran

Automobil nemá servo motorek, pro otáčení řídící tyče používá stejnosměrný motor, který je nevhodný pro robotické řízení, protože neumožňuje přesné nastavení polohy volantu. Vhodnější by bylo krokový motorek, který se stará o otáčení volantu.
Otáčky volantu jsou volně nastavitelné. Motorek však jak jsem již vysvětlil, umí otočit volantem vpravo nebo vlevo. Neumí volant vrátit do původního stavu. Vozidlo má dvě polohy otáčení volantu na každou ze stran. Jedna poloha je otočení volantu v celém rozsahu a druhá pozice je otočení volantu pouze do poloviny možného otočení.
Velikost otáčení volantu nastavuje funkce $delay(number)$, kde $number$ znamená, jak dlouho mo-

torek má jet v ms. Toto absence návratu volantu do původní hodnoty výrazně zhoršuje ovládání vozidla. Vozidlo aby dokázalo jet rovně, musí v pravidelném intervalu měnit stranu natočení volantu.

```
if (incomingByte == 'd') {
    setMotor(3,'L',150);
    delay(400);
    closeMotor(3,'L');
}

if (incomingByte == 'D') {
    setMotor(3,'L',150);
    delay(100);
    closeMotor(3,'L');
}
```

Výpis 1: Ukázka ovládání zatáčení

4.1.2 Pohyb dopředu a dozadu

Při poslání znaku w , automobil zrychluje, tj. přidává rychlost vždy o deset. Zastavení vozidla je pomocí znaku s. Pro couvání se používá znak x, platí stejné pravidla jako pro jízdu dopředu.

```
if ( Serial . available () > 0) {
        // read the incoming byte:
        incomingByte = Serial.read();
        if (incomingByte == 'w') speed+=10;
        if (incomingByte == 's') {
            closeMotor(1,'R');
            closeMotor(1,'L');
            closeMotor(2,'R');
            closeMotor(2,'L');
            speed = 0;
            return;
        }
if (incomingByte == 'x') speed-=10;
        }
```

Výpis 2: Pohyb dopředu a dozadu

4.2 Komunikace mezi Android zařízením a automobilem

Tato komunikace probíhá přes *Bluetooth* standart. Při startu aplikace se mobilní zařízení nejdříve snaží připojit na adresu automobilu a poté se vytvoří vstupní a výstupní *stream*, přes který budou chodit data mezi automobilem a Android zařízením.

```
        Log.d(TAG, "onActivityResult␣" );

        String address =BluetoothAddress;
          // Get the BluetoothDevice object
          adapter=BluetoothAdapter.getDefaultAdapter();
          BluetoothDevice device = adapter.getRemoteDevice(address);

          try {
              socket = device.createRfcommSocketToServiceRecord(UUID
                      .fromString("00001101-0000-1000-8000-00805F9B34FB"));
              Log.d(TAG, "Socket␣before␣connect");
              socket.connect();
              Log.d(TAG, "Socket␣connected");
              ins = socket.getInputStream();
              ons = socket.getOutputStream();

              osw = new OutputStreamWriter(ons);
              isr = new InputStreamReader(ins);

          } catch (IOException e) {
              // TODO Auto-generated catch block
              e.printStackTrace();

          }.
```

Výpis 3: Komunikace mezi Android zařízením a automobilem

Poté pomocí metody *sendBT(char z)* můžeme posílat příkazy automobilu. Pokud vše dopadlo úspěšně, uložíme veškeré informace o komunikaci mezi zařízeními do proměnné *device*.

```
    BroadcastReceiver receiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            Log.d(TAG, "onReceive");
            String accion = intent.getAction();

            if (BluetoothDevice.ACTION_FOUND.equals(accion)) {
                BluetoothDevice device = intent
                        .getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                Log.d(TAG, device.getName() + "␣" + device.getAddress());

            } else if (BluetoothAdapter.ACTION_DISCOVERY_STARTED.equals(accion)) {
                Log.d(TAG, "ACTION_DISCOVERY_STARTED");
```

```
        } else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED
                .equals(accion)) {
            Log.d(TAG, "ACTION_DISCOVERY_FINISHED");
        }

    }
};
.
```

Výpis 4: Zapnutí Bluetooth komunikace

5 Zpracování obrazu v reálném čase

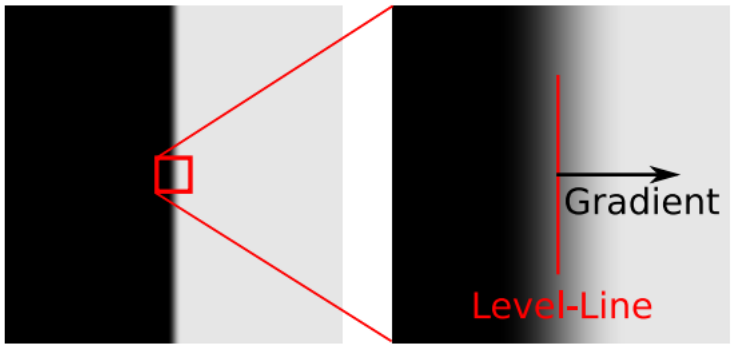
Nejlepší metodou jak rozpoznávat překážky je pomocí senzorů ale také pomocí zpracování obrazu. Pomocí zpracování obrazu dokážeme nalézt pohybující se objekty, nebo vyhodnotit objekt a rozpoznat kupříkladu auto nebo člověka. Klasickou úlohou v rozpoznávání objektů je detekce obličeje. Tato metoda funguje téměř skvěle a v reálném čase dokáže rozpoznávat s vysokou rychlostí. Při detekci pohybujících se objektů nastává problém, jelikož pokud je robot v pohybu, zdá se mu být v pohybu i vše ostatní, i když se třeba jedná o stacionární objekty.

5.1 OpenCV

OpenCV je knihovna, která byla vytvořena pro efektivní práci v náročných úkolech, jako jsou detekce objektů v reálném čase, práce s videem, práce s obrázky a nebo robotiky. Knihovna OpenCV je šířena pod licencí BSD, tudíž není žádný problém ji používat jak v akademickém tak komerčním provozu. Většina knihovny je napsána v jazyce *C++/C*. Nicméně OpenCV je podporován i v ostatních jazycích jako *Java*,*Python*, aj. A je samozřejmě nezávislá na distribuci OS. Knihovna podporuje *multi-core* a také hardwarovou akceleraci.

5.2 Line Segment Detector

Detekce hran je dobrá, pokud chceme najít, kde končí objekt a začíná jiný objekt. Popřípadě, pokud se objekt skládá z vícero částí, jak jsou tyto části strukturované. Pokud chceme detekovat hrany metoda *LSD* patří mezi jednu z nejlepších. LSD detekuje rovné kontury na obrázku. V tomto případě jsou kontury myšleny jako části obrazu, kde stupeň šedé se mění dostatečně rychle, tzn. měnící se ze světlé na tmavou nebo naopak. V reálu hledáme gradient funkce a hustotu obrysu hran. [18] [19]

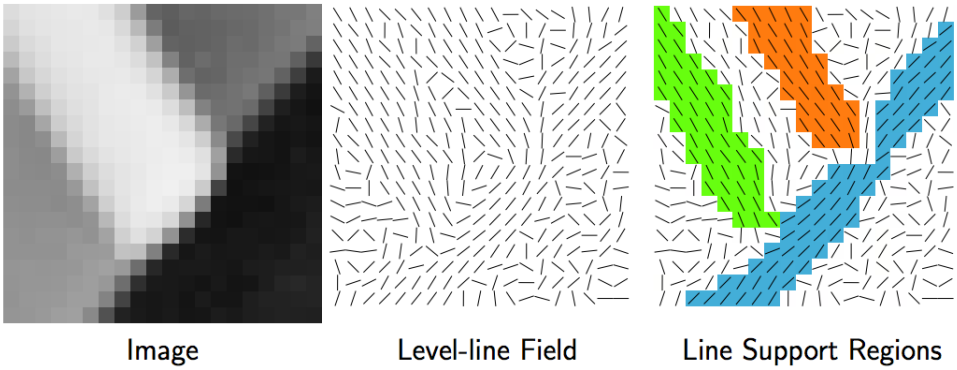


[19]

Obrázek 14: Hledání přechodů hran

Algoritmus začíná výpočtem úhlu vůči každému pixelu a následně se tyto hodnoty uloží do pole. To znamená, že jednotkový vektor je určen tak, aby všechny ostatní vektory byly tečnou k

vrstevnici, která prochází hlavním bodem. Toto pole rozdělíme do spojených oblastí obrazových bodů, které mají podobné velikosti úhlu, kde tolerance je dána τ. Tyto spojené části se nazývají podpůrné regiony. [19]



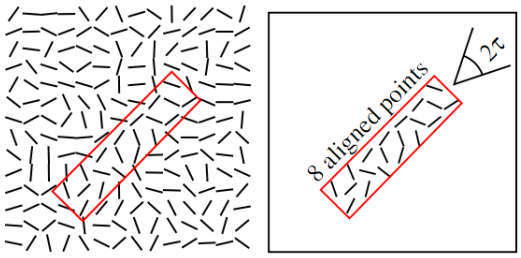
Obrázek 15: Hledání přechodů hran [19]

Každá oblast podpůrných regionů (souboru pixelů) je brána jak potenciální hrana obrázku. Poté si zvolíme odpovídající geometrický objekt, který by měl co nejvíce odpovídat tomuto souboru pixelů. Hlavní osa setrvačnosti [19] podpůrného regionu je použita pro směr geometrického objektu (kupříkladu obdélníku). Velikost obdélníku musí pokrývat celou oblast. Každý obdélník

[19]

Obrázek 16: Zaplnění podpůrného regionu obdélníkem [19]

je následně testován. Pixely v obdélníku jejichž hranový úhel odpovídá úhlu obdélníku v toleranci τ jsou nazývány zarovnanými body. Celkový počet pixelů v čtverci a počet bodů, které



[19]

Obrázek 17: Zarovnané body [19]

patří k zarovnaným bodů jsou nakonec porovnány a zjišťuje se, zdali se jedná o hranu obrázku nebo nikoliv. V příkladu 5.2.1 se můžete podívat, jak nalézt co nejjednodušeji hrany objektů přímo přes metody knihovny *OpenCV*.

5.2.1 Algoritmus výpočtu LSD

Je velmi jednoduché použít metody, které jsou již implementovány na použití v reálném prostředí. Stačí použít detektor hran od *OpenCV* a zavolat metodu *detect()*. Detektor sám nalezne hrany. Stačí využít již nadefinované metody knihovny *OpenCV*. [18]

```
public Mat LSD(Mat inputFrame)
 {
     System.gc();
     Mat mat=new Mat();
     Imgproc.cvtColor(inputFrame, mat, Imgproc.COLOR_RGB2GRAY);


       LineSegmentDetector s=Imgproc.createLineSegmentDetector();
     Log.d(this.getClass().getSimpleName(), "line segment detector");
     Mat lines=new MatOfFloat4();
     Log.d(this.getClass().getSimpleName(), "detekce");
     s.detect(mat, lines);
     Log.d(this.getClass().getSimpleName(), "konec detekce");
     s.drawSegments(mat, lines);
     Log.d(this.getClass().getSimpleName(), "vykresleni hran");

     System.gc();
     return mat;
 }
```

Výpis 5: LSD algoritmus

Obrázek 18: Ukázka algoritmu LSD [19]

5.3 Detekce pozadí

Detekce pozadí je speciální technika v zpracování obrazu, kde pohybující se obraz v popředí je získán pro další zpracování. Obecně do popředí obrazu se často dostávají automobily, lidé a nebo také čtení z výsledkové tabule (zápas v ping pongu). Po předzpracování obrazu jako je obrazový filtr náhodného šumu nebo post processing je důležitá lokalizace objektu pomocí určité metody. Odečet pozadí je klasický přístup pro detekci pohybujících se objektů na popředí z určitých video sekvencí. Odečet pozadí se vesměs využívá v případě, pokud je video natáčeno ze statických kamer . Tato metoda sama o sobě je však velmi nepřesná, jelikož v reálném prostředí mohou nastat změny pozadí a to třeba změna jasu nebo změna počasí. [21]
Robustní algoritmy by měly detekci pozadí zvládnout i se změnou osvětlení, odrážející se hladiny vody nebo nenápadné změny scény. Naše analýza bude využívat funkci $V(x, y, t)$ jako videosekvenci, kde t je časová dimenze a x,y jsou body z obraze.

5.3.1 Metody používané v detekci pozadí

5.3.1.1 Frame differencing Tento algoritmus vsází na rozdělení pohybujících se objektů na popředí a rozdělení těchto objektů od pozadí. Nejjednodušší způsob jak rozdělit pohyblivé objekty od pozadí je vzít snímky získané v čase t, označíme si $I(t)$ a porovnat s obrázkem který

byl jako předloha (byl statický, tj. bez pohybujících se objektů). Pomocí jednoduché aritmetiky, můžeme určit části objektů. Stačí pouze jednoduše použít odečítací techniku, tzn. že pro každý pixel v $I(t)$ vzít hodnotu pixelu, kterou si označíme třeba jako $P[i(t)]$ a odečteme ji se stejně pozicovaným pixelem na původním obrázku označeným jako $P[B]$. [21]

$$P[F(t)] = P[I(t)] - P[B] \tag{11}$$

Rozdílový obrázek, který nám vznikne by měl ukazovat intenzitu pro body, které se změnily. I když tímto zkoušíme odstranit pozadí, tato technika bude fungovat pouze a tehdy, pokud všechny body popředí (tj. pohyblivých objektů) jsou v pohybu a pozadí je absolutně statické. V těchto případech se používá aplikace prahové funkce (tj. *Image tresholding*), která zlepšuje odečítání obrázků.

$$|P[F(t)] - P[(F(t+1)]| > Treshold \tag{12}$$

[21] Tímto je myšleno, že rozdíl v pixelech obrázků a v jejich intenzitě je použit nějaký filtr,který zlepší výslednou hodnotu pixelu, většinou na základě hodnoty *Tresholderu*. Přesnost tohoto přístupu je závislá na rychlosti pohybu scény. Čím rychlejší pohyb, tím by měla být vyšší prahová hodnota.

5.3.1.2 Mean filter Pro zjištění pozadí obrázku se používá technika průměru všech hodnot, kde N je počet předcházejících snímků, t je v jakém čase. Toto průměrování hodnot je závislé na průměru odpovídajících pixelů v daných obrázcích. N závisí na rychlosti videa (tj. kolik je snímků za sekundu), a také na počtu hýbajících se objektů na video sekvenci. Po výpočtu pozadí $B(x, y, t)$ odečteme tuto hodnotu od obrázku $V(x, y, t)$, v čase $t = t$ a použijeme práhové čištění. Podobně lze použít medián namísto střední hodnoty $B(x, y, t)$. Pokud použijeme stále stejný *Treshhold* může to zapříčinit, že správnost výsledků nemusí být správná. [21]

$$B(x, y, t) = \frac{1}{N} \sum_{i=1}^{N} V(x, y, t - i) \tag{13}$$

$$|V(x, y, t) - B(x, y, t)| > Treshold \tag{14}$$

5.3.1.3 Gaussova pravděpodobnostní funkce Tato metoda je vždy aplikována na posledních n snímků. Prvně je vždy vypočten průměr. Gaussova pravděpodobnostní hustota každého pixelu je charakterizována jako střední hodnota μ_t a odchylka δ_t^2. Dále si můžeme nastavit počáteční stavy těchto hodnot, za předpokladu, že každý inicializovaný pixel je pozadí.

$$\mu_0 = I_0, \delta_0^2 = Prvotníhodnota, \tag{15}$$

kde I_t je hodnota intenzity pixelu v čase t. V případě inicializace rozptylu, můžeme například rozptyl v x, y souřadnici použít s pomocí sousedících bodů okolo každého pixelu. Pokud se

pozadí bude měnit v průběhu času, tj. například změnou osvětlení v detekovaném místě nebo nestatickými objekty na pozadí. Musíme zareagovat na tyto změny. Na každém snímku t , všechny střední hodnoty pixelů a jejich rozptyl musí být znova vypočten.

$$\mu_t = \varrho I_t + (1 - \varrho)\mu_{t-1}, \tag{16}$$

$$\delta_t^2 = d^2\varrho + (1 - \varrho)\delta_{t-1}^2, \tag{17}$$

$$d = |(I_t - \mu_t)|, \tag{18}$$

kde ϱ určuje velikost časového okna, které se požívá, a nastavuje se tak, aby seděl s Gaussovou pravděpodobnostní hustotou (obvykle se volí ϱ [21], d je euklidovská vzdálenost mezi střední hodnotou a hodnotou pixelu.
Nyní můžeme určit jaké pixely jsou pozadí, pokud je jeho intenzita leží v určitém intervalu spolehlivosti distribuční funkce.

$$\frac{|I_t - \mu_t|}{\delta_t} > k \rightarrow Popředí, \tag{19}$$

$$\frac{|I_t - \mu_t|}{\delta_t} \leq k \rightarrow Pozadí, \tag{20}$$

kde parametr k je stupeň práhování, a obvykle se volí $k = 2.5$. Vyšší hodnoty umožňují dynamičtější pozadí, zatímco nižší hodnoty zvyšují pravděpodobnost zjištění přechodu z pozadí do popředí v důsledku jemnějších změn.
V jedné z variant metod hledání změn v obrazu je technika, která mění distribuční funkci která je aplikována na pixely pouze tehdy, pokud jsou klasifikovány jako pozadí. Tímto se zabrání, pokud jsou nalezeny objekty na popředí a změní se jejich jas, aby se nestaly pozadím. Vzorec, který pracuje s touto metodou počítá se změnou šedi.

$$\mu_t = M\mu_{t-1} + (1 - M)(I_t\varrho + (1 - \varrho)\mu_{t-1}), \tag{21}$$

kde $M = 1$ pokud I_t je popředí, jinak $M = 0$. Když je $M = 1$, pak pokud je pixel detekován jako popředí, měl by zůstat už stále popředím. Ve výsledku, pixel, který se nastaví jako popředí, se může stát pozadím pouze tehdy, když intenzita hodnoty pixelu se přiblíží původní hodnotě, jakou měl, než se z něho stalo popředí. Nicméně tato metoda funguje pouze tehdy, když všechny pixely jsou na začátku metody nastaveny jako pozadí. Tato metoda také nepočítá s postupnou změnou pozadí. Pokud je kupříkladu pozadí postupně jinak osvětlováno, mohlo by to mít za následek chybné zpracování popředí a pozadí. [21]

5.3.1.4 Background mixture models Tato metoda vyžívá kombinaci Gaussového rozostření, kde každý pixel je podroben Gaussovému rozostření a také v reálném čase aktualizaci modelu. Při tomto postupu se předpokládá, že každý pixel hodnoty intenzity ve snímku lze modelovat pomocí Gaussového rozostření. Jednoduchá heuristika určuje, které intenzity jsou s největší pravděpodobností na pozadí.[21] Pak pixely, které neodpovídají této pravděpodobnosti jsou s jistou pravděpodobností pixely popředí. Pixely na popředí jsou spojeny pomocí 2D kompozitní analýzy.
V každém čase t, v konkrétních pixelu (x_0, y_0) je jeho historie dána

$$X_1, ..., X_t = V(x_0, y_0, i) : 1 \leq i \leq t \tag{22}$$

A historie je modelována pomocí kombinace K Gaussové distribuce

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} N(X_t | \mu_{i,t}, \sum_{i,t}), \tag{23}$$

kde

$$N(X_t | \mu_{i,t}, \sum i, t) = \frac{1}{(2\pi)^(D/2)} \frac{1}{|\sum_{i,t}|^{1/2}} exp(-\frac{1}{2}(X_t - \mu_{i,t})^T \sum_{i,t}^{-1}(X_t - \mu_{i,t})). \tag{24}$$

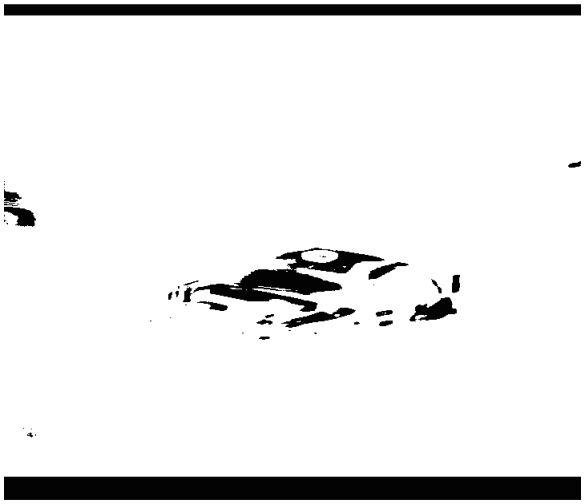
V reálném čase aproximace K průměrů je následně využita k aktualizaci Gaussového rozostření. Tato metoda byla navržena *Staufferem* a *Grimsonem*. Standardní způsob nalezení pozadí je průměrovat obrázky v reálném čase, vytvářet aproximací pozadí, která je podobná původní statické scéně ze které se čerpalo. Na obrázku 19 můžete vidět jak algoritmus pracuje.

5.3.1.5 Algoritmus detekce pozadí Algoritmus detekce pozadí Samotný algoritmus není nijak složitý, pouze si uložíme předešlý a současný frame. Poté předešlý frame odečteme od současného framu. Na výsledek použijeme *Treshold*. Poté dostaneme výsledek. Velmi jednoduché.

```
public Mat BackgroundDetection(Mat inputFrame)
{
   // https://gist.github.com/Benjit87/4245576
    System.gc();
    Mat mat=new Mat();
    Imgproc.cvtColor(inputFrame, mat, Imgproc.COLOR_RGB2GRAY);
    if( isFirstPicture )
    {
        LastPicture=new Mat();
        LastPicture=mat;
         isFirstPicture =false;
        return mat;
    }
    inputFrame.release();
    Core.absdiff(LastPicture, mat, mat);
    Imgproc.threshold(mat, mat, 15, 255, Imgproc.THRESH_BINARY);
```

```
        LastPicture=mat;
        return mat;

    }
```

Výpis 6: Detekce pozadí



Obrázek 19: Detekce automobilu v reálném čase

5.3.2 Detekce pohybu

Při detekci pohybu je podobný postup jako při detekci pozadí. Nejdříve si zvolíme obrázek se statickým pozadí jako předlohu pro následnou práci. Poté začneme snímat pomocí kamery. Algoritmus následný snímek vezme, a použije na něj Gaussovo rozostření. Poté odečte předchozí snímek od nynějšího. Na tento snímek, který vznikne odečtením, se použije adaptivní práhování (*Treshold*). Poté detekujeme kontury ve snímku. Kde byly nalezeny kontury na snímku, tam se

předpokládá, že se vyskytl pohyb. Poté už tuto oblast pouze vyznačíme.
Na obrázku 20 můžete vidět, jak v reálném čase aplikace pracuje snaží se snímat pohyb objektů.

```
public Mat MovingDetection(Mat inputFrame)
  {
    //https:// ratiler .wordpress.com/2014/09/08/detection-de-mouvement-avec-javacv/
    System.gc();
    Mat m=new Mat();

    Imgproc.cvtColor(inputFrame, m, Imgproc.COLOR_RGB2GRAY);
    if( isFirstPicture )
    {
        MovingResult=new Mat();
    }

    if( isFirstPicture )
    {
        LastPicture=new Mat();
        m.copyTo(LastPicture);
         isFirstPicture =false;

        return inputFrame;
    }

    inputFrame.copyTo(MovingResult);
    inputFrame.release();
    Mat diff=new Mat();
    Imgproc.GaussianBlur(m, diff, new Size(3, 3), 0);
    Core.subtract(LastPicture, m, diff);

    Imgproc.adaptiveThreshold( diff , diff , 255,
            Imgproc.ADAPTIVE_THRESH_MEAN_C,
            Imgproc.THRESH_BINARY_INV, 5, 2);
    ArrayList<Rect> r=new ArrayList<Rect>();

    r = detection_contours( diff );
    if (r.size() > 0) {

        Iterator <Rect> it2 = r.iterator();
        while (it2.hasNext()) {
            Rect obj = it2.next();
            Imgproc.rectangle(MovingResult, obj.br(), obj.tl(),
                    new Scalar(0, 255, 0), 1);
        }

    }
    m.copyTo(LastPicture);
    m.release();
```

```
        return MovingResult;
    }
```

Výpis 7: Detekce pohybu

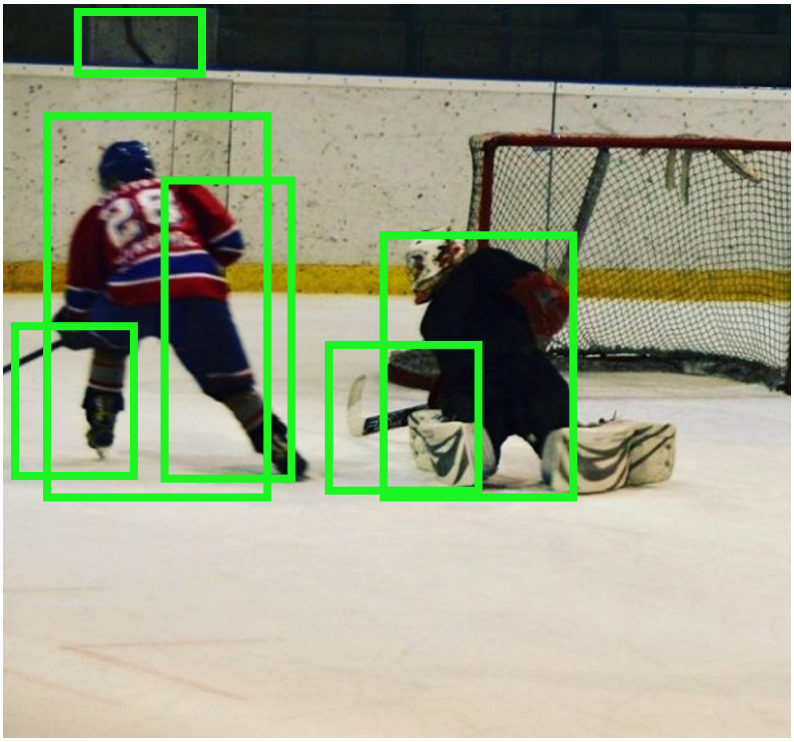
Pomocná metoda na detekování kontur

```
public  ArrayList<Rect> detection_contours(Mat outmat) {
        Mat v = new Mat();
       // Mat vv = outmat.clone();
        List<MatOfPoint> contours = new ArrayList<MatOfPoint>();
        Imgproc.findContours(outmat, contours, new Mat(), Imgproc.RETR_LIST,Imgproc.
            CHAIN_APPROX_SIMPLE);
        System.gc();
        double maxArea = 100;
        int maxAreaIdx = -1;
        Rect r = null;
        ArrayList<Rect> rect_array = new ArrayList<Rect>();

        for (int idx = 0; idx < contours.size(); idx++) {
            Mat contour = contours.get(idx);
            double contourarea = Imgproc.contourArea(contour);

            if (contourarea > maxArea) {
                // maxArea = contourarea;
                maxAreaIdx = idx;
                r = Imgproc.boundingRect(contours.get(maxAreaIdx));
                rect_array.add(r);
                Imgproc.drawContours(MovingResult, contours, maxAreaIdx, new Scalar(0,0, 255));
            }

        }
        if(contours.size()==0)
            isFirstPicture =true;

        v.release();

        return rect_array;

    }
```

Výpis 8: Pomocná metoda na detekování kontur

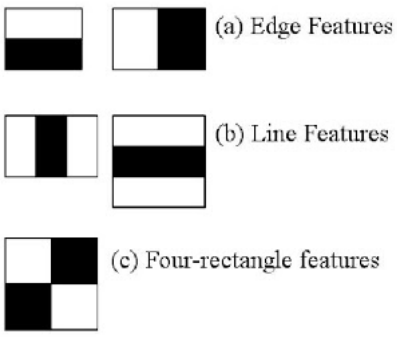
5.3.3 Detekce obličeje v reálném čase v OpenCV

Detekce obličeje pomocí Haar funkcí, které jsou založeny na kaskádovém klasifikátoru. Tato metoda byla navržena *Paulem Viola* a *Michaelem Jonesem* v roce 2001. Tato technika byla navržena na orientovaném přístupu strojového učení, kde kaskádová funkce je trénována z hodně



Obrázek 20: Detekce pohybu objektů v reálném čase

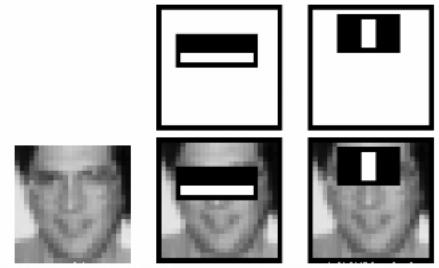
pravých a také falešných obrázků. Algoritmus se naučí, jak vypadá objekt, který obsahuje obličej a jak vypadá, pokud obličej neobsahuje. Pak tento přístup lze využít k detekci jiných obrazů. Pro detekci obličeje potřebujeme opravdu hodně snímků, ze kterých se náš algoritmus bude učit. Z těchto obrázků jsou pak následně extrahovány specifické vlastnosti. K tomuto se využívají *Haar příznaky*, které jsou uvedené na obrázku 21. Tyto příznaky můžeme také nazývat naším konvolučním jádrem. Každá funkce je jedna hodnota, která je získána odečtením součtu pixelů pod bílým obdélníkem od součtu pixelů pod černým obdélníkem Nyní všechny možné



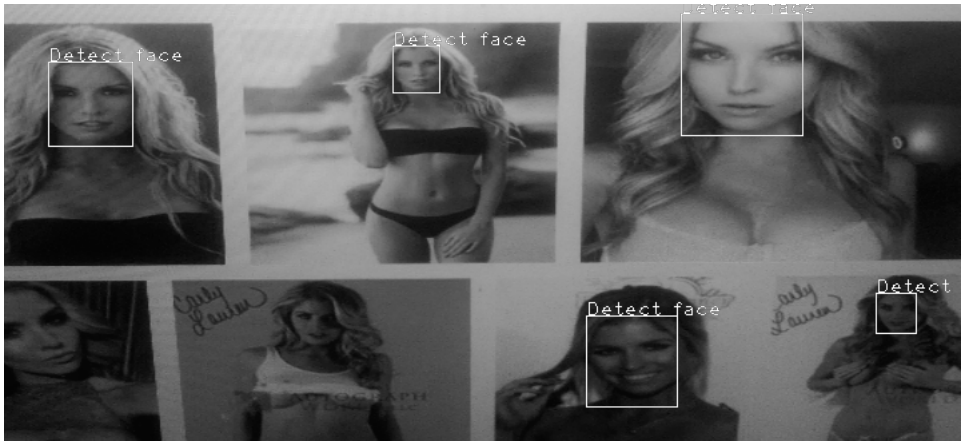
Obrázek 21: Haar příznaky [22]

velikosti a lokace každého jádra (tj. jádra obličeje) jsou počítány pomocí mnoha funkcí. (Pro

představu pro okno 24x24 pixelů je přes 16000 funkcí). Pro každý výpočet funkce, potřebujeme najít sumu bílých a černých obdélníku. Pro vyřešení tohoto problému zavedli *Jones* a *Viola* integrální obrazy. Jedná se pouze o zjednodušený výpočet součtu bodů.
Na obrázku 22 si můžete povšimnout zajímavých vlastností. První část obrázku 22 se zaměřuje na oblast očí a oblast nosu a tváře. Oblast očí je často tmavší než oblast nosu a tváře. Spodní část obrázku se zaměřuje na oblast očí a oblast kořene nosu. Oblast očí je tmavší než kořen nosu. Tyto všechny funkce jsou trénovány na mnoha desítkách tísících fotkách, kde část fotek obsahuje obličej a část neobsahuje. I tak ale nastávají chyby nebo chybové klasifikace. Proto vybíráme funkce, které dávají nejvyšší přesnost výpočtu. Tyto vybrané klasifikátory jsou znova zpracovány a pak uloženy do pomocného souboru. Tento soubor pak obsahuje kaskádové klasifikátory, tj. naučené rozměry obličejů a další důležité informace o obličeji. Z tohoto souboru se poté vyhodnocují fotky, jestli je na nich obsažen obličej.



Obrázek 22: Příznaky použité v praxi
[22]



Obrázek 23: Detekce obličeje v reálném čase

5.3.4 Detekce obličeje pomocí Android knihoven

Android knihovny mají v sobě zapouzdřenou detekci obličeje z obrázku. Postup je velmi podobný tomu v *OpenCV*, nicméně rychlost výpočtu je o poznání nižší. Při načtení obrázku se volá funkce,

která má v parametru maximální počet obličejů na obrázku. Tato podle tohoto čísla nastaví klasifikátory a funkce poté hledá po oblastech obličeje. Kupříkladu, pokud nastavíme číslo na 4, tak nejdříve bude zkoušet najít jeden obličej, pak dva, Tj. že pokusí rozdělit fotku na části a v nich hledat, pak bude zkoušet hledat tři obličeje, čtyři. Pokaždé zkouší procházet obrázek, pouze podle tohoto čísla zmenšuje oblast, kterou posunuje. Pokud nalezne obličeje, vrátí jejich souřadnice x, y.

5.3.5 Detekce automobilů

Metoda detekce automobilu je velmi podobná detekci obličeje.Námi použitou vymysleli na univerzitě v Illionois [23] Nejdříve se algoritmus učí z určitého počtu obrázků, v našem případě je to 550 obrázků, na kterých se vyskytují automobily a 500, na kterých se nevyskytují. Z tohoto vzorku algoritmus vytáhne určité důležité klasifikátory, podle kterých se dá rozpoznat automobil. Tyto klasifikátory jsou poté zapsány v souboru, tak, jak to bylo u detekce obličeje. Tento algoritmus je velmi náročný na spuštění v reálném čase, jelikož se čerpá z pěti dokumentů. Proto jsem ho odzkoušel pouze na množině obrázků. Na obrázku 24 si můžete povšimnout, že pokud je automobil detekovaný, nad jeho výběrem je i napsaná metoda (dokument), který byl použit, pokud by by nález byl pozitivní u vícero metod, zvýraznění se překreslí.

Obrázek 24: Detekce automobilu z pořízené fotografie

5.3.6 Lidar

Lidar slouží k určení přesné vzdálenosti zařízení (lidaru) od překážky. Lidar obsahuje zdroj, ze kterého bude vycházet laserové záření, nějakou optickou soustavu, mechanický prvek, detektor

elektromagnetického záření a velmi přesné hodiny. [24]
Pokud potřebujeme lidar, který bude mít velký dosah, nejspíše použijeme zdroj ve formě pevnolátkového laseru rubínového. Pokud dosah lidaru nemusí být nijak velký, stačí nám jako zdroj využít diodové lasery. Pomocí optické soustavy zajistíme aby záření probíhalo ve velmi úzkých svazcích a také by měla býti zajištěna souosost detektoru a emitoru. K těmto vlastnostem optické soustavy se využívá polopropustný hranol. Tento poslední prvek v optické soustavě ať už zrcadlo a nebo hranol, je připevněn na mechanickém prvku, který zajišťuje směrování paprsku vždy pod jiným úhlem [24]. Kvůli tomuto mechanickému prvku nemusíme otáčet celým systémem, abychom nasnímali celý prostor, stačí změnit úhel snímání.
Velmi přesné hodiny se využívají k měření času mezi vysláním svazku paprsků a jejich detekci na detektoru. Jelikož je známá rychlost světla v prostoru, z těchto údajů dokážeme zjistit vzdálenost od lidaru k objektu. Také pomocí směru vysílání a odvozené vzdálenosti můžeme zjistit polohu každého měřeného bodu. Pokud máme kontinuální laser, paprsek je frekvenčně modulován a pak vzdálenost je určena přes fázový posun [24].
V mém projektu využíváme *Hokuyo* lidar (25). Tento lidar snímá v úhlu 240 stupňů z přední části. Toto snímání je navíc rozděleno do podúhlů po 0.36 stupních. Čas potřebný ke snímání překážek je velmi malý a to 100 ms.

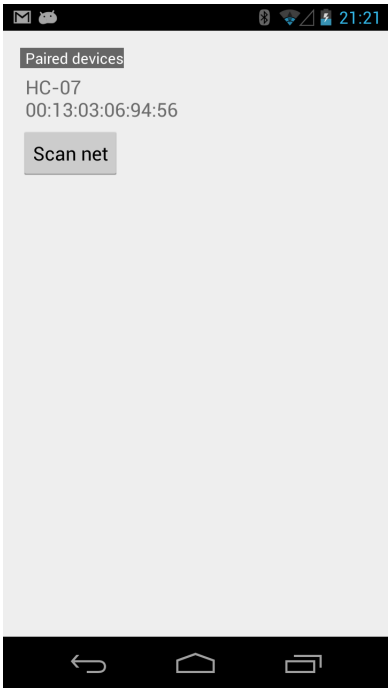
Obrázek 25: Hokuyo lidar [25]

6 Implementace algoritmu pro řízení řízení autonomního vozidla

Vyvinout aplikaci, která by byla uživatelsky jednoduchá a zároveň přehledná a plnila by všechny funkce je velmi těžké navrhnout. V nejjednodušším rozdělení bych aplikaci rozdělil na *back side* a *front side*. Tyto dvě částí jsou navzájem propojeny a *front side* čerpá data z *back side*. Je velmi důležité, aby uživatel v *front side* viděl jen důležité informace a ovládání. Ostatní věci jsou velmi nevhodné a uživatel by mohl mít poté problém s ovládáním. Nyní se budu věnovat detailněji *back side* a *front side*.

6.1 Front Side

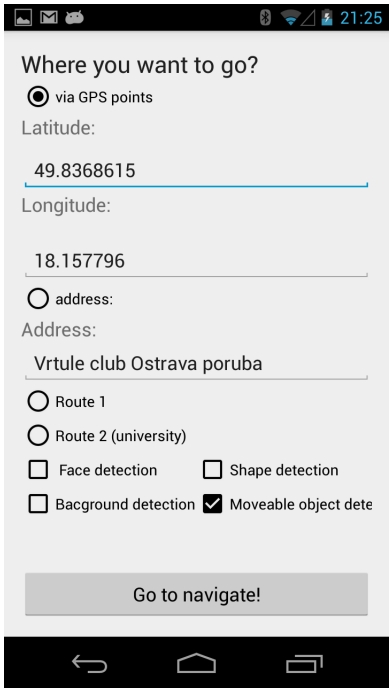
Uživatelské rozhraní se skládá ze tří aktivit. Při spuštění aplikace vám vyskočí nabídka na vyhledání *Bluetooth* zařízení, se kterým se musíte spárovat (obrázek 26). Tímto zařízením je myšlen náš automobil nebo jakékoliv jiné ovládané zařízení. Po úspěšném spárování se uživatel dostane



Obrázek 26: Výběr spárovatelného zařízení

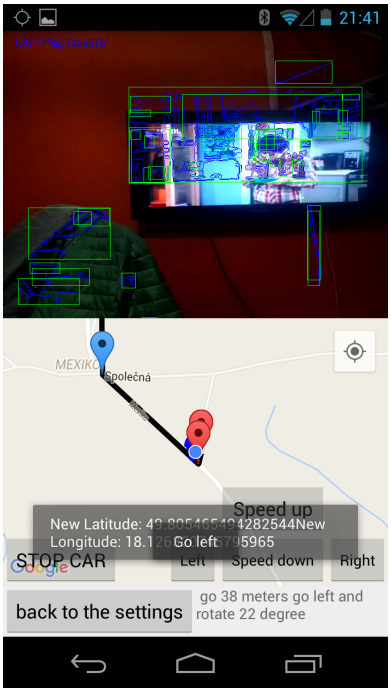
do hlavní nabídky aplikace. V této nabídce si můžeme vybrat, jestli budeme koncový bod zadávat pomocí *GPS* souřadnic, adresy a nebo využijeme na pevno zadané dvě cesty. Jedna cesta je okolo mého bydliště a druhá je okolo Fakulty elektrotechniky a informatiky.
Výběr pomocí *GPS* nebo pomocí adresy zprostředkovává přímo *Google API*, které dokáže vygenerovat trasu z počátečního a koncového bodu (adresy). Toto budu detailněji popisovat v části *Backside*. Další z věcí, které si můžeme v aplikaci vybrat je, jaký filtr využijeme. Máme možnost

detekovat za chodu aplikace obličeje nebo hrany objektů. Pokud zařízení (automobil) zastaví, a máme zaškrtnutou detekci pozadí nebo detekci pohybu v reálném čase, můžeme pomocí těchto metod zjistit, které objekty jsou v pohybu a vyhodnotit, pokud ohrožují naše vozidlo.



Obrázek 27: Opravdu máme mnoho možností nastavení aplikace

Třetí a poslední aktivita nám zobrazuje všechny potřebné informace o řízení vozidla. Aktivita je rozdělena na poloviny. První část snímá vše před mobilním zařízením a aplikuje filtry, pokud jsme je zaškrtli. Spodní část aktivity zobrazuje mapu, na které jsou body, kterýma se vozidlo má vydat. Také je zaznamenána poloha vozidla, a to modrou blikající tečkou a se šipkou, kam vozidlo směřuje. Aplikace vypočte, který bod z cesty je nejblíže pozici vozidla a označí ho červenou barvou. Vozidlo se pak snaží dostat k tomuto nejbližšímu bodu. Pokud se změní pozice vozidla, vyskočí ukazatel na obrazovku nových souřadnic a hned posléze vyskočí, jestli má vozidlo jet doprava nebo doleva. Ve spodní části jsou sděleny informace. Jaká je vzdálenost k nejbližšímu bodu, jakým směrem se má automobil otočit a o kolik stupňů se má vytočit. Pokud by nastaly nějaké problémy s řízením, ať už se jedná o kolizi nebo nesprávné vyhodnocení trasy, může uživatel ovládat vozidlo přes pomocné tlačítka. Uživatel může vozidlo zastavit, ale také mu dávat příkazy vpravo, vlevo, dozadu, dopředu. Ukazatele a typ mapy může uživatel změnit jedním kliknutím, tak jak je zvyklý z *Google maps*. Tzn. že uživatel může vidět mapy v *2D*, ale také v textit3D režimu a nebo jako letecké záběry.

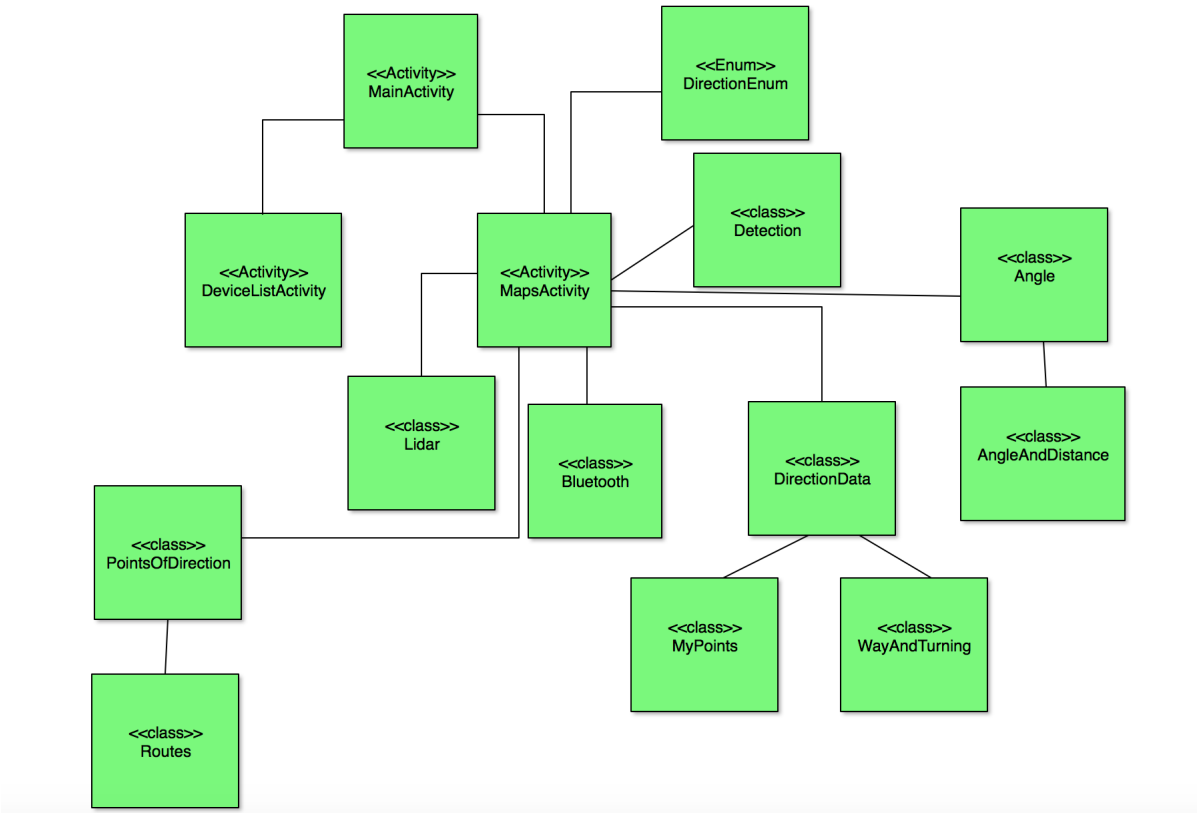


Obrázek 28: Aplikace v plném testování, hlavní aktivita aplikace

Jelikož je tato část visuální, nebylo v ní tolik naimplementováno, jako v části *backside*. V *backside* nalezneme celý mozek aplikace a různé propojení tříd a práci s *Google API*.

6.2 Back Side

Jak jsem již napsal o odstavec výše, tato část řídí celou aplikaci. Je jejím mozkem. Jedná se hlavně o logiku práce s body, vzdálenostmi, cestami mezi body. Také o využití *Google api* jako vrácení nejrychlejší cesty mezi dvěma body nebo adaptivní mapy.
Jako první krok implementace aplikace bylo naprogramování logiky tříd mezi aktivitou *MapsActivity* a třídami *Angle*, *AngleAndDistance*, *DirectionData*, *GPSTracker*, *MyPoints*, *PointsOfDirection*, *Routes* a *WayAndTurning*. Některé třídy mají relace mezi sebou. Některé třídy jsou pouze podpůrné a pomáhají vyhodnotit data jako třeba třída *Routes*, která pouze zpracovává na vstupu *JSON* data a jejím výstupem jsou seřazené body od nejbližšího po nejvzdálenější.
Poté bylo důležité řešit logiku spojení *Bluetooth* zařízení a mobilního zařízení. Tzn. jak připojit zařízení, aby bylo co nejvíce uživatelsky jednoduché. A jak poté komunikovat mezi sebou, aby nedocházelo k pádu aplikace. Tomuto problému se věnují třídy *Bluetooth* a aktivity *DeviceListActivity*, *MainActivity*, *MapsActivity*.
Jako předposlední část jsem řešil aplikaci detekce objektů v reálném čase a implementace do mé aplikace, tomuto problému se věnuje třída *Detection* a aktivita *MainActivity*. Poslední část byla lidarové čídlo, které snímá okolí automobilu. Třída *Lidar* komunikuje s aktivitou *MapsActivity*.
Nyní se budu zabývat každou částí zvlášť.



Obrázek 29: Třídní diagram

6.2.1 Angle

Třída *Angle* slouží k nalezení úhlu mezi nejbližším bodem a aktuálním bodem. Také umí vrátit vzdálenost mezi těmito body. Třída taktéž obsahuje metodu *smallestDistanceOfPoints(double lat1,double lon1,List<LatLng> l)*, která z kolekce bodů a aktuální pozice dokáže najít nejbližší bod z kolekce k aktuální pozici. Toto se hodí kupříkladu, pokud vozidlo zvolí jiný směr jízdy, ujede jistou vzdálenost a pak se chce přiblížit plánované trase. Metoda poté co nalezne nejbližší bod, přeskupí body a pokud některé body již nejsou aktuální, tj. že už jsou více vzdáleny od cíle než bod aktuální pozice, tyto body vymaže z kolekce. Při výpočtu úhlu a vzdálenosti využívá aktuální azimut a vzorce, které počítají s zakřivením země.

6.2.2 AngleAndDistance

Pomocná třída k třídě *Angle*. Konstruktor třídy uloží úhel, vzdálenost a nejbližší bod k aktuální pozici. Když robot postupuje nadefinovanou cestou, ukládá tento objekt do kolekce. Kdyby se musel vrátit, věděl by pomocí této kolekce předchozí hodnoty.

6.2.3 Detection

Tato třída slouží k detekci objektů pomocí kamery a knihovny *OpenCV*. Každá metoda ve třídě má na vstupu matici, která reprezentuje aktuálně nasnímaný obraz z kamery. Jako výstup je také matice dat, které se poté zobrazují na obrazovce. Metoda *Mat FaceDetectionAndroid(Mat inputFrame)* využívá detekci obličeje přímo zabudovanou v knihovně *Android*. Jako povinný faktor je nastavení maximálního počtu obličejů na jednom snímku. Poté snímek převedeme do stupně šedi. Poté práce je velmi jednoduchá pomocí zabudované třídy *FaceDetector*. Té se předávají inicializační hodnoty jako je velikost obrázku a maximální počet možných detekovaných obličejů. Třída poté podle vlastních pravidel hledá možné potenciální obličeje na fotografii. Pokud nastavíme kupříkladu, že na snímku mohou být pouze dva obličeje. Aplikace vypočítá jakou teoretickou velikost by měl mít obličej na fotografii a prochází pomocí naučených kritérií obrázek.
Jako další metoda je *Mat LSD(Mat inputFrame)*. Tato metoda detekuje hrany pomocí zabudovaných metod *OpenCV*.
Další metoda *Mat BackgroundDetection(Mat inputFrame)* detekuje pohyb na pozadí. Nejdříve si uložíme aktuální snímek a poté, pokud již máme uložený snímek v paměti, předchozí a aktuální snímek převedeme do stupně šedi. Poté z těchto snímků zobrazíme absolutní rozdíl a na tento výsledek použijeme prahování. Výsledek zobrazíme na obrazovce.
Mat MovingDetection(Mat inputFrame) funguje na podobném principu, ale místo hledání absolutního rozdílu snímků, tyto dva snímky od sebe odečteme. Poté se snažíme detekovat kontury vzniklého obrázku . Pokud nějaké kontury najdeme, uložíme je do paměti. Kde se nacházejí kontury, tam byl jistý pohyb a tímto jsme detekovali pohyb i souřadnice pohybu.
Metoda *Mat FaceDetection(Mat inputFrame)* používá k nalezení obličeje na snímku kaskádové klasifikátory (pravidla), které jsou přímo zapouzdřeny v knihovně *OpenCV*.
Metoda *Mat CarDetection(Mat inputFrame)* funguje téměř stejně jako detekce obličeje. Jediný rozdíl je v tom, že využívá jiné klasifikátory. Tyto klasifikátory jsou naučeny detekovat automobily.

6.2.4 DeviceListActivity

Aktivita obsluhuje a detekuje a spojení pomocí *Bluetooth* mezi mobilním zařízením a automobilem. Zobrazí všechny možné zařízení a popřípadě zobrazí ty, se kterýma jsme již spárování. Pokud párování proběhne úspěšně, přepne se do aktivity *MainActivity*.

6.2.5 GPSTracker

Speciální třída, která využívá ke své práci *LocationListener*. Klasické detekování *GPS* signálu pomocí *Android* knihoven. Nejdříve se spustí *LocationManager* z aktuální aktivity. Poté se zjišťuje, zdali *GPS* je zapnuté a jestli je dostupný provider sítě. Pokud je vše zapnuté a dostupné, zavolá

se metoda, která vynutí aktualizaci lokace. Poté pokud tato aktualizace proběhla v pořádku, zjistíme přes aktualizovaný objekt zeměpisnou šířku a délku.

6.2.6 Lidar

Třída *Lidar* obsluhuje senzor, který je umístěný na automobilu. Při volání konstruktoru předáváme adresu čidla a port. Třída dědí z třídy *Thread* , takže obsahuje metodu *run*. V této metodě nejdříve pošleme na inicializovanou adresu z konstruktoru potřebnou sérii znaků. Poté ve smyčce se snažíme číst data, které chodí z adresy. Tyto data uložíme do bufferu o velikosti 2048. Poté procházíme buffer a hledáme znak nového řádku. Pokud nalezneme tento znak. Přečteme vše co je před tímto znakem. Poté tuto sekvenci znaků čteme znak po znaku a provedeme na těchto znacích bitové operace. Poté máme nalezenou hodnotu, které vrátilo lidarové čidlo. Pokud dočteme všechny znaky před znakem nový řádek, buffer posuneme na pozici za nový řádek. Tuto operaci opakujeme dokud buffer nebude ukazovat na svůj konec. Tento postup se poté znova opakuje.

6.2.7 MainActivity

Tato aktivita se nám zobrazí hned zapnutí aplikace. Nicméně poté, zavolá aktivitu *DeviceListActivity*, kterou jsem popisoval již dřívěji. Tato aktivita slouží k čistě k vybrání metod a nastavení počátečních hodnot. Po své inicializaci zavolá *Bluetooth adapter* a zapne komunikaci přes tuto technologii. Pokud uživatel vybral a nastavil potřebné hodnoty,aktivita tyto hodnoty předá aktivitě *MapsActivity* a poté se sama ukončí.

6.2.8 MapsActivity

Hlavní aktivita, ve které dochází k celé komunikaci mezi třídami, přenášením dat z lidarového čidla nebo skrze *Bluetooth* technologii, nebo k rozpoznávání a detekci objektů v reálném čase. Pokud aktivita zaznamená první změnu *gps* souřadnic, pak nastartuje automobil. Při další změně již vyčte kde se má automobil natočit a pošle skrze *Bluetooth* tyto informace řídící jednotce autíčka. V tom stejném čase kamera snímá dění před automobilem a pomocí metody, kterou jsme si vybrali analyzuje prostředí. Lidarové čidlo snímá koridor před automobilem a pokud detekuje překážku na krátkou vzdálenost, dá automobilu pokyn zastavit. Mapa ve spodní části aktivity zobrazuje mapu okolí, body naší trasy a naši aktuální pozici a také šipku, kterým směrem směřujeme.

6.2.9 MyPoints

Pomocná třída, která pouze spravuje body na trase. Obsahuje metodu *addPoint(double lat, double lon)*, která do kolekce bodů přidá nový bod. Také obsahuje metody typu *getPoint(int id),getLast()*, které vracejí bod v prostoru.

6.2.10 PointsOfDirection

Tato třída využívá mapy od *Googlu*. Odešle speciální webovou adresu, na které zadá konečný a počáteční bod. Poté čte z této adresy. Pokud nějaká data přijdou, jsou ve formátu *JSON*. Poté tato třída předá celý stáhnutý dokument třídě *Routes*, která provede parsování a ukládání dat ze souboru do kolekce.

6.2.11 Routes

Třída je volána třídou *PointsOfDirection*. Při zavolání konstruktoru dostane na vstupu data ve formě *JSON*. Tyto data poté parsuje a snaží se z nich vytáhnout všechny body, které budou následovat na trase. Poté je seřadí a odstraní duplicitní body a uloží je do určité kolekce, tak aby se s nimi dalo později pracovat.

6.2.12 WayAndTurning

Pomocná třída, která je vyrobena do budoucna, pokud by bylo možné s volantem otáčet v jakémkoliv rozsahu a poté volant vrátit zpět na původní pozici.

7 Testování aplikace

Testování aplikace bych rád rozdělil do dvou částí. V první části se budu věnovat testování detekce objektů pomocí kamery na mobilním zařízení a také detekci překážek pomocí lidarového čidla. V druhé části budu věnovat autonomní řízení po dané trase pomocí velmi hustého rozdělení bodů na trase.

7.1 Testování detekce překážek pomocí kamery

Toto testování probíhalo jak na uložené kolekci obrázků nebo videí, tak také v reálném prostředí snímáním přes kameru mobilního zařízení.
Při testování algoritmu z odstavce 5.3.3 jsem měl vzorek o velikosti 1521 obrázků obličejů. Obrázky jsem použil z volně dostupné databáze obličejů [26]. Rychlost algoritmu nad vzorkem [26] byla testována 10 krát. Střední hodnota rychlosti algoritmu byla 304.28 s. Nejnižší naměřený čas z 10 testování byl 299.31 s. Nejvyšší naměřený čas z 10 testování byl 301.54 s. Z kolekce čítající 1521 obličejů algoritmus dokázal vyhodnotit jako 1509 positivních nálezů, to je pravděpodobnost 99.21 %. Průměrný čas pro testování jednoho snímku byl 0.199 sekund. Tento algoritmus byl testován pouze na jednom vzorku.
Testování algoritmu z odstavce 5.3.3 přes kameru mobilního zařízení proběhlo na 5 různých osobách celkem padesátkrát. Z padesáti zaměření kamerou na osobu algoritmus detekoval obličej osoby 42 krát. Nejkratší čas k nalezení obličeje pomocí kamery byl 0.72 sekund. Nejdelší čas byl 4.23 sekund. Průměrný potřebný čas k detekci obličeje v reálném čase na algoritmu 5.3.3 byl 2.1 sekundy.
Při testování algoritmu detekce automobilu z odstavce 5.3.5 jsem použili dva statistické vzorky z University of Illionis [23]. Také jsem využil jejich soubory obsahující pravidla pro vyhledávání automobilů na obrázku. První vzorek obsahoval 169 obrázků. Druhý vzorek obsahoval 107 obrázků. Rychlost algoritmu nad vzorkem 1 byla testována 10 krát. Střední hodnota délky trvání algoritmu [23] nad vzorkem 1 byla 32.25 sekund. Nejnižší naměřená doba trvání algoritmu byla 29.25 sekund. Nejvyšší naměřená doba trvání algoritmu byla 34.57 sekund. Úspěšnost detekce automobilu u vzorku 1 byla 11.25 %. Průměrná doba detekování jednoho automobilu nad vzorkem 1 byla 0.19 sekund. Rychlost algoritmu nad vzorkem 2 byla testována 10 krát. Střední hodnota délky trvání algoritmu [23] nad vzorkem 2 byla 50.3 sekund. Nejnižší naměřená délka trvání algoritmu nad vzorkem 2 byla 49.1 sekund. Nejvyšší naměřená délka 52.1 sekund. Úspěšnost detekce automobilu u vzorku 2 byla 55.1 %. Doba potřebná k detekci jednoho automobilu byla 0.47 sekund. Obrázky ze vzorku 1 měly velikost 250 px $\times$ 123 px. Obrázky ze vzorku 2 měly velikost 283 px $\times$ 171 px. Pro testování přes kameru mobilního zařízení se nepodařilo tento algoritmus úspěšně otestovat. Mobilní zařízení nedisponovalo potřebným výkonem.

Algoritmus pro detekci pozadí z kapitoly 5.3.1 jsem testoval na dvou krátkých videích. První vzorek měl délku 5 s a 30 snímků za sekundu. Video bylo natáčeno přes mobilní telefon upev-

něný na stativu v místnosti. Úspěšnost testování pomocí algoritmu z kapitoly 5.3.1 byla 81.3 %. Druhý vzorek měl délku 10 s a 30 snímků za sekudu. Video bylo natáčeno přes mobilní telefon upevněný na stativu na parkovišti za denního světla. Úspěšnost testování pomocí algoritmu z kapitoly 5.3.1 byla 68.1 %. Tento procentuální rozdíl je zapříčiněn, jelikož při venkovním natáčení se různě odráželo světlo a tím algoritmus detekoval falešné pohyby.

Algoritmus pro detekci pohybu z kapitoly 5.3.2 jsem testoval na stejných testovacích množinách jako algoritmus pro detekci pozadi (5.3.1). Úspěšnost testování na prvním vzorku byla 87.2 %. Úspěšnost na druhém vzorku byla 64.7 %. Tento procentuální rozdíl je zapříčiněn, protože při venkovním natáčení se různě odráželo světlo a tím algoritmus detekoval falešné pohyby.

7.2 Lidarové čidlo

Doba, kterou trvá lidarovému čidlu nasnímat překážku je 100 ms. Lidar snímá každých 0.36 stupňů. Lidar jsem nastavoval přímo na automobilu. Pokud lidar detekuje překážku na 15 cm, automobil zastaví. Do rychlosti 2.5 km/h automobil bezpečně zastaví na vzdálenost 15 cm od překážky. Lidar je nastaven tak, aby detekoval překážku i podélně okolo automobilu na vzdálenost 5 cm.

7.3 Testování autonomního řízení

Testování aplikace pro autonomní pohyb jsem testoval na dvou místech. Pro první testování jsem zvolil trasu od souřadnic $Lat : 49.805336, Lng : 18.125792$ do souřadnic $Lat : 49.805072, Lng : 18.126075$. První trasu jsem testoval 15 krát. Při testování aplikace jsem pokaždé zvolil jiný úhel natočení mobilního zařízení. Aplikaci jsem testoval jako navigaci. Z patnácti testování aplikace dosáhla cíle 8 krát. V ostatních měřeních byl problém s výpadkem signálu GPS.
Pro druhé testování jsem zvolil trasu od souřadnic $Lat : 49.832000, Lng : 18.161057$ do souřadnic $Lat : 49.832134, Lng : 18.161560$. Toto testování jsem prováděl jak formou navigace tak formou ovládání automobilu. Formou navigace jsem tuto aplikaci otestoval 10 krát. Pokaždé jsem zvolil jiný počáteční úhel natočení mobilního zařízení. Aplikace byla úspěšná v čtyřech testech z deseti. Neúspěšné testy byly zapříčiněny nepřesností GPS systému. Formou řízení autonomního automobilu jsem aplikaci otestoval 20 krát. V tomto případě ani jednou nedokázal automobil dojet do cíle. Toto bylo zapříčiněno pomalou odezvou kompasu, GPS systému a nepřesným řízením volantu.

8 Závěr

V mé diplomové práci jsem se věnoval teorii řízení robotů a detekci objektů pomocí kamery a senzorů. Pro pochopení tohoto problému jsem si musel nastudovat algoritmy pro práci s obrazem a algoritmy pro řízení a detekci směru.
Ve své aplikaci používám detekci obličeje, detekci pozadí a detekci pohybu. Detekce obličeje je použitelná metoda v reálném světě. Pro detekci obličeje jsem využil knihoven *OpenCV*. V dnešní době pomocí této metody můžeme mít přehled o pohybu osob na letišti, fotbalových hřištích a velkých prostorech. Tato detekce by mohla mít propojení s databází policií. Další z využití je zamykání a odemykání pomocí detekce obličeje.
Detekci pozadí a detekci pohybu využívám k nalezení překážek před mobilním zařízením. V mém případě pro aplikaci těchto algoritmů je nutné, aby nebylo mobilní zařízení v pohybu. Využití těchto algoritmů je v automobilismu. Detekování pohybu může předejít nehodě.
Má aplikace na autonomní řízení vozidla obsahuje přehlednou mapu, na které jsou vyznačené body trasy. Tyto body jsou velmi hustě rozložené tak, aby automobil neměl problém s nalezením dalšího následujícího bodu. Také ukazuje na aktuální pozici automobilu a šipka ukazuje, jakým směrem automobil směřuje. Při pohybu automobilu se zaznamenává trasa pohybu a následně se zobrazuje na mapě. Uživatel tedy může vidět, jak moc se automobil přiblížil navrhované trase. Aplikace vypočítává úhel mezi aktuální pozicí natočením a následujícím bodem, ke kterému by se měl automobil přiblížit. Také určuje vzdálenost mezi aktuální pozicí a následujícím bodem. Také dokáže určit vzdálenost do cíle.Pro řízení autonomního robota jsem využíval teoretických podkladů [16].
Aplikaci jsem testoval jak na umělých datech, tak na reálném prostředí. Opakujícím se testováním jsem odstranil chyby, které nastaly při běhu programu a zlepšil jsem chod aplikace. Tato práce s detekcí v reálném čase a navigováním byla zajímavá a rád bych se ji věnoval i v příštím čase.

Literatura

[1] Článek o vozítku Couriosity [online]. [cit. 2016-04-17].

<https://en.wikipedia.org/wiki/Curiosity_(rover)>.

[2] Přehled autonomních systémů v automobilech [online]. [cit. 2016-04-17].

<http://www.fastcompany.com/3024362/innovation-agents/10-autonomous-driving-companies-to-watch>.

[3] Autonomní řízení [online]. [cit. 2016-04-17].

<https://www.google.com/selfdrivingcar/>.

[4] Jak se řídí Google car [online]. [cit. 2016-04-17].

<http://www.makeuseof.com/tag/how-self-driving-cars-work-the-nuts-and-bolts-behind-g

[5] Přehled o firmě Delphi [online]. [cit. 2016-04-17].

<http://www.freep.com/story/money/business/2015/03/13/delphi-autonomous-test-drive-/70198196/>.

[6] Obrázek automobilu firmy Delphi [online]. [cit. 2016-04-17].

<http://www.gannett-cdn.com/-mm-/4b4285c1d64d50de5d52d9f73b4d2a8eb66f33a6/c=326-0-5433-3840&r=x408&c=540x405/local/-/media/2015/03/12/DetroitFreePress/DetroitFreePress/635617457749453028-Automated-Driving-Vehicle-Passen
jpg >.

[7] Přehled o firmě QNX [online]. [cit. 2016-04-17].

<http://www.qnx.com/solutions/industries/automotive/driver_assistance.html?lang=de/>.

[8] Technologie Volvo [online]. [cit. 2016-04-17].

<http://www.volvocars.com/intl/about/our-innovation-brands/intellisafe/intellisafe-autopilot/this-is-autopilot/the-tech/>.

[9] Přehled nejnovějších technologií Mercedes-Benz [online]. [cit. 2016-04-17].

<https://www.mbusa.com/mercedes/technology/>.

[10] Recenze o řízení poloautonomního Nissanu Leaf I [online]. [cit. 2016-04-17].

<http://fortune.com/2016/01/08/self-driving-nissan-leaf/>.

[11] Recenze o řízení poloautonomního Nissanu Leaf II [online]. [cit. 2016-04-17].

<http://www.digitaltrends.com/cars/nissan-leaf-piloted-drive-1-0-concept-specs-news/>

[12] Vývoj autonomních systémů firmou Continental [online]. [cit. 2016-04-17].

<http://articles.sae.org/14039/>.

[13] Interview o řízení a přehledu technologií používaných v elektro automobilu Tesla [online]. [cit. 2016-04-17].

<https://www.technologyreview.com/s/600772/10-breakthrough-technologies-2016-tesla-au

[14] Obrázek řídícího pultu automobilu Tesla [online]. [cit. 2016-04-17].

<https://d267cvn3rvuq91.cloudfront.net/i/images/ma16-10tesla-3.jpg?sw=1180&cx=0&cy=0&cw=1803&ch=1352>.

[15] Detekce okolí v automobilu Tesla [online]. [cit. 2016-04-17].

<http://www.gizmag.com/tesla-autopilot-test/40642/pictures#9>.

[16] Autonomous robot navigation using adaptive potential fields,*Teorie autonomního řízení robotů* [online]. [cit. 2016-04-17].

<http://www.sciencedirect.com/science/article/pii/S0895717704003097>.

[17] Genetické algoritmy a jejich aplikace v praxi [online]. [cit. 2016-04-17].

<http://programujte.com/clanek/2005072601-geneticke-algoritmy-a-jejich-aplikace-v-pra

[18] Line segments detector [online]. [cit. 2016-04-17].

<http://docs.opencv.org/3.0-beta/modules/line_descriptor/doc/LSDDetector.html>.

[19] Line segments detector theory [online]. [cit. 2016-04-17].

<http://www.ipol.im/pub/art/2012/gjmr-lsd/article.pdf>.

[20] Robust techniques for background subtraction in urban traffic video [online]. [cit. 2016-04-17].

<https://computation.llnl.gov/casc/sapphire/pubs/UCRL-CONF-200706.pdf>.

[21] Background subtraction [online]. [cit. 2016-04-17].

<https://en.wikipedia.org/wiki/Background_subtraction>.

[22] Face Detection using Haar Cascades [online]. [cit. 2016-04-17].

<http://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.html#gsc.tab=0>.

[23] UIUC Image Database for Car Detection [online]. [cit. 2016-04-17].

<http://cogcomp.cs.illinois.edu/Data/Car/>.

[24] Lidary a letecké laserové skenování [online]. [cit. 2016-04-17].

<http://wvc.pf.jcu.cz/ki/data/files/160lidaryweb.pdf>.

[25] Scanning range finder [online]. [cit. 2016-04-17].

<https://www.hokuyo-aut.jp/02sensor/07scanner/urg_04lx.html>.

[26] Database of faces [online]. [cit. 2016-04-17].

<https://ftp.uni-erlangen.de/pub/facedb/>.