Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Riadenie vozidla po dráhe osadenej 3D značkami

Diplomová práca

2018 Bc. Katarína Šimnová

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Riadenie vozidla po dráhe osadenej 3D značkami

Diplomová práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Mgr. Andrej Mihálik, PhD.

Bratislava, 2018 Bc. Katarína Šimnová

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Katarína Šimnová
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: aplikovaná informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Riadenie vozidla po dráhe osadenej 3D značkami.
Vehicle control over the trajectory using 3D markers.

Anotácia: Autonómne auto je auto bez šoféra, ktoré je schopné sledovať okolité prostredie a pohybovať sa v ňom bez ľudského zásahu. Niektoré autá využívajú na autonómne riadenie kamery, radary a senzory. Iné vozidlá disponujú profesionálnym systémom skenovania okolitého prostredia. Rozpoznávanie prostredia a identifikácia polohy je výpočtovo náročný proces, preto sme sa rozhodli riadiť auto využitím profilometra, tak že na ceste budú osadené 3D značky.

Cieľ: Navrhnúť vhodný tvar 3D značiek pre rozpoznávanie profilometrom. Navrhnúť algoritmus rozpoznávania vodiacich značiek osadených na ceste. Pokúsiť sa nájsť ideálnu trajektóriu kolies pri otáčaní do oblúka. Preskúmať možnosti autonómneho riadenia po trajektórií vyznačenej vodiacimi značkami.

Literatúra: http://www.cs.princeton.edu/~funk/iccv09.pdf

Vedúci: Mgr. Andrej Mihálik, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 09.10.2016

Dátum schválenia: 14.10.2016 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

Šimnová
.....
študent

.....
vedúci práce

Abstrakt

Vývoj autonómnej mobility je v posledných rokoch veľkým trendom v oblasti výskumu vo svete. Cieľom tohoto projektu na FMFI je priniesť alternatívu k existujúcim riešeniam a prispieť tak k zlepšeniu bezpečnosti a komfortu na cestách. Hlavnou ideou nie je vytvoriť vozidlo schopné pohybovať sa po všetkých bežných cestách ale vytvoriť špeciálne upravenú inteligentnú vozovku, ktorá vytvorí dokonalé podmienky pre bezpečnú, plynulú a bezchybnú jazdu vozidla z bodu A do bodu B. Pohyb v rámci trasy autonómneho vozidla je riadený globálne. Na lokálnej úrovni sa rieši optimálna jazda po vozovke na rovnej ceste a v zákrutách pomocou vodiacich 3D značiek. Táto diplomová práca rieši návrh a implementáciu rozpoznávania takýchto 3D značiek. Druhá časť práce je venovaná najväčšiemu problému, ktorý brzdil vývoj 3D značiek. Ide o problém kedy vozidlo v zárute nie je schopné efektívne snímať profil vozovky pred sebou čoho výsledkom je nespoľahlivé rozpoznávanie.

Abstract

The development of autonomous mobility has been a major trend in research in the world over recent years. The aim of this project at FMFI is to provide an alternative to existing solutions to help improve road safety and comfort. The main idea is not to create a vehicle capable of moving on all normal routes but to create a specially modified intelligent road that will create the perfect conditions for safe, smooth and faultless driving. Motion within the autonomous vehicle route is managed globally. At the local level, optimal road ride on a straight road and in a twist is guided by guiding 3D tags. This diploma thesis solves the design and implementation of recognition of such 3D tags. The second part is devoted to the biggest problem that has hampered the development of 3D tags. This is a problem when the warranty vehicle is unable to effectively capture the road profile ahead of itself, resulting in unreliable recognition.

Obsah

1 Úvod **1**
1.1 Cieľ . . . 2

2 Východiská práce **3**
2.1 Automatické vs. autonómne autá . . . 4
2.2 Autonómne vozidlá . . . 5
2.3 Priemyselná automatizácia . . . 7
2.4 Existujúce riešenia . . . 8
2.4.1 Sedric . . . 8
2.4.2 Tesla Autopilot . . . 9
2.4.3 Waymo . . . 10

3 Technické vybavenie laboratória **11**
3.1 Gocator . . . 11
3.2 Vozidlo . . . 15

4 Návrh riešenia pre rozpoznávanie značiek **17**
4.1 3D vodiace značky . . . 17
4.2 Virtuálna mapa . . . 18
4.3 Dáta na vstupe . . . 20
4.4 Požiadavky na algoritmus . . . 21
4.5 Architektúra softvérového riešenia . . . 24

5 Rozpoznávanie značiek **27**
5.1 Algoritmus s využitím bitmáp . . . 28
5.2 Algoritmus s využitím hrán . . . 30
5.3 Výsledky algoritmov rozpoznávania . . . 33

6 Problémy vozidla v zákrutách **39**
6.1 Ackermannov model riadenia . . . 39
6.2 Simulácia riadenia . . . 40
6.3 Evolučný algoritmus pre nájdenie ideálnej trajektórie auta . . . 43

6.4 Analytické riešenie 47

7 Záver **49**

1. Úvod

Autonómny automobil je vozidlo, ktoré je schopné zosnímať svoje prostredie a získané dáta použiť tak, aby bolo schopné navigovať sa bez akéhokoľvek ľudského vstupu. Aktuálne sa vynakladá veľké úsilie na zkonštruovanie čo najdokonalejšej verzie takéhoto vozidla. Hlavné výhody autonómnosti zahŕňajú zníženie nákladov na mobilitu a infraštruktúru, zvýšenú bezpečnosť, zvýšenú mobilitu a spokojnosť zákazníkov.

Niekoľko firiem z oblasti automobilového priemyslu ponúka rôzne modely autonómnych vozidiel, ale väčšinou sa jedná o koncepty alebo o čiastočne autonómne vozidlá. Zdokonaľovanie týchto existujúcich riešení je stále predmetom výskumu. Hlavnými brzdami nasadenia sú technologické výzvy, spory týkajúce sa zodpovednosti, obavy o bezpečnosť spotrebiteľov, riziko straty súkromia, obavy o bezpečnosť z hľadiska hackerov alebo terorizmu a obavy z výslednej straty pracovných miest súvisiacich s riadením v odvetví cestnej dopravy.

Problémy z technologického hľadiska sa týkajú napríklad chýb pri rozpoznávaní okolitého prostredia, ktoré mávajú často nepríjemné následky. Zatúlané autonómne vozidlo môže pri snahe dostať sa do cieľa poškodiť seba alebo svoje okolie. Ďalším problémom je nízka rýchlosť vozidla, ktorá súvisí s potrebou dbať pri jazde na bezpečnosť ľudí vo svojom okolí a tomu prispôsobiť rýchlosť a štýl jazdy. Nezanedbateľný čas musí vozidlo venovať aj samotnému čítaniu a rozpoznávaniu prostredia, pretože ide o spracovanie veľkého množstva dát.

Ideou výskumu je namiesto drahého a komplikovaného systému, ktorý dokáže jazdiť po dnešných cestách, vytvoriť systém, ktorý bude jazdiť po špeciálne upravených cestách. Výsledkom bude lacnejší, presnejší, spoľahlivejší a bezpečnejší systém. Hlavnými cieľmi je maximalizovať bezpečnosť a kvalitu rozpoznávania okolia a minimalizovať čas potrebný na rozpoznávanie, aby bolo vozilo schopné bezpečného pohybu vo vyšších rýchlostiach.

V druhej kapitole sa nachádzajú východiská práce. Venujem sa tu hlavne definícii autonómnych vozidiel a popisu ich využitia. Ďalej sa tu nachádza zoznam aktuálnych existujúcich riešení.

V tretej kapitole je popis vybavenia laboratória. Popisujem komponenty, z ktorých

je poskladané testovacie vozidlo. Špeciálna pozornosť je venovaná najdôležitejšiemu senzoru vozidla - profilometru.

Vo štvrtej kapitole sa nachádza návrh riešenia. Je tu popísaný návrh tvarov 3D vodiacich značiek, virtuálna mapa dráhy a dáta na vstupe algoritmu. Koniec kapitoly je venovaný požiadavkám na algoritmus a z neho vyplývajúci návrh architektúry riešenia.

V piatej kapitole je popis algoritmov na rozpoznávanie značiek, ich problémov a porovnanie výsledkov.

Posledná kapitola je venovaná simulácii jazdy vozidla v zákrute a hľadaniu optimálnej trajektórie vozidla s použitím evolučného algoritmu.

1.1 Cieľ

Úlohy, ktoré diplomová práca rieši sú súčasťou projektu, ktorý sa venuje problematike automatických transportných systémov. Cieľom projektu je zostrojiť vozidlo, ktorésa skenovaním a rozpoznávaním značiek bude schopné autonómne pohybovať vo vyznačenom priestore vo veľkých rýchlostiach. Cieľom mojej diplomovej práce je navrhnúť vhodné tvary značiek a implementovať ich rozpoznávanie. Vozidlo musí byť schopné orientovať sa vo virtuálnej mape priestoru, udržať sa na ceste a správne odbočovať. V práci sa venujem aj problematike optimalizácie riadenia vozdila, ktorá je potrebná pre zisk vhodných dát pre potreby rozpoznávania.

Konkrétnymi úlohami, ktoré riešim sú:

- návrh a implementácia robustného algoritmu rozpoznávania, ktorý rieši všetky možné okrajové prípady a problémy
- testovanie navrhnutého riešenia
- vizualizácia nasnímaných dát
- simulácia pohybu auta po vozovke
- vypočet a vizualizácia trajektórie auta, ktorá bude naoptimálnešia pre zisk potrebných dát z prostredia

2. Východiská práce

Výsledkom projektu je vozidlo určené na prepravu osôb alebo vecí. Vozidlo nebude riadiť človek, bude sa pohybovať samostatne. Orientovať sa bude na základe rozpoznávania značiek na vozovke. Pohyb takéhoto vozidla v bežnej premávke zatiaľ nie je vhodný z bezpečnostých dôvodov a z dôvodu spomalenia plynulosti premávky. Vozidlá sa budú pohybovať len vo vopred určenej zóne, v ktorej sa neočakáva pohyb neautonómnych vozidiel. Vozidlo bude mať k dispozícii virtuálnu mapu priestoru, v ktorom sa môže pohybovať. Trasa jazdy bude určená centrálnym globálnym systémom.

O správnu polohu auta na vozovke, správne odbočovanie, parkovanie atď. sa bude starať každé auto samostatne na lokálnej úrovni s použitím vodiacich značiek na vozovke. Značky na ceste je potrebné vhodne navrhnúť tak, aby ich bolo možné jednoznačne a rýchlo prečítať a rozoznať. Bude vytvorená séria značiek, pričom každej z nich bude pridelená nejaká funkcia. Pomocou nich bude vozidlo získavať informácie o svojej polohe na vozovke a možnostiach svojho pohybu. Vo výsledku budú značky hovoriť o tom, či vozidlo može ísť dopredu, odbočiť vpravo, odbočiť vľavo prípadne a či môže parkovať.

Systém využívajúci rozpoznávanie obrazových snímkov s použitím počítačového videnia nie je vhodný kvôli časovej náročnosti. Značky budeme rozoznávať na základe ich tvaru v 3D priestore.

V praxi to znamená, že na vozidle bude pripevnený prístroj (profilometer), ktorý dokáže snímať vozovku pred ním a zachytiť jej profil. Tento prístroj bude pomocou laseru zachytávať profil vozovky so všetkým čo sa na nej nachádza. Výstupom systému nie je tradičná snímka, nemáme k dispozícii farebnú informáciu. Výstupom je len zhluk bodov a ich poloha v priestore. Profilometer pri pohybe po vozovke skenuje po čiarach. Jednotkou výstupu profilometra je tzv. skenovacia čiara (resp. riadkový sken), ktorá obsahuje body nachádzajúce sa na jednej priamke. Vyselektovaním niekoľkých skenovacích čiar získame potrebné údaje k danej značke. Koľkokrát stihne profilometer zachytiť skenovacou čiarou značku závisí nielen od nastavenia frekvencie profilometra, ale aj od rýchlosti vozidla. Pri tomto procese skenovania a rozpoznávania sme časovo obmedzení. Bude potrebné z veľkého množstva dát vybrať len dáta, ktoré nás zaujímajú, a ostatné

zahodiť.

Zámery projektu:

- Univerzálnosť a komplexnosť: Systém zabezpečí v ohraničenom priestore vysoko organizovanú efektívnu a bezpečnú osobnú aj nákladnú automobilovú dopravu.
- Budú použité priemyselne vyrábané elektrické vozidlá rôznych typov (osobné, autobusy, nákladné atď.) riadené palubnými počítačmi.
- Budú nasadené elektromobily s nulovými emisiami.
- Informáciu o polohe vozidla na vozovke bude autonómne vozidlo získavať cez navigačný systém čítaním pasívnych identifikátorov na vozovke.
- Reakčný čas systému by mal byť menej ako 0,1 sekundy.
- Systém bude schopný vyriešiť akúkoľvek dopravnú situáciu, križovatku, parkovanie, automatické nabíjanie.
- Centrálny riadiaci systém (CRS) riadi dopravu tak, aby bola vysoko organizovaná, plynulá a bezpečná.
- Nízka cena.

2.1 Automatické vs. autonómne autá

V súvislosti s vozidlami bez vodiča vznikli rôzne zmätky z hľadiska názvoslovia. Samostatnosť vozidla môžeme rozdeliť na niekoľko úrovní [1]:

- Automatický - zariadenie je schopné vykonávať súbor súvisiacich funkcií automaticky. Fukcie sú vykonávané na pokyn operátora. Pre semi-autonomitu alebo autonomitu je potrebné spojiť veľké množsvo automatických sekvencií.
- Semi-autonómny - niekoľko automatických sekvencií, ktoré môže stroj vykonávať bez ľudského vstupu. Semi-autonómna prevádzka stroja predpokladá, že prevádzkovateľ vykonáva určité úlohy.
- Autonómny - zariadenie je schopné vykonávať programované operácie za definovaných podmienok bez ľudského vstupu alebo vedenia.

Level	Koncept	Definícia	
0	Riadené človekom	Prístroj je po celý čas kontrolovaný človekom.	
1	Automatické	Človek je zodpovedný za bezpečnú prevádzku prístroja. Niektoré úkony dokáže prístoj vykonávať bez ľudského vstupu.	
2	Semi-autonómne	Prístroj vykonáva množinu svojich úloh bez pomoci človeka. Človek vykonáva zvyšné úlohy.	
3	Autonómne	Prístroj dokáže vykonávať všetky definované úlohy bez pomoci človeka. Zodpovedá aj za bezpečnosť.	

Obr. 2.1: Úrovne autonomity.

2.2 Autonómne vozidlá

Vývoj modernej automobility začal jednoduchými zlepšeniami akými sú tempomat, palubný počítač, parkovacie kamery alebo rôzne bezpečnostné asistenčné systémy. Túto sekciu som spracovala podľa informatívneho prehľadového článku o evolúcii autonómnych vozidiel [1]. Auto už nie je len mechanickým zariadením, vo vozidlách pribúda stále viac elektroniky. Digitálny svet sa stále rozširuje a automobilový priemysel s ním musí držať krok. Dnešnými cieľmi sú autonómnosť, zvýšenie ekologickosti, rôzne využitia pripojenia auta na internet. Prvoradé je zvýšenie komfortu a bezpečnosti.

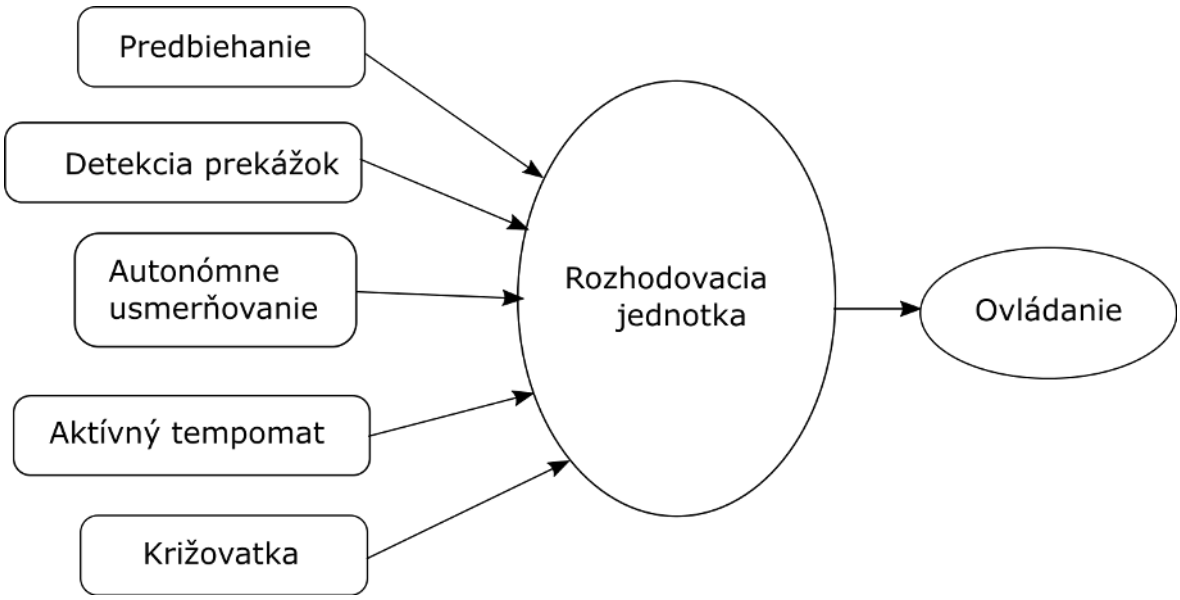
Prvým krokom pri vývoji je upravenie poháňacích zariadení tak, aby bolo možné automatické ovládanie. Treba prerobiť volant, brzdy a plyn tak, aby ich bolo možné riadiť pomocou elektroniky.

V ďalšom kroku je potrebné zabezpečiť dôkladnú analýzu prostredia. Pre dosiahnutie ideálnej kontroly nad vozidlom je možné využiť sériu rôznych senzorov:

- bezdrôtová lokálna sieť pre komunikáciu s okolitou infraštruktúrou a inými vozidlami.
- RTK-DGPS (diferenciálne GPS s využitím kinematickej metódy v reálnom čase) sa používa na získanie informácií o jazde. S týmito informáciami a mapou cesty vieme riadiť auto v priestore s centimetrovou presnosťou.
- IMU (Inerciálna meracia jednotka) je sekundárny polohovací systém, ktorý dopĺňa GPS, ak jeho presnosť nestačí pre bezpečnú jazdu. Dodáva informácie o rýchlosti, rýchlosti otáčania, smere, sklone vozidla a zrýchlení.
- Kamery pre lepšie vnímanie prostredia. Použitie na detekciu chodcov a iných prekážok.

Vychádzajúc zo skutočných dopravných podmienok a situacií musí byť autonómne vozidlo schopné vykonávať niekoľko manévrov:

- Autonómne usmerňovanie - ide o kontrolu samotného auta bez ohľadu na okolie. Kontroluje sa správna poloha auta na rovných úsekoch a v zákrutách pre ovládanie volantu.
- Aktívny tempomat s funkciou „Stop and Go“ pomáha udržať bezpečnú vzdialenosť od okolitých áut. Pomocou bezdrôtovej lokálnej siete auta vieme získať informácie o okolitých vozidlách.
- Predbiehanie - na dvojsmernej ceste sa dá predbiehanie chápať ako dvojitú zmenu jazdného pruhu. Používa sa podobná kontrola ako pri autonómnom usmerňovaní.
- Križovatka - použijú sa polohy, vzdialenosti a rýchlosti vozidiel na križovatke a určí kedy a akou rýchlosťou cez križovatku prejde.
- Detekcia a ochrana chodcov.



Obr. 2.2: Diagram manévrov.

2.3 Priemyselná automatizácia

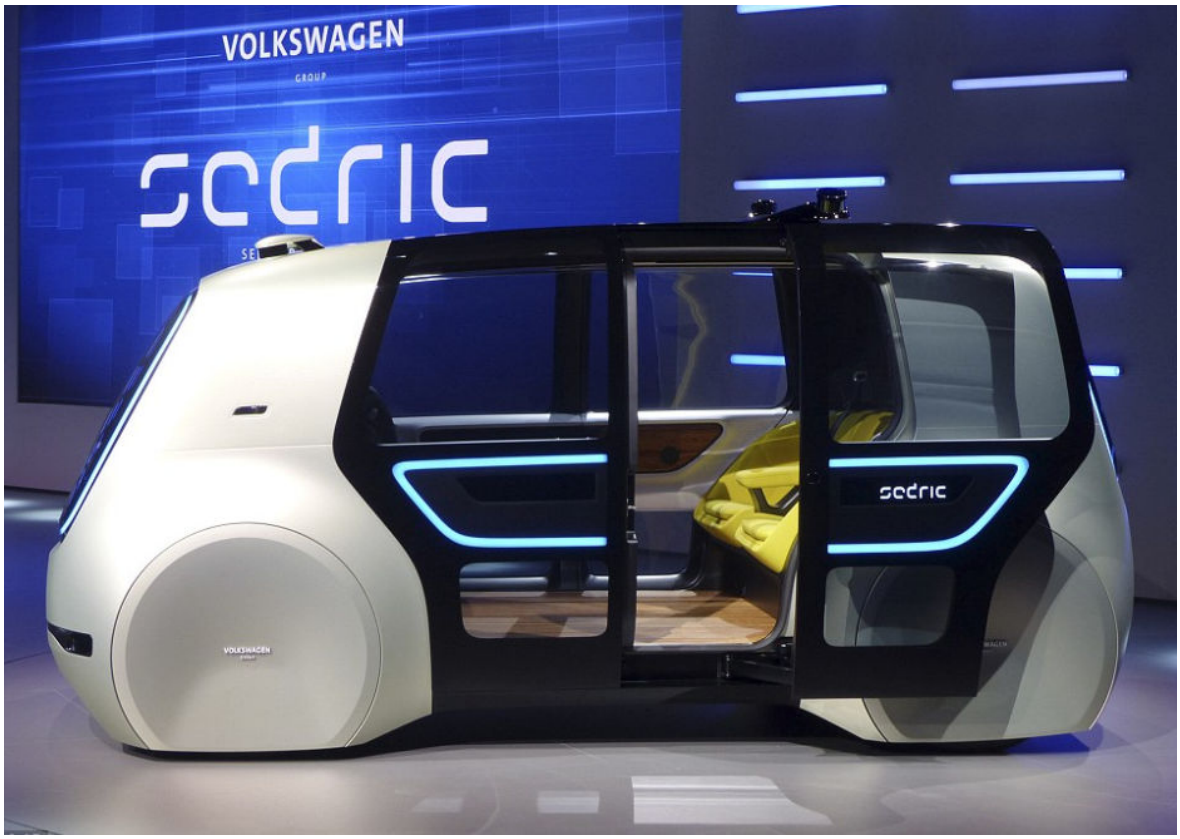
Priemyselná automatizácia je použitie riadiacich systémov, ako sú počítače alebo roboty, a informačné technológie na manipuláciu s rôznymi strojmi v priemysle. Účelom týchto strojov je nahradenie človeka. Je to cesta, ako dosiahnuť vyššiu efektívnosť a kvalitu výroby, a pritom zredukovať výrobné náklady. Pôvodne bola automatizácia skôr nástrojom na zvýšenie produktivity. Výroba nebola obmedzená dovolenkami ani finančne zaťažená rôznymi výdavkami na zamestnancov. Momentálne sa na automatizácii veľmi cení zvýšenie kvality výroby a zvýšenie bezpečnosti pri výrobe. Chybovosť ľudí v porovnaní so strojmi je štatisticky neporovnateľne vyššia. Nevýhodou sú vysoké počiatočné náklady a výdavky spojené so zaúčaním ľudí.

Jedným zo systémov priemyselnej automatizácie je automaticky riadené vozidlo. Je to robot, ktorý sa pomocou čítania značiek a čiar dokáže automaticky pohybovať v priestore. Na snímanie okolitého sveta sa používa buď kamera alebo rôzne laserové 3D snímače. Rozpoznávanie prvkov okolia je následne vykonané metódami počítačového videnia alebo počítačovej grafiky. Takéto vozidlá sa najčastejšie používajú pri prevoze materiálu zo skladu do skladu alebo pri premiestňovaní materiálov pre výrobné zariadenie. Ak hovoríme o automatickom vozidle v súvislosti s priemyselnou automatizáciou je zrejmé, že takéto auto sa bude pohybovať len v rámci nejakej priemyselnej zóny.

2.4 Existujúce riešenia

2.4.1 Sedric

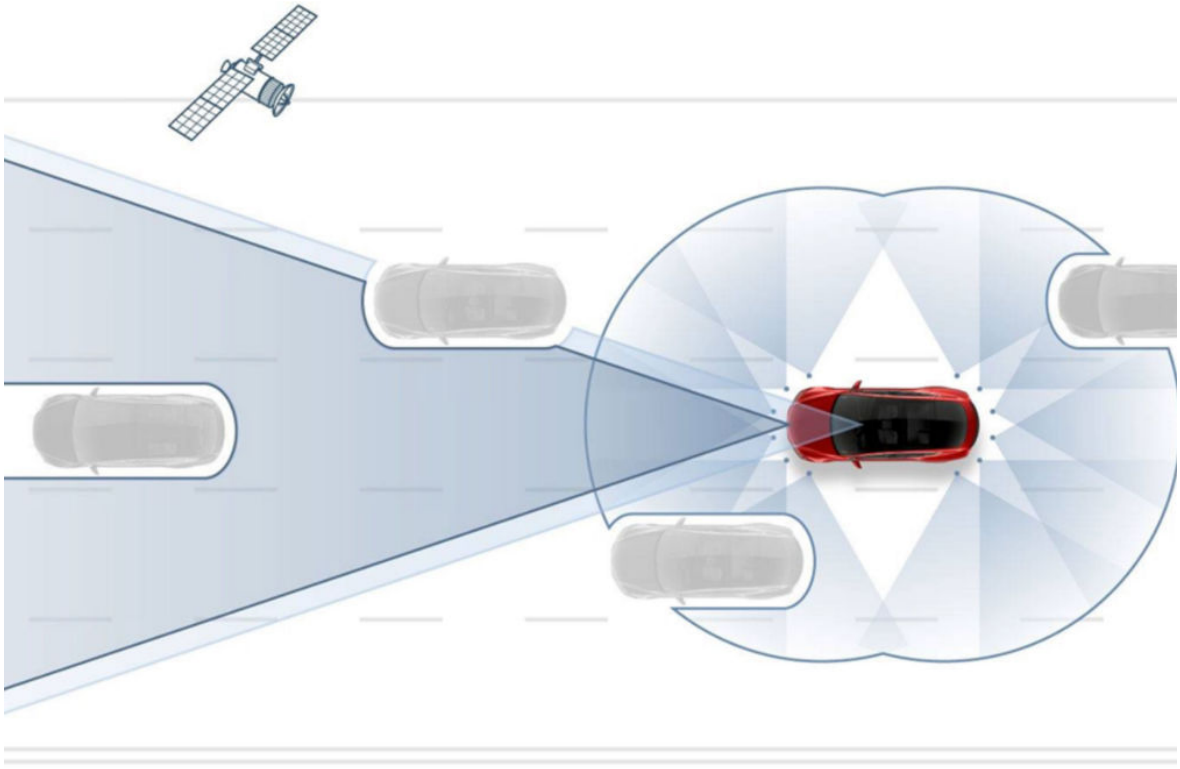
Sedric je prvý samojazdiaci automobil od spoločnosti Volkswagen. Je schopný vykonávať nezávisle všetky jazdné funkcie. Je veľmi jednoduchý na ovládanie - nepotrebuje volant ani pedále. Je elektricky poháňaný, vďaka čomu je tichý a ekologický. Základom konceptu je vysoká úroveň automatizácie. Vozidlá dokážu pracovať bez ľudských vstupov na akýchkoľvek cestách za akýchkoľvek podmienok. Umožňuje jednoduchú, pohodlnú a bezpečnú jazdu pre každého vrátane detí jediným dotykom tlačidla. Sedric je zatiaľ len koncept, je otázne kedy toto vozidlo bude možné reálne používať [2].



Obr. 2.3: Sedric - Prvý samojazdiaci automobil od spoločnosti Volkswagen[2].

2.4.2 Tesla Autopilot

Tesla Autopilot je systém od spoločnosti Tesla, ktorý umožňuje prepnúť ovládanie auta do semi-autonómneho stavu. Tento systém je v dnešnej dobe reálne používaný, ale nie je plne autonómny. Je potrebná neustála pozornosť vodiča aj počas doby, kedy sa auto riadi samo. Autopilot je tvorený viacerými snímačmi umiestnenými všade okolo auta. Tieto senzory pomáhajú automobilu pochopiť jeho prostredie, aby sa mohlo bezpečne riadiť vo väčšine situácií na cestách. Vozidlo, ktoré má aktivovaný autopilot, je schopné samostatne otáčať volantom, prechádzať medzi pruhmi vozovky a parkovať. Momentálne je jeho najväčšou výhodou zvýšenie bezpečnosti na cestách. Systém nedovolí vodičovi zraziť sa s iným autom, prekážkou alebo človekom[3].



Obr. 2.4: Semi-autonómny systém od spoločnosti Tesla[3].

2.4.3 Waymo

Waymo je projekt samojazdiacich áut od spoločnosti Google. Vozidlá majú senzory a softvér, ktoré umožnujú detekciu chodcov, cyklistov a vozidiel vo vzdialenosti až dvoch futbalových ihrísk vo všetkých smeroch. Vozidlo by malo byť schopné samostatne jazdiť vo väčšine prostredí a podmienok na ceste. Každou verziou sa systém vylepšuje, pričom dosiahnutie želanej funkčnosti sa očakáva do roku 2020[4].

K vybaveniu Waymo patrí:

- Lidar, ktorý pomocou lasera mapuje svoje okolie,
- Radarový senzor, ktorý dokáže detekovať objekty v hmle, daždi a snehu,
- 360° kamera.

Obr. 2.5: Waymo - plne autonómny systém od spoločnosti Google[4].

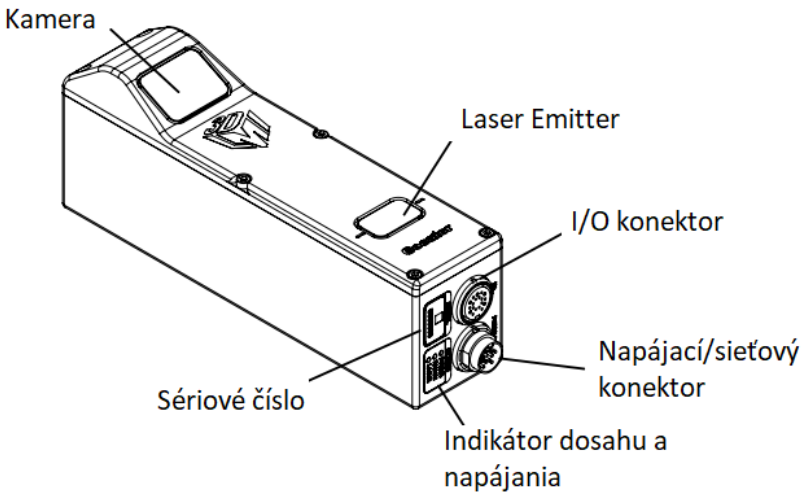
3. Technické vybavenie laboratória

3.1 Gocator

Gocator je vysokorýchlostné (32 kHz) laserové zariadenie na bezkontaktné meranie profilov. Prístroj vyšle laserové lúče smerom k profilu, od ktorého sa lúče odrazia. Odrazené lúče zachytí kamera a na základe ich vlastností dokáže charakterizovať povrch, od ktorého sa lúče odrazili. Prístroj bol vytvorený spoločnostou LMI Technologies v rámci vývoja novej generácie inteligentných senzorových 3D technológií pre rýchle a presné meranie. Snímač nevyžaduje žiadne ďalšie riadiace jednotky, zosilňovače alebo PC. S hmotnosťou menej ako 1,5 kg je ideálny na montáž do stiesnených priestorov a na robotické ramená. Gocator má vstavané GUI, ktoré umožňuje flexibilnú konfiguráciu, nastavenie profilometra a nástroje na meranie pomocou ľubovoľného prehliadača a operačného systému. Všetky údaje o Gocatore som čerpala z jeho používateľskej príručky.

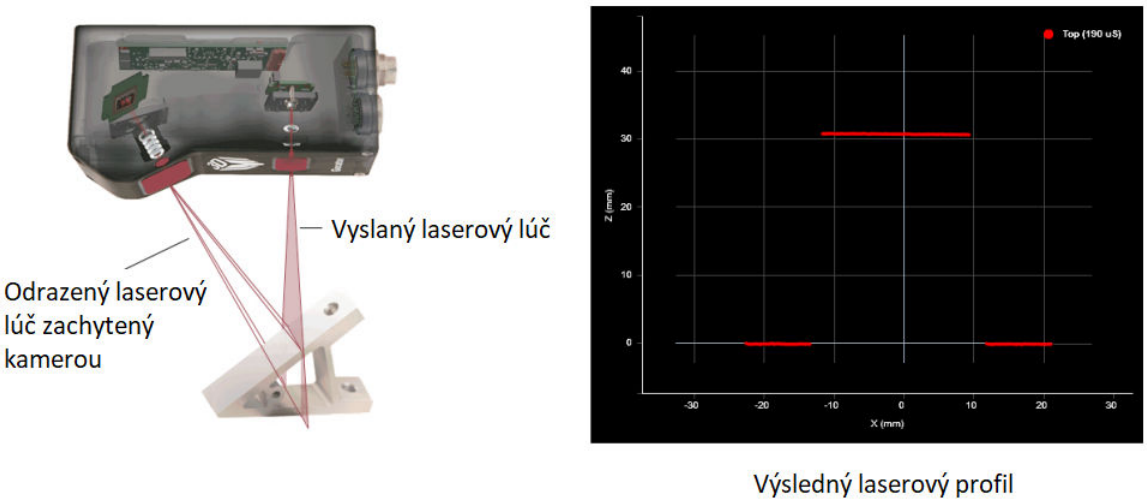
Časti Gocatora:

- Kamera - pozoruje laserové svetlo odrazené od cieľových plôch.
- Laser Emitter - vysiela štruktúrované svetelné lúče pre laserové profilovanie.
- I/O konektor - prijíma vstupné a výstupné signály.
- Napájancí/sieťový konektor - prijíma signály laseru a pripája sa k sieti.
- Indikátor napájania - svieti pri napájaní (modrá farba).
- Indikátor dosahu - rozsvieti sa, keď kamera detekuje laserové svetlo a nachádza sa v cieľovej oblasti (zelená farba).
- Indikátor lasera - svieti, keď je aktívny laser (oranžová farba).



Obr. 3.1: Časti Gocatora

Kamera senzora sleduje laserovú čiaru, ktorá je namierená na objekt a zachytáva odrazy lúčov laseru od objektu. Kamera zaznamená jeden 3D profil objektu pre každú polohu kamery. Dá sa povedať, že zachytáva rezy profilu. Odrazený laserový lúč dopadá na kameru z rôznych pozícií a uhlov v závislosti od vzdialenosti objektu a senzora. Vyžarovač laserových lúčov, kamera a snímaná časť objektu tvoria trojuholník. Gocator pozná vzdialenosť medzi laserovým žiaričom a kamerou a dva uhly, vďaka ktorým dokáže vypočítať vzdialenosť žiariča od objektu. Táto metóda výpočtu vzdialenosti sa volá laserová triangulácia.



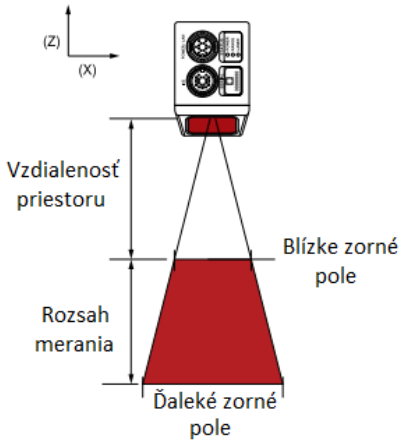
Výsledný laserový profil

Obr. 3.2: Spôsob fungovania Gocatora

V našom prípade bude snímač pripevnený na pohybujúcom sa aute. Cieľovými objektami budú značky na dráhe auta. Pri pohybe senzor vytvorí sériu 3D profilov, ktoré spolu tvoria kompletný sken objektu. Rýchlosť senzora a čas potrebný na zoskenovanie objektu je kritický pre systém.

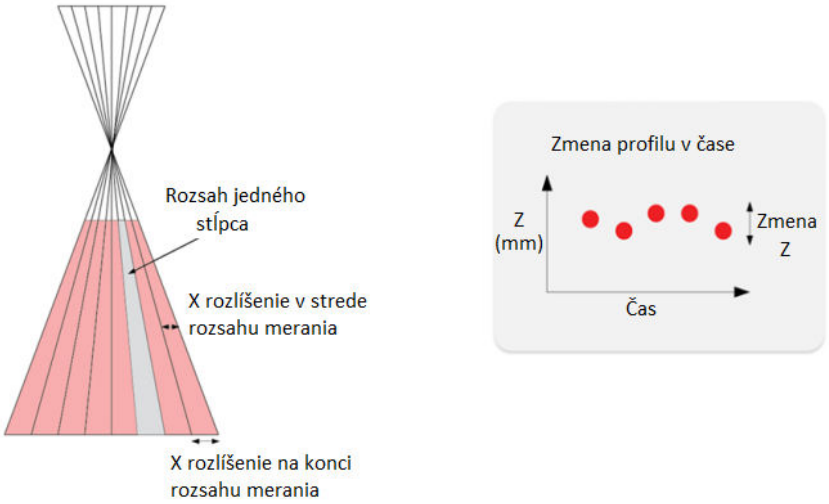
Pre správne pochopenie fungovania Gocatora je potrebné rozumieť aj týmto pojmom:

- Vzdialenosť priestoru - minimálna vzdialenosť objektu od snímača, aby mohol byť cieľ naskenovaný. Objekt nachádzajúci sa bližšie ako táto vzdialenoť sa nezosníma správne.
- Rozsah merania - hĺbka, v ktorej prístroj dokáže merať. Začína tam, kde končí vzdialenoť priestoru. V rozsahu merania by sa mal nachádzať celý cieľový objekt.
- Blízke zorné pole - šírka záberu na osi X, ktorá je v rámci rozsahu merania najbližšie k senzoru. V tejto oblasti je zorné pole menšie ale X rozlíšenie je vyššie.
- Ďaleké zorné pole - šírka záberu na osi X, ktorá je v rámci rozsahu merania najďalej od senzoru. V tejto oblasti je zorné pole väčšie ale X a Z rozlíšenia sú nižšie.



Obr. 3.3: Popis meraného priestoru

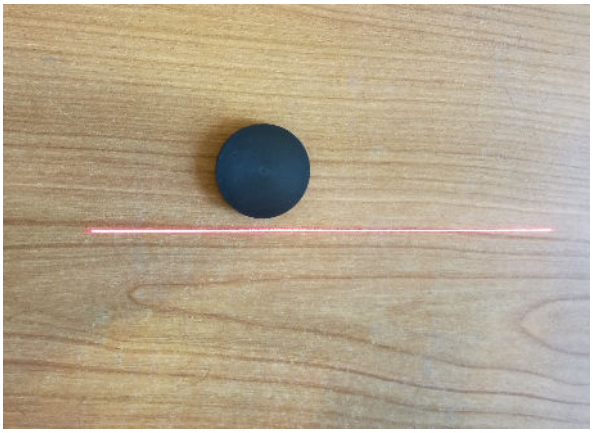
- X - rozlíšenie - horizontálne rozlíšenie pozdĺž skenovacej čiary. Vzdialenosť medzi bodmi na začiatku rozsahu merania je menšia ako na konci rozsahu merania (obr. 3.4).
- Z - rozlíšenie udáva nejmenši zistiteľný výškový rozdiel v každom bode, resp. aká je presnosť pri meraní výšky objektu. Čím bližšie sa náchadzajú objekt a senzor, tým je Z rozlíšenie vyššie.



Obr. 3.4: Rozlíšenie X a Y

Výstupom Gocatora je séria bodov, ktoré sú dané súradnicami X a Z. Rozsah merania sa nachádza pozdĺž osi Z. Čím bližšie je meraný objekt k snímaču tým výššia je hodnota Z. Zorné pole sa nachádza pozdĺž osi X, ktorej počiatok sa nachádza na priesečníku osi X a Z. Súradnicu Y treba dopočítať na základe vzdialenosti medzi skenovacími čiarami. Poloha Y sa zvyšuje, keď sa objekt posúva dopredu.

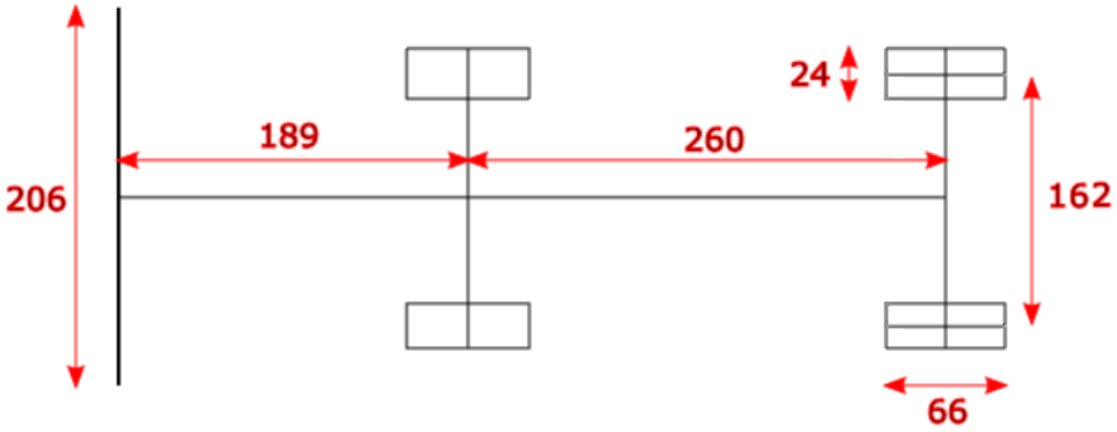
Profilometer je potrebné nakalibrovať tak, aby sa počiatok Z-ovej súradnice nachádzal vždy na povrchu vozovky. Pre tieto účely sa používa kalibračný disk (Obr. 3.5).



Obr. 3.5: Kalibračný disk

3.2 Vozidlo

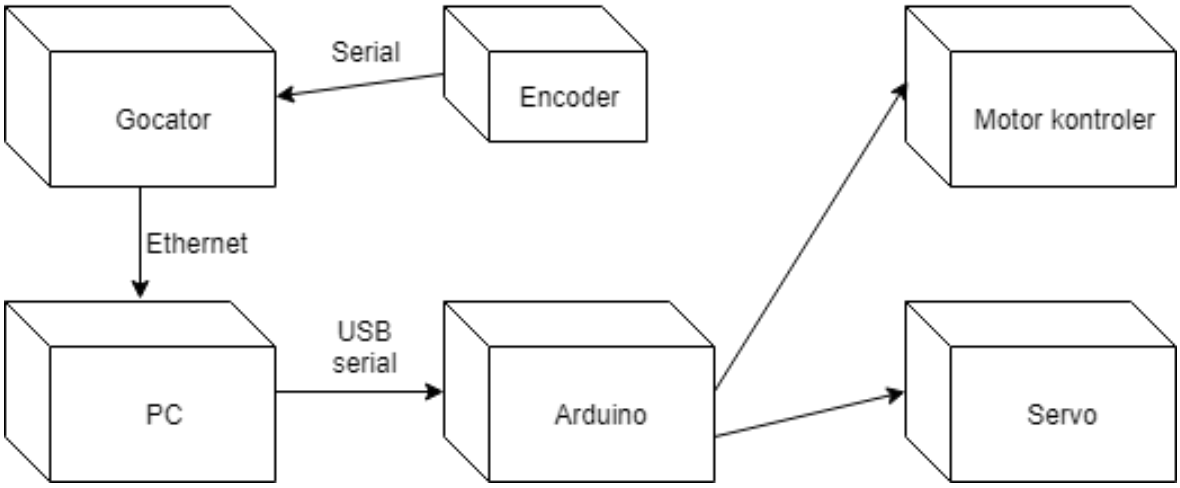
Projekt predpokladá, že v reálnej prevádzke sa budú používať vozidlá všetkých typov (osobné aj nákladné). V laboratóriu budeme na testovanie používať RC model auta s mierkou 1:10. Ide o diaľkovo ovládateľný model vozidla s maximálnou rýchlosťou 25 km/h. Vzdialenosť medzi osami náprav vozidla je 260 mm. Vzdialenosť medzi osami kolies na náprave 162 mm. Maximálny uhol vnútorného predného kolesa je 24° čo zodpovedá minimálnemu polomeru otáčania približne 665 mm. Poloha laserového snímača je 189 mm pred prednými kolesami. Dĺžka laserovej čiary je 206 mm.



Obr. 3.6: Rozmery vozidla v milimetroch

Použitý hardvér a jeho prepojenie popisuje obrázok 3.6:

- Encoder - sníma rýchlosť vozidla
- Gocator - zariadenie na meranie profilov
- PC - počítač, na ktorom bežia výpočty
- Arduino - elektronická platforma pre spojenie digitálneho a fyzického sveta
- Servomotor - nastavuje polohu kolies
- Motor kontrolér - ovládanie motora



Obr. 3.7: Deployment diagram

Snímanie aktuálnej rýchlosti encoderom by malo v ideálnom prípade poslúžiť presnejšiemu určeniu polohy skenovacej čiary na osi Y. Udaje o rýchlosti sú zaznamenávané s určitým oneskorením, čoho výsledkom je zavádzajúca neaktuálna informácia o prejdenej vzdialenosti. Taktiež nevieme, či budú mať cieľové vozidlá nainštalované encoder a ak aj áno, nepoznáme jeho presnosť a aktuálnosť. Z tohto dôvodu padlo rozhodnutie vytvoriť systém, ktorý bude nezávislý od encodera.

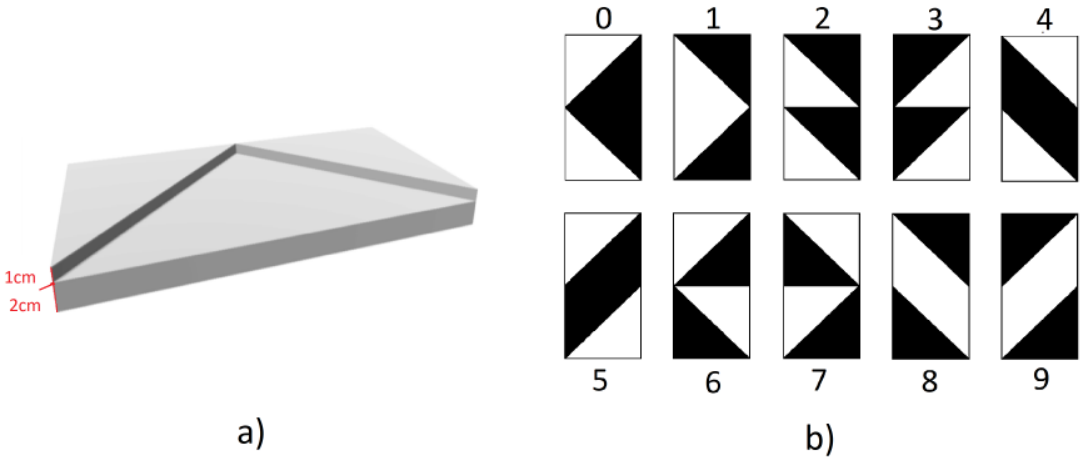
Obr. 3.8: Prototyp vozidla

4. Návrh riešenia pre rozpoznávanie značiek

Pri vývoji a testovaní bol použitý model auta s mierkou 1:10. Všetky parametre popisujúce veľkosť vozidla sú udávané v mierke tohto modelu.

4.1 3D vodiace značky

Vozidlo vykonáva určité úkony len na lokálnej úrovni. Ide hlavne o udržiavanie správnej polohy vozidla vzhľadom na vozovku. Pre účely lokálnej navigácie je vytvorená séria 3D vodiacich značiek, ktoré budú tvoriť zdroj informácií o charaktere vozovky. Tvar značiek je prispôsobený tak, aby ich bolo možné skenovať a rozpoznávať čo najjednoduchšie a najrýchlešie. Značka je obdĺžnikového tvaru s veľkosťou 2 cm $\times$ 4 cm. Je tvorená podstavou s výškou 2 mm, na ktorej sa nachádza konkrétny tvar značky s výškou 1 mm (obr 4.1 a). Tvar značky predstavujú dva trojuholníky, ktoré vypĺňajú plochu značky. Rôznym zoradením a otočením týchto trojuholníkov získavame sériu 10 značiek označených číslami od 0 po 9 (obr 4.1 b). Každá zo značiek je jednoznačne rozlíšiteľná od ostatných bez ohľadu na otočenie samotnej značky v priestore.



Obr. 4.1: Navrhnuté tvary vodiacich značiek.

4.2 Virtuálna mapa

Vozidlo má pri riadení k dispozícii virtuálnu mapu priestoru, ktorá obsahuje všetky dostupné informácie o trasách jazdy a ich parametroch. Mapa pozostáva z niekoľkých segmentov, ktoré spolu tvoria ucelenú trasu vhodnú pre pohyb vozidla. Každý segment je opísaný množinou atribútov:

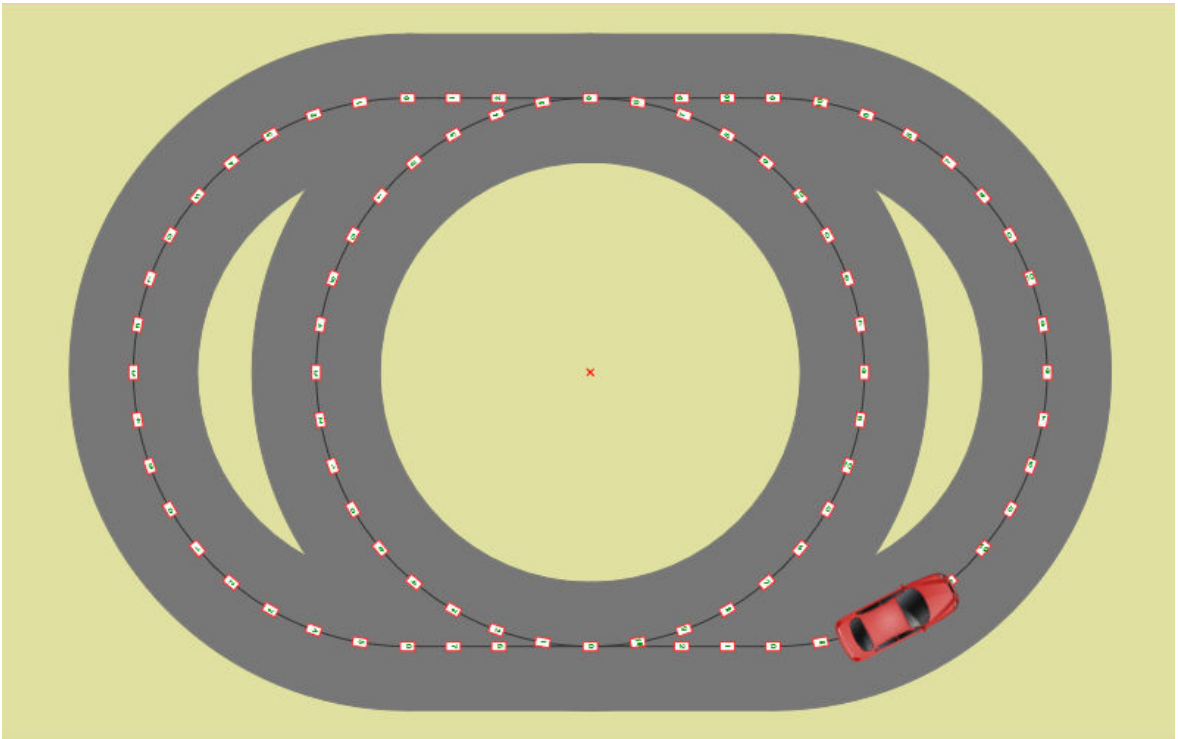
- jedinečný identifikátor
- absolútne súradnice začiatku segmentu
- zoznam identifikátorov segmentov, ktoré sú pripojené k začiatku segmentu
- absolútne súradnice konca segmentu
- zoznam identifikátorov segmentov, ktoré sú pripojené ku koncu segmentu
- tvar segmentu
- šírka segmentu
- možné smery jazdy
- zoznam značiek
- zoznam križovatiek a ich pozícia v rámci segmentu
- typ povrchu jazdnej plochy
- okraje cesty
- maximálna rýchlosť jazdy

- minimálna rýchlosť jazdy
- odporúčaná rýchlosť jazdy na začiatku segmentu

Tvary značiek kódujú rôzne úkony na ceste z nasledujúcej množiny:

- rovný pruh
- zakrivený pruh (polomer a stred otáčania)
- prechod
- typ prechodu (X, T, L atď.)
- krátkodobé parkovacie miesto
- dlhodobé parkovacie miesto
- miesto naloženia/vyloženia nákladu

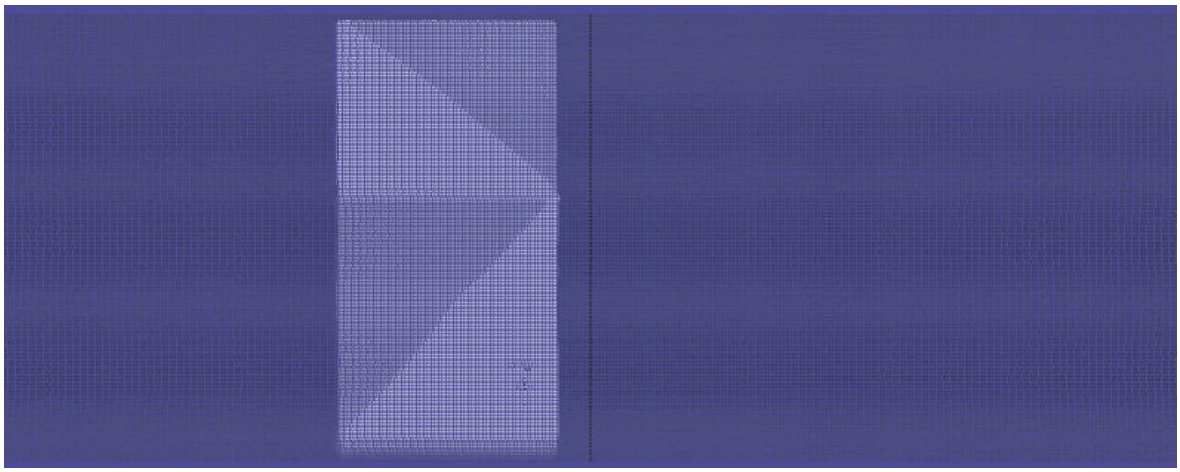
Parametre dráhy vytvorenej spojením viacerých segmentov sú uložené v súbore typu JSON. Ukážkový príklad JSONu pre experimentálny dopravný okruh v laboratóriu F1-114 sa nachádza v elektronickej prílohe práce. Na obrázku 4.2 je vizualizácia dráhy načítanej z JSON súboru.



Obr. 4.2: Virtuálna mapa

4.3 Dáta na vstupe

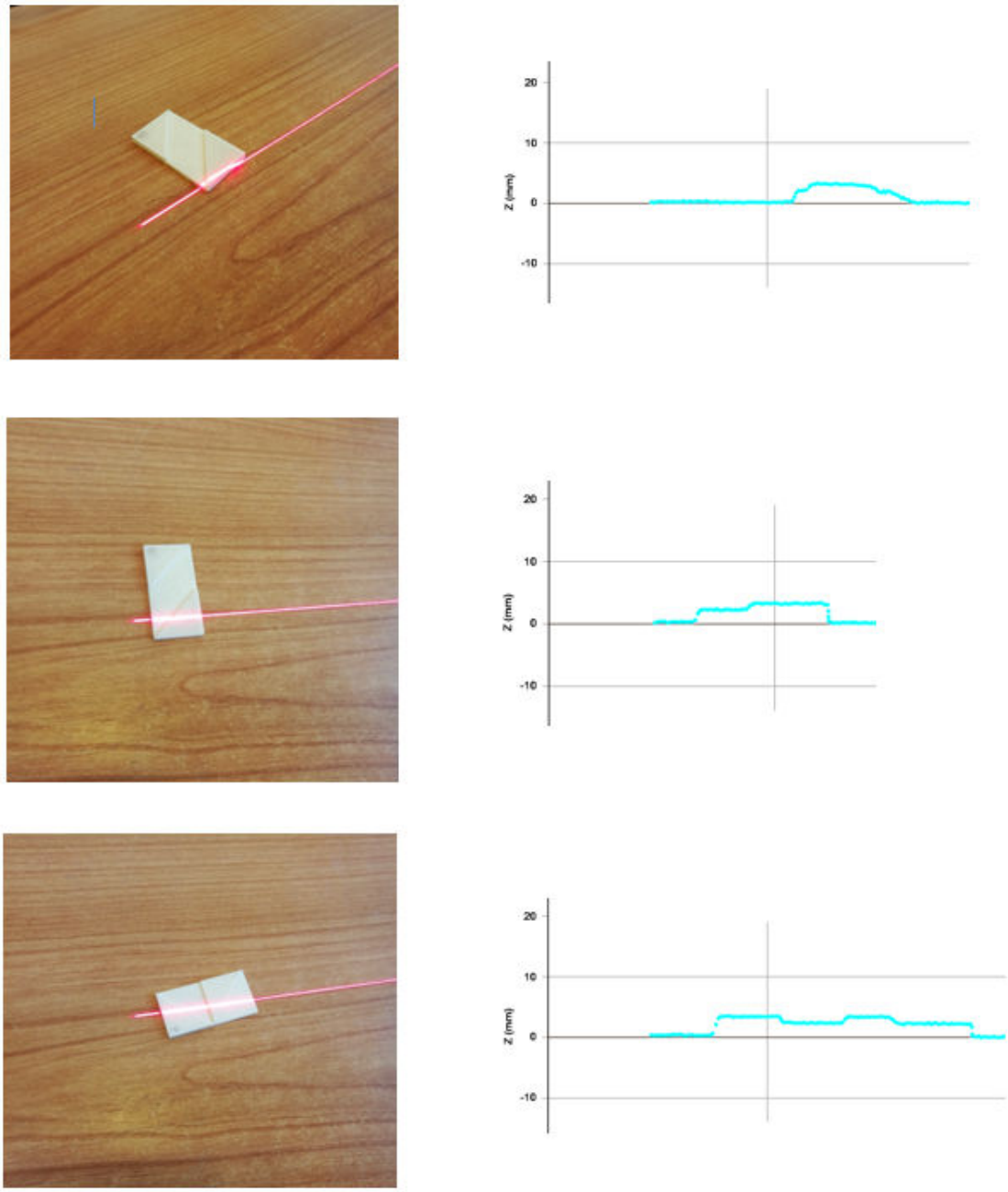
Výstupom laserového senzora je zhluk bodov v priestore. Pre každý bod poznáme jeho súradnice X a Z. Súradnica Y závisí od rýchlosti auta a hustoty skenovania. Hustota skenovania je určená snímkovacou frekvenciou Gocatora. Čím rýchlejšie sa autíčko pohybuje, tým menej skenovacích čiar stihne senzor na danej dráhe zachytiť. Na obrázku 4.3 je zachytený výstup senzora, pričom každý z bodov je vykreslený na svojej pozícii X a Y. Súradnica Z je reprezentovaná odtieňami šedej farby, kde čierne body majú najnižšiu hodnotu Z a biele najvyššiu. Modrá farba predstavuje priestor, kde nebol odmeraný žiadny bod. Čierna čiara v strede znázorňuje os senzora.



Obr. 4.3: Vizualizácia surových dát z Gocatora.

Keďže senzor vzdialenosti nemeria z pozície kolmo na značku, ale pod určitým uhlom, vznikajú na značke deformácie. Spodná strana značky pôsobí rozmazane, pretože skener pri skenovaní pod uhlom zachytil bočnú hranu značky.

Na obrázku 4.4 sú značky merané na rôznych pozíciách a k nim prislúchajúce výstupy z profilometra. Ak laserová čiara prechádza cez hranu, získaný profil je zaoblený, čo je potrebné ošetriť v algoritme rozpoznávania.

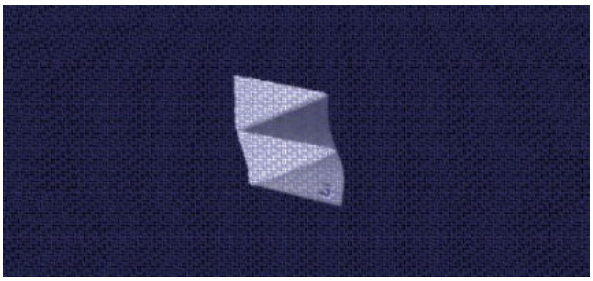


Obr. 4.4: Porovnanie výstupu Gocatora s realitou

4.4 Požiadavky na algoritmus

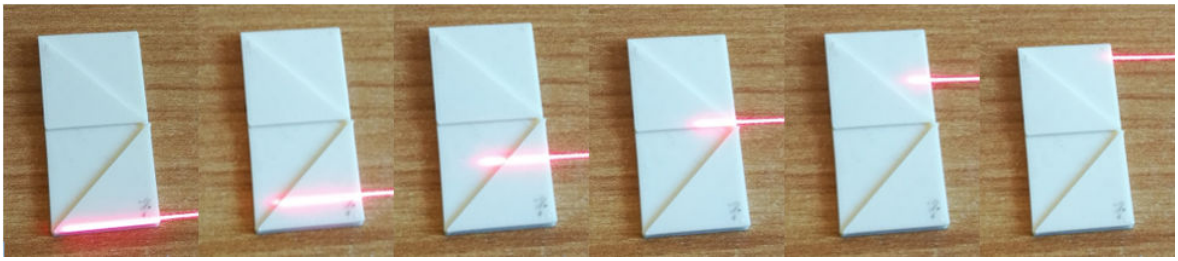
Prvou požiadavkou je čo najväčšia rýchlosť čítania a rozpoznávania. Vozidlo musí pohotovo reagovať vo všetkých situáciách a rýchlostiach tak, aby jazdilo plynulo a nijakým spôsobom neohrozovalo svoje okolie. Druhou dôležitou požiadavkou je robustnosť al-

goritmu. Spoľahlivosť rozpoznávania značiek v akýchkoľvek podmienkach je potrebné maximalizovať. Tu je hlavným problémom šum a deformácia značky spôsobená vplyvmi okolia. Konkrétne ide o zošikmenie značky spôsobené otáčaním auta so senzorom v zákrute a zvlnenie značky spôsobené nakláňaním sa auta vpred a vzad počas jazdy.



Obr. 4.5: Vizualizácia zvlnenej a zošikmenej značky

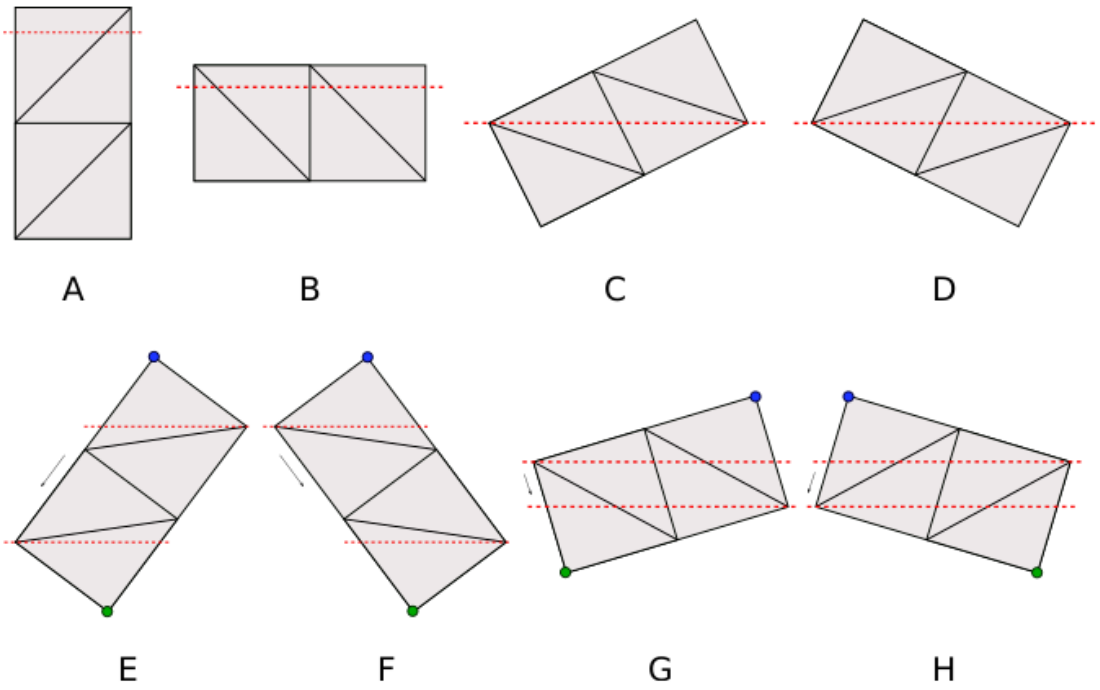
Na obrázku 4.6 sú zobrazené postupné fázy priechodu lúča cez značku počas pohybu vozidla tak, že na začiatku je lúč zarovnaný s dolnou stranou obdĺžnika značky a vozidlo zatáča vpravo po kruhovom oblúku. Na obrázku je vidno, že prechádzajúc po dĺžke značky sa ľavý okraj lúča presunul z ľavého okraja značky na pravý okraj značky. Tomu zodpovedá skreslenie súradnicovej sústavy nasnímanej značky, kvôli ktorému nie je možné pri kruhovom pohybe vozidla využiť bitmapové rozpoznávanie značiek, ktoré funguje spoľahlivo iba počas priameho pohybu vozidla.



Obr. 4.6: Pozícia lasera vzhľadom na značku v rôznych fázach zákruty

Obrázok 4.7 zobrazuje rôzne prípady nábehu vozidla na značku. Algoritmus musí byť schopný spoľahlivo rozpoznať značku vo všetkých okrajových prípadoch. Červená prerušovaná čiara znázorňuje šírku značky, ako ju systém vníma pri rozličných naklo-neniach značky. Dvojica prerušovaných čiar vymedzuje stredovú časť značky, kde je jej šírka maximálna a dá sa použiť na určenie naklonenia značky. Zachytená šírka sa pohybuje v intervale (w, h), kde w je bežná šírka značky (prípad A) a h je bežná výška značky (prípad B). Ak naklonenie značky nie je prípadom A alebo B tak rozlišujeme

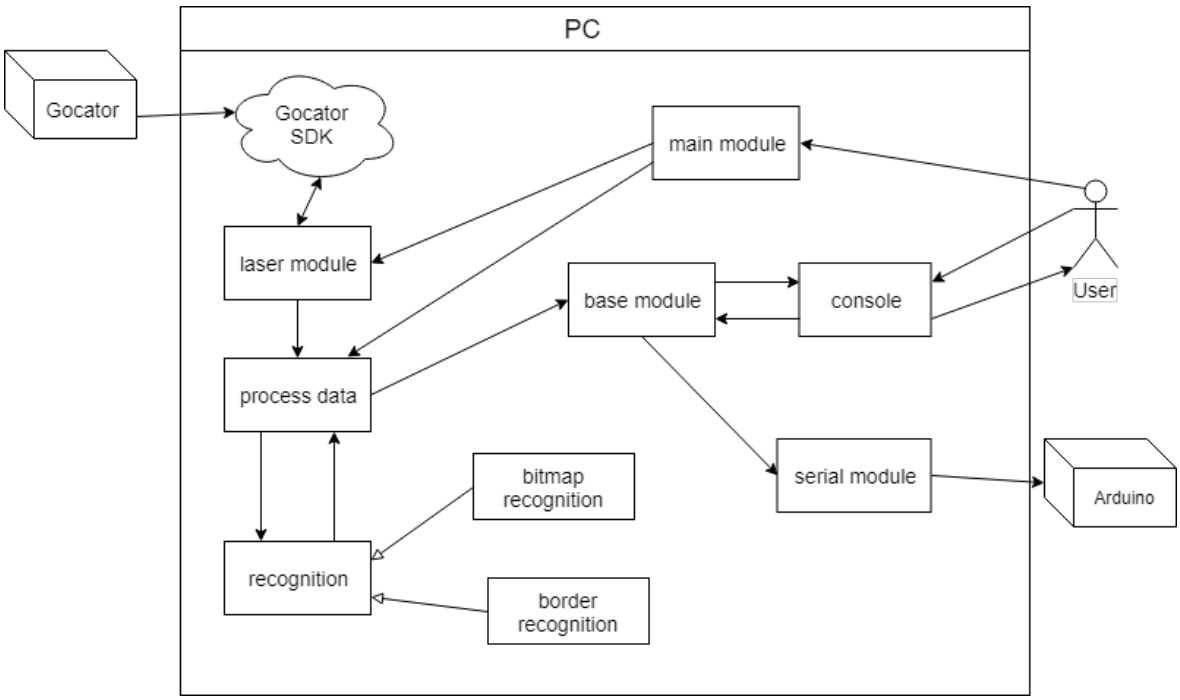
štyri možnosti, ako je značka položená (prípady E, F, G, H). Rozlíšime ich podľa usporiadania x-ových súradníc horného a dolného rohu (znázornené ako zelené a modré vrcholy) a podľa znamienka derivácie x-ovej súradnice v strede značky (označené malou šípkou na ľavej strane značky). Výsledkom je uhol naklonenia značky v rozmedzí $-\pi / 2, \pi / 2$.



Obr. 4.7: Rôzne naklonenia značky vzľadom na senzor

4.5 Architektúra softvérového riešenia

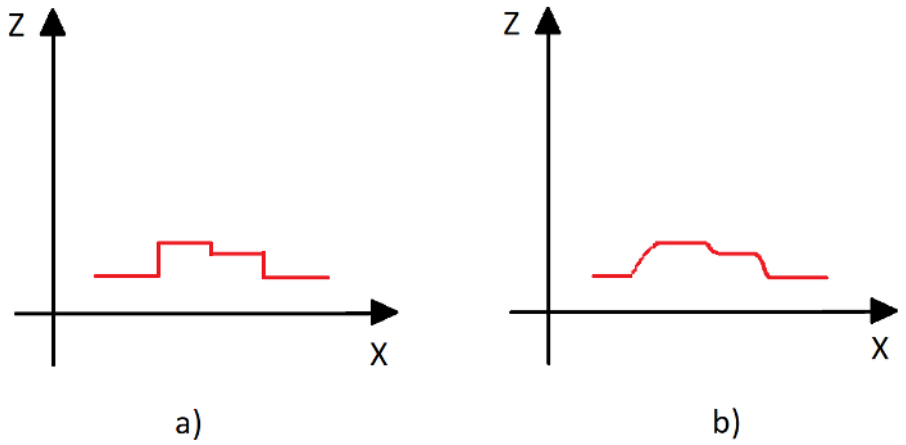
Programová časť je napísaná v jazyku C. Je rozdelená na viacero modulov, ktoré popisuje obrázok 5.6:



Obr. 4.8: Moduly softvérového riešenia

- Gocator SDK - knižnice pre programový prístup a kontrolu Gocator senzoru
- laser_module - získava okamžité informácie o naskenovaných dátach z SDK Gocatora zavolaním callback funkcie. Jeden packet s dátami môže obsahovať aj viac riadkových skenov. Pre každý riadkový sken sa vyberú body a uložia sa do buffera. Načítané riadky sa ďalej spracovávajú v module process_data. Dáta o pohybe vozidla získané z encoderu sa tu prepošlú priamo do base_module pre arduino, kde je možné ich využiť pri riadení rýchlosti.
- process_data - z celých riadkových skenov vyberie body, ktoré sa pravdepodobne nachádzajú na vnútorných a vonkajších hranách značiek. Hrany sa určujú na základe okamžitej z-ovej zmeny v profile o 1, 2 alebo 3 mm. Výstupom je pole bodov, ktoré sa nachádzajú na hranách značky resp. ich súradnice X a Z. Z je diskrétna hodnota 0, 2 alebo 3 podľa toho, o aký typ hrany sa jedná. Algoritmus predpokladá, že okraj lúča je na rovnej vozovke v úrovni 0 a potom každý skok o očakávanú zmenu 1, 2 alebo 3 spôsobí zmenu aktuálnej výšky. Pre reálnu prevádzku by bolo potrebné algoritmus zovšeobecniť a od tohto predpokladu upus-

tiť. Uhol pohľadu kamery, nedokonalé vyhotovenie značky a vysoká vzorkovacia frekvencia spôsobujú, že nábežné hrany identifikátorov nie sú bodové (ostré). Obrázok 4.9 zobrazuje skutočný profil značky a k nemu prislúchajúci deformovaný profil získaný zo senzora. Algoritmus sa musí s takýmto postupným nábehom vysporiadať. Zároveň musí byť robustný vzhľadom na zašumené hodnoty.



Obr. 4.9: a) skutočný tvar značky, b) deformovaný výstup senzora

Algoritmus udržuje dva priemery niekoľkých za sebou idúcich bodov pred pomyselným schodom a za ním. Sleduje Z-ovú súradnicu reprezentantov týchto priemerov. Hľadá medzi nimi schodík minimálnej požadovanej výšky. Keďže šírka schodu sa pri rôznych uhloch líši, je potrebné zistiť, kde schod končí - preto vyhľadáva najbližšiu rovnú plochu (keď sa bod od priemeru svojho okolia už skoro nelíši), pričom zohľadňujeme aj celkové naklonenie lúča pre prípad, že vozidlo je naklonené v zákrute.

Následne algoritmus nájde takú x-ovú súradnicu medzi dvoma bodmi, ktoré vymedzujú schod, kde je z-ová súradnica najviac podobná priemeru z-ových súradníc bodov vymedzujúcich schod.

Do zoznamu nájdených bodov sa pridá nový bod oklasifikovaný zmenou z-ovej súradnice 0, 1, 2 alebo 3.

Po každom nájdenom bode sa algoritmus posunie do pravého vymedzujúceho bodu, aby ten istý bod nenašiel duplicitne.

V prípade, že algoritmus usúdi, že ďalšie riadkové skeny už nepatria k aktuálne načítavanej značke, čiže nazbieral všetky dáta (body), ktoré boli k dispozícii,

množinu zaznamenaných bodov prepošle do modulu rozpozania identifikátora.

- recognition - počas vývoja sme navrhli dve verzie rozpoznávania. Obidve sa snažia rozpoznať značku bez potreby presnej Y-ovej súradnice bodov získavanej z enkodéra. Jedna pri rozpoznávaní využíva bitmapy a druhá využíva hrany identifikátora.
- base_module - má na starosti komunikáciu s dolnou vrstvou, ktorá beží na platforme Arduino a poskytuje rozhranie na riadenie vozidla. Okrem toho preposiela údaje z enkodéra pripojeného ku Gocatoru do Arduina pre prípadné riadenie rýchlosti so spätnou väzbou a obsahuje hlavnú riadiacu procedúru, ktorá vypočítava riadiaci signál pre servo riadenia natáčajúce prednú nápravu vľavo/vpravo.

  Riadenie sa realizuje modifikovaným PD-kontrolerom (proportional-derivative), ktorého konštanty sme našli empirickými testami:

  $$\beta = 65 \cdot sgn(e_{offset}) \cdot \sqrt{e_{offset}} + 60 \cdot \Delta e_{offset} + 40 \cdot e_{\alpha}$$

  kde e_{offset} je chyba x-ovej pozície rozpoznanej značky oproti očakávanej x-ovej pozícii (v strede riadkového skenu), Δe_{offset} je zmena tejto chyby oproti chybe na predchádzajúcej značke a e_{α} je chyba natočenia rozpoznanej značky oproti očakávanému (nulovému natočeniu).

  Tento modul má vlastné výpočtové vlákno kvôli blokujúcej komunikácii cez sériový port, ktorá prebieha výmenou paketov na úrovni vlastného aplikačného protokolu. Celý protokol je popísaný v zdrojovom kóde pre nízkoúrovňový modul na platforme Arduino (base.info).
- serial_module - zabezpečuje komunikáciu s arduinom cez virtuálny sériový port odoslaním alebo prijatím poľa bajtov, obsahuje platformovo-závislé funkcie na komunikáciu cez sériový port.
- console - obsahuje vlastné výpočtové vlákno, ktoré len načítava zo štandardného vstupu, v prípade požiadavky používateľa na ukončenie (klávesa 'q'), alebo zastavenie motorov (klávesa 'x') príslušnú požiadavku realizuje.
- main_module - má na starosti odštartovanie a inicializovanie všetkých modulov a ich postupné povypínanie pri ukončovaní aplikácie

5. Rozpoznávanie značiek

Prvým krokom v oboch algoritmoch je nájsť v mračne bodov značku a vyselektovať jej hrany. Na skenovacích čiarach nájdeme miesta, kde sa mení výška resp. hľadáme výškový skok, ktorý pripomína hranu značky. Typy skokov:

- 0 mm → 2 mm
- 0 mm → 3 mm
- 2 mm → 3 mm
- 3 mm → 2 mm
- 3 mm → 0 mm
- 2 mm → 0 mm

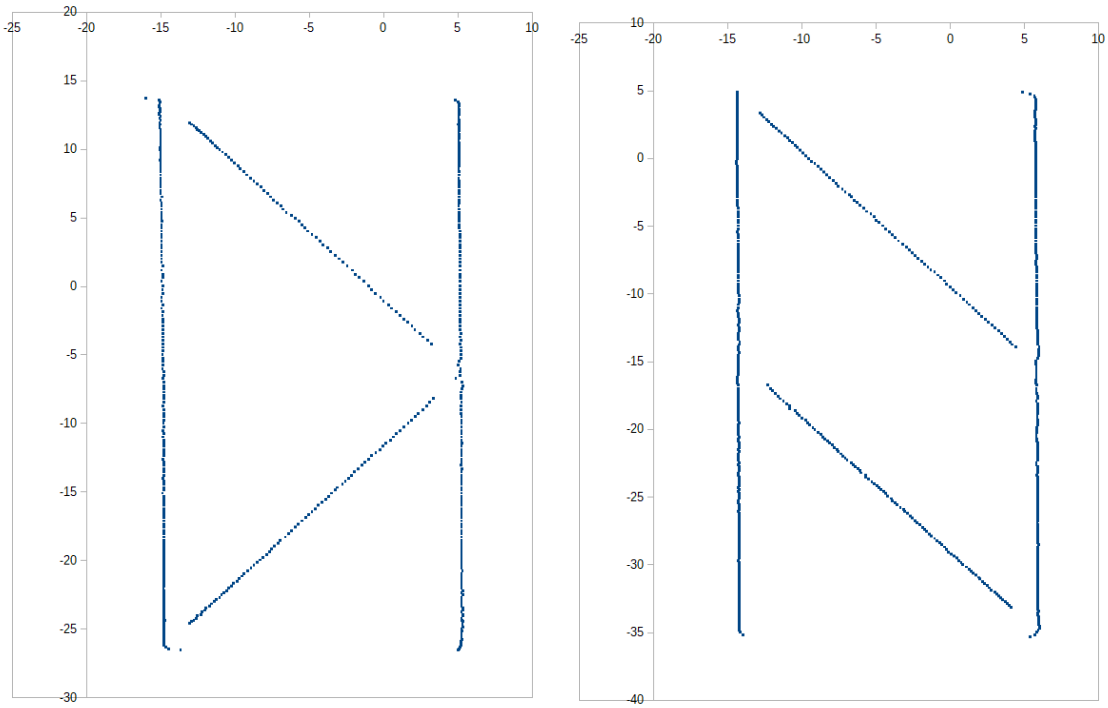
Výstupom prvej fázy je dátová štruktúra so zoznamom všetkých relevantných bodov, na ktorých dochádza ku výškovému skoku, a zoznam riadkových skenov obsahujúcich indexy začiatočných bodov na každom riadku. Použité štruktúry:

```
typedef struct
{
   double x;  // x-coordinate in [mm] - position along laser line
   double y;  // from encoder
   int ln;    // scanline number (can be used as y, if encoder not used)
   int level[2]; // step specification (i.e. 0,3 or 0,2 or 3,2 etc.)
} IdPoint;

typedef struct
{
   int point_index; // index to array PointsFound[]
   double width_mm; // = right_edge - left_edge
   double left_edge; // in mm from scanline start
   double right_edge; // in mm from scanline start
} ScanLine;

extern IdPoint PointsFound[MAX_POINTS_PER_TAG];
extern ScanLine lines[MAX_LINES_PER_TAG];
```

Príklady vizualizácie naskenovaných schodových bodov pre značky č.6 a č.9 sú na obr. 5.1.



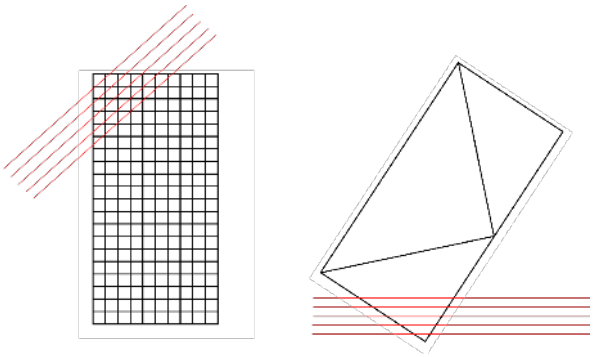
Obr. 5.1: Nájdené schodové body v dvoch skenovaných značkách

5.1 Algoritmus s využitím bitmáp

Rozpoznávanie identifikátorov pomocou bitmapy pracuje s mriežkovými štruktúrami. Dáta získané s Gocatora zapíše do bitmapovej mriežky 10x20 bodov, ktorú následne porovnáva so vzorovými bitmapami 11-tich značiek. Zo vzorových značiek sa vyberie ten, ktorý má v porovnaní s naskenovanou bitmapou najmenšiu chybu. ID tejto nájdenej značky je výstupom algoritmu rozpoznávania.

Aby bolo možné riadkové skeny do mriežky zakresľovať, je potrebné najskôr odhadnúť orientáciu značky (jej naklonenie oproti vzpriamenej polohe) a z tohto naklonenia určiť naškálovanie na osi y.

Ak je škálovanie na osi y a sklon známy, pre každý riadkový sken vieme presne akým spôsobom 'krája' vytváranú mriežku. Všetky preťaté body mriežky započítavajú aktuálnu z-ovú súradnicu z každého pretínajúceho riadkového skenu do príslušnej bunky mriežky (obr. 5.2).



Obr. 5.2: Vypĺňanie mriežky dátami zo senzora

1. Odstránenia sa riadky s výbežkami. Ak ľavý alebo pravý koniec riadku nie je ani na očakavánej x-ovej súradnici vypočítanej z ľavých koncov predchádzajúcich dvoch riadkov ani na x-ovej súradnici vypočítanej z ľavých koncov nasledujúcich dvoch riadkov, alebo analogicky nesedí x-ová súradnica pravého konca, celý riadkový sken je vyradený zo zoznamu

2. Skontroluje sa, či ľavá hrana obsahuje roh, alebo nie. Ak ľavá hrana neobsahuje roh, tak sa značka sníma priamo a nebude potrebná rotácia lúčov, inak je potrebné určiť uhol naklonenia identifikátora.

3. V prípade, že značka nie je naklonená overí sa, či je natočená priamo, alebo je o 90 stupňov otočená (ležato). Ďalej sa zisťuje, či šírka zodpovedá normovanej šírke a výške značky s určitou toleranciou, ak nie, ide o falošnú značku.

4. Ak je značka naklonená, hľadá sa index riadku, na ktorom je zlomový bod (roh) na ľavej (resp. pravej) hrane. Vypočíta sa rozdiel x-ových súradníc hrán každých dvoch za sebou idúcich riadkov. Na rohu značky sa mení znamienko týchto rozdielov. Potom sa nájde taký index R, kde absolútna hodnota rozdielu U-V je maximálna, pričom

   - U = je súčet rozdielov x-ových súradníc ľavej (resp. pravej) hrany pre riadky 1..R
   - V = je súčet rozdielov x-ových súradníc ľavej (resp. pravej) hrany pre riadky R..N

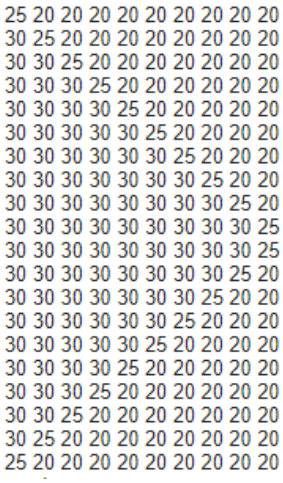
   Zjednodušene povedané nájde sa bod, kde sa kladné hodnoty rozdielov menia na záporné alebo naopak. R je najlepší možný zlomový bod, čiže zodpovedá rohu na ľavej, resp. pravej hrany.

5. Znamienko naklonenia určíme podľa toho, ktorý roh je vyššie. Pre nejednoznačné prípady rozlíšime naklonenie značky podľa toho, či x-ová súradnica ľavej hrany

stredných riadkov má vzostupný alebo zostupný trend. Po určení uhla naklonenia vypočítame očakávanú celkovú výšku projekcie značky v riadkových skenoch a z nej aj riadkovú rýchlosť mm/riadkový sken.

6. Riadky sa vhodne preškálujú a Z-ové hodnoty sa uložia do pripravenej mriežky. Výsledok bude vyzerať ako na obrázku 5.3:

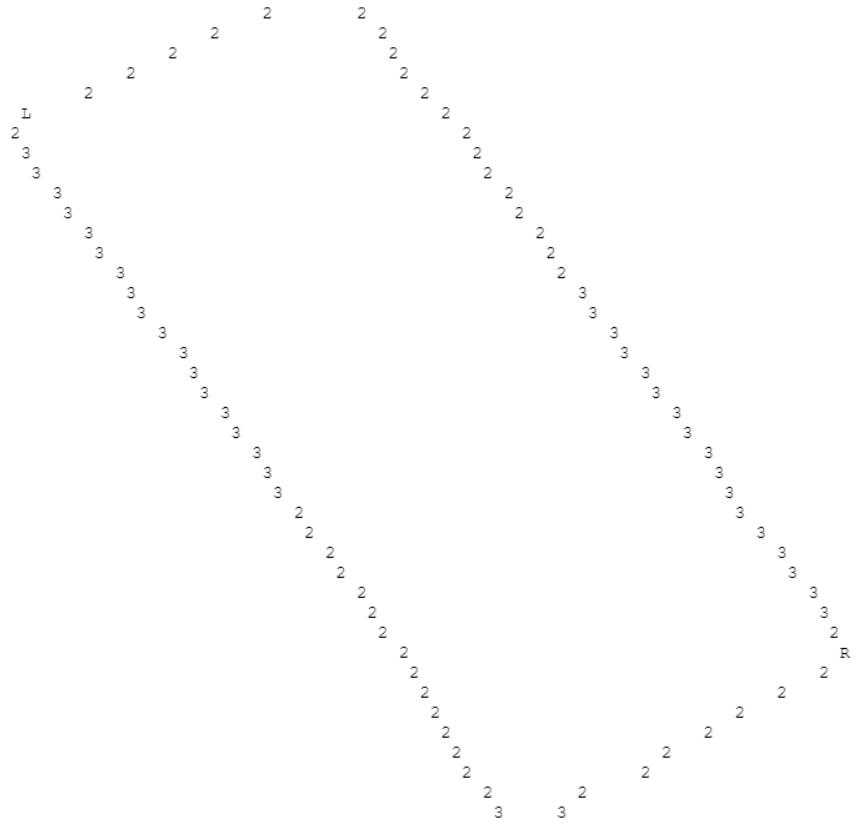
   - 20 - plocha vo výške 2 mm
   - 30 - plocha vo výške 3 mm
   - 25 - hrana



Obr. 5.3: Mriežka bitmapy s hodnotami

5.2 Algoritmus s využitím hrán

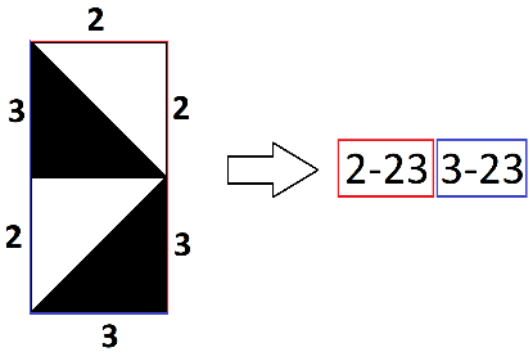
Algoritmus využíva schodové body z predchádzajúcej fázy iba pre vonkajšie okraje značky. Body vnútorných uhlopriečok nie sú potrebné, pretože z okrajových hrán priamo jednoznačne vyplýva typ značky.



Obr. 5.4: Vyselektované hranové body

Obrázok 5.4 zobrazuje výsledok selekcie hranových bodov. Čísla 2 zobrazujú hrany s výškou 2 mm a čísla 3 zobrazujú hrany s výškou 3 mm. Algoritmus spracováva dáta zo senzora vo forme bodov, ktoré tvoria úsečky. Algoritmus pre ne nepotrebuje samostatnú dátovú štruktúru, reprezentuje ich implicitne.

Hranový algoritmus je založený na jednoduchej, elegantnej, výpočtovo veľmi efektívnej, robustnej a flexibilnej metóde. Uvážme, aké charakteristické vlastnosti značiek nám ich pomôžu od seba jednoznačne rozlíšiť. Zoberieme do úvahy iba okrajové hrany značky a označme si ich číslami 2 a 3 podľa toho, či daná hrana tvorí prechod 0 mm $\rightarrow$ 2 mm alebo 0 mm $\rightarrow$ 3 mm. Čísla hrán spolu tvoria postupnosť čísel, ktorá je pre každú značku unikátna. (obr 5.5) Postupnosť je vytvorená výberom čísel hrán v smere hodinových ručičiek. Kód je rozdelený na dve časti. Prvá je zápisom hornej a pravej hrany a druhá spodnej a ľavej hrany.



Obr. 5.5: Kódovanie značky

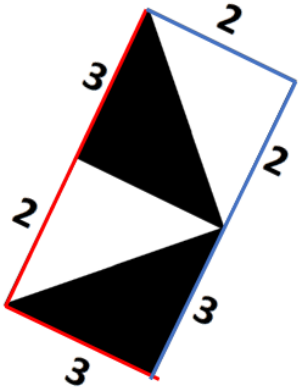
Charakteristiku všetkých typov značiek 0-10 môžeme ľahko reprezentovať dvojrozmerným poľom. Algoritmus vie rozpoznať aj špeciálnu značku s kódom 10, ktorá nemá uhlopriečky ale zvislú úsečku v strede.

```
static border_models[NUMBER_OF_TAGS][4] = {
          { 2, 22, 2, 33 },
          { 3, 33, 3, 22 },
          { 2, 33, 3, 22 },
          { 2, 22, 3, 33 },
          { 2, 32, 2, 32 },
          { 2, 23, 2, 23 },
          { 3, 23, 2, 23 },
          { 3, 32, 2, 32 },
          { 3, 23, 3, 23 },
          { 3, 32, 3, 32 },
          { 32, 22, 23, 33 } };
```

Algoritmus je robustný voči šumu, keďže rozhodnutie o type úsečky robí väčšinovým hlasovaním bodov. Vďaka tomu si vie poradiť aj s niekoľkými nesprávne prečítanými schodovými bodmi. Platné body teda svojím počtom vyradia zle zosnímané body spôsobené šumom. Riešime dva typy situácií, ktoré nastávajú pri rôznych natočeniach značky:

1. Značka je pootočená - predpokladá sa, že vo väčšine prípadov, keď sa vozidlo nepohybuje po priamke nastane táto situácia. Vtedy je prechod auta po vozovke dokonale kolmo na značku málo pravdepodobný. V tomto prípade zoberieme najprv všetky hranové body z pravej strany značky, čiže všetky prechodové body x mm $\rightarrow$ 0 mm. Celú pravú stranu rozdelíme pomocou pravého horného vrcholu na dve časti a vzniknuté segmenty porovnáme so vzormi značiek. Rovnako postupujeme aj pre ľavú stranu. Vrchol značky sa nachádza v bode, kde nastáva zmena

smeru hrany. Pre značku z obrázku 5.6 bude úvaha nasledovná: Pravú stranu značky rozdelíme pravým horným rohom na dva segmenty. V prvom segmente sa nachádzajú len prechody z 2 mm do 0 mm. V druhom segmente sa nachádzajú prechody z 2 mm do 0 mm a z 3 mm do 0 mm približne pol na pol. Vieme preto povedať, že pravá strana bude mať kód 2-23. Podobný postup zopakujeme aj pre ľavú stranu.



Obr. 5.6: Príklad značky s označenými hranovými prechodmi. Zvlášť sa spracovávaju modré hrany a zvlášť červené

Problém môže nastať v situácii, keď sa na jednej strane značky nachádzajú len čísla 2 alebo 3. V tomto prípade nevieme určiť, či sa jedná o postupnosť 2-22 alebo 22-2. Riešením je porovnať, ktorá časť strany značky je dlhšia a ktorá kratšia. Toto može byť problém, ak je značka výrazne natiahnutá alebo podobne deformovaná. Je preto potrebné zistiť, o koľko je značka natiahnutá a preškálovať ju.

2. Značka je natočená priamo - v tomto prípade môžme použiť postup ako pri prvej situácii až na jednu výnimku. Značka je vzhľadom na snímač v polohe 'rovno' teda hrana značky je rovnobežná so skenovacou čiarou senzora. To znamená, že žiadna skenovacia čiara nepretne hornú a dolnú hranu značky. Riešením je zobrať niekoľko prvých a posledných skenovacích čiar a pomocou vnútornej hrany trojuholníka zistiť, aký je typ prechodu vrchnej a spodnej hrany.

5.3 Výsledky algoritmov rozpoznávania

Oba algoritmy sme najskôr testovali na jednotlivých značkách v umelých podmienkach, keď vozidlo aj značka boli položené na stole a vozidlom sme hýbali ručne. Po

dosiahnutí dostatočnej spoľahlivosti (všetky značky boli úspešne opakovane správne rozpoznávané) sme si zostrojili niekoľko testovacích dráh v Laboratóriu autonómnej mobility. Na obrázku 5.7 je zobrazená veľká testovacia dráha pozostávajúca z kružnice s priemerom 3 m a 5 m širokého oválu. Na testovanie priameho pohybu sme používali rovnú 8 m dráhu. Značky sú umiestňované každých 25 cm.

Obr. 5.7: Veľka testovacia dráha v Laboratóriu autonómnej mobility FMFI UK

Na zdokumentovanie funkcionality riešení uvádzame tri podrobné záznamy jázd:

1. priebeh jazdy algoritmu rozpoznávajúceho pomocou bitmáp na rovnej 5.5 m

dráhe,

2. jazda algoritmu rozpoznávajúceho pomocou hrán na 8 m rovnej dráhe,

3. jazda algoritmu rozpoznávajúceho pomocou hrán na kružnici vo veľkej testovacej dráhe.

Vo všetkých troch prípadoch boli všetky značky rozpoznané úspešne. V záznamoch sú uvedené odchýlky značky od stredného bodu riadkového skenu i zistené naklonenie značky. Tieto hodnoty sú použité na výpočet riadiaceho signálu pre servomotor vo funkcii update_direction(), pozri sekciu 4.5.

Záznam 1: rovná 5.5 m dráha, bitmapový algoritmus:

id	or	speed	c_x	error	confid	time[s]	lines	points	dir
8	1	0.0922	0.420	1.090	2.354	0.000	434	1208	1
2	2	0.1399	7.076	0.914	2.679	0.016	286	782	9
7	2	0.1556	14.021	0.900	2.514	0.000	257	715	12
2	2	0.1509	20.388	0.453	5.546	0.000	265	728	14
6	2	0.1695	27.531	0.888	2.963	0.000	236	659	16
9	1	0.1521	34.448	0.943	2.140	0.000	263	724	17
5	1	0.1527	37.553	0.344	7.259	0.000	262	728	17
8	1	0.1688	41.783	0.719	3.018	0.016	237	663	18
4	1	0.1619	47.294	0.426	5.698	0.016	247	695	19
5	1	0.1544	50.209	0.723	3.486	0.000	259	717	19
3	1	0.1544	52.487	0.693	3.162	0.000	259	715	19
8	1	0.1914	53.915	1.038	2.218	0.015	209	585	19
2	2	0.1556	55.563	1.434	1.811	0.015	257	704	20
3	2	0.1896	57.126	0.329	7.114	0.016	211	603	20
0	2	0.1878	57.344	0.309	7.484	0.000	213	597	20
7	2	0.2073	55.002	0.750	3.049	0.000	193	537	18
9	1	0.1653	51.346	1.049	1.929	0.000	242	664	17
7	2	0.1646	47.291	1.756	1.250	0.000	243	675	16
8	1	0.1413	41.838	0.756	3.197	0.000	283	788	15
4	1	0.1626	35.716	0.971	2.630	0.000	246	670	13
7	2	0.1653	28.382	1.448	1.626	0.016	242	669	11
3	2	0.1521	18.282	0.525	4.394	0.000	263	738	7
6	2	0.1688	7.837	0.644	3.781	0.000	237	664	3

Záznam 2: rovná 8 m dráha, hranový algoritmus:

id	or	tilt	c_x	time[s]	lines	points	dir
9	0	-0.178	-0.291	0.000	376	1031	-3
2	0	-0.514	1.212	0.000	267	732	7
9	0	-1.676	-1.618	0.000	213	589	-10
5	0	0.747	-3.348	0.000	208	570	-12
3	2	-3.467	-7.529	0.000	172	466	-22
8	0	0.308	-10.338	0.000	196	535	-21
4	0	0.040	-7.235	0.000	169	458	-15
7	2	2.352	-0.051	0.000	169	473	4
5	0	0.764	8.139	0.000	192	527	23
9	0	-2.416	4.554	0.000	204	548	9
8	0	-2.485	-6.656	0.000	207	567	-24
2	2	-0.503	-13.535	0.000	213	581	-27
3	2	0.152	-11.796	0.000	243	651	-20
6	2	0.675	-7.941	0.000	221	609	-15
5	0	3.459	0.195	0.000	216	587	9
8	0	-1.404	4.997	0.000	195	540	15
7	2	-0.531	4.899	0.000	201	558	13
4	0	-0.430	-2.551	0.000	223	600	-14
8	0	-1.302	-7.503	0.000	230	628	-21
7	2	0.980	-5.695	0.000	208	575	-13
6	2	-0.351	0.207	0.000	212	583	6
7	2	1.074	4.847	0.000	207	574	17
8	0	-1.960	0.172	0.000	211	577	-1
3	2	-1.909	-10.086	0.000	196	529	-27
2	2	0.230	-16.510	0.000	190	513	-29
4	0	1.036	-14.112	0.000	192	521	-21
9	0	0.252	-9.219	0.000	192	520	-16
8	0	1.296	-1.907	0.000	195	541	-3
2	2	3.476	7.775	0.000	190	525	25
3	0	-1.404	7.980	0.000	179	493	17
0	0	-4.476	-3.997	0.000	175	474	-22
1	2	-0.936	-12.664	0.000	177	482	-28

Záznam 3: kružnicová dráha s priemerom 3 m, hranový algoritmus:

id	or	tilt	c_x	time[s]	lines	points	dir
2	3	-1.704	42.664	0.000	530	1534	169
3	3	-0.319	42.818	0.000	315	768	169
4	1	0.335	46.756	0.000	260	732	184
5	1	-0.107	50.016	0.000	257	614	190
0	3	0.000	54.718	0.000	178	449	199
1	3	-1.163	54.497	0.000	262	677	195
2	3	-0.486	58.327	0.000	230	687	203
3	3	-0.919	52.889	0.000	224	553	190
4	1	-2.871	42.453	0.000	213	594	159
5	1	0.000	32.406	0.000	176	433	123
0	1	0.295	37.201	0.000	178	470	141
1	3	8.075	61.529	0.000	185	477	224
2	3	3.173	80.244	0.000	178	523	236
3	3	0.553	81.321	0.000	179	432	225
4	1	-1.141	79.750	0.000	179	507	221
5	1	-3.871	69.694	0.000	177	429	205
0	3	-3.524	52.221	0.000	179	458	180
6	3	1.127	42.516	0.000	157	418	162
7	1	0.000	32.001	0.000	110	284	122
8	1	9.502	37.698	0.000	169	483	149
9	1	1.875	58.732	0.000	175	418	215
5	1	-1.091	77.169	0.000	176	415	230
0	3	-0.770	76.966	0.000	189	488	219
6	3	-6.075	68.988	0.000	182	494	204
7	3	0.000	60.228	0.000	160	430	198
8	1	-6.373	39.427	0.000	182	512	128
9	1	3.328	35.801	0.000	145	348	135
5	1	3.230	62.269	0.000	175	415	223
0	3	3.346	87.209	0.000	163	417	245
6	3	0.621	85.688	0.000	181	490	227
7	3	-2.959	68.217	0.000	166	442	200
8	1	-4.336	49.536	0.000	180	515	174
9	1	0.733	36.630	0.000	163	385	130
5	1	1.110	42.468	0.000	194	467	171
0	3	5.249	58.809	0.000	200	530	215
1	3	2.473	68.094	0.016	222	570	219

```
2  3  0.435   70.556  0.000   206   596    216
```

Legenda:

- id - id rozpoznanej značky 0-10,
- or - orientácia značky 1 (dopredu) a 2 (dozadu),
- tilt - sklon značky,
- c_x - umiestnenie značky z pohľadu senzora ,
  - 0 = stred,
  - kladná hodnota = pravá strana,
  - záporná hodnota = ľavá strana,
- time[s] - čas rozpoznávania,
- lines - počet skenovacích čiar pretínajúcich značku,
- points - celkový počet rozpoznaných bodov na značke,
- dir - signál vyslaný riadeniu vozidla,
  - no = bez zmeny.
  - kladná hodnota = odboč vpravo,
  - záporná hodnota = odboč vľavo,

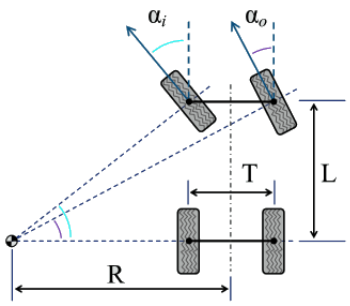
Za výborný výsledok považujeme, že hranový algoritmus pri jazde po 8 m dráhe nemá nikdy výchylku viac ako 20 mm od želanej trajektórie. V prípade pohybu po kružnici je želaná poloha značky 42 mm vpravo od stredu riadkového skenu. Z uvedených výsledkov vidieť, že oba algoritmy rozpoznávania značiek fungujú a boli úspešne integrované do celkového funkčného systému riadenia vozidla. Výhodou hranového algoritmu oproti iným riešeniam je hlavne robustnosť. Pri jazde vozidla v zákrute vznikajú v naskenovaných dátach značné deformácie, ktoré iným algoritmom znemožňujú rozpoznať značky korektne. Hranový algoritmus dosahuje úspešnosť rozpoznávania cez 95%, pričom je odolný voči deformáciám. Dosiahnutá rýchlosť rozpoznávania 0,1 sekundy spĺňa praktické požiadavky.

6. Problémy vozidla v zákrutách

6.1 Ackermannov model riadenia

Autá, ktoré sa používaju pre potreby projektu, používajú Ackermannov model riadenia. Ackermannov model definuje geometriu riadenia, ktorá zabezpečuje. aby malo vozidlo správne natočenie volantu v zákrutách. Využíva mechanickú koordináciu uhla predných dvoch kolies auta, čím zaisťuje konzistentnú a plynulejšiu jazdu a zabraňuje kĺzaniu pneumatík počas jazdy. Každé koleso bude mať iné natočenie, inú rýchlosť a pôjde po inej kružnici. Obrázok 6.1 zobrazuje geometriu Ackermannovho riadenia kde:

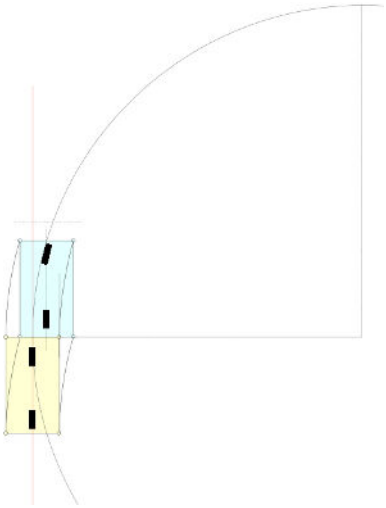
- L je vzdialenosť medzi prednou a zadnou nápravou,
- T je vzdialenosť medzi osami pneumatík na náprave,
- R je polomer otáčania. Zadná náprava smeruje priamo do stredu otáčania. Inak povedané, vozidlo je vždy kolmo na stred otáčania,
- Uhly α_0 a α_1 uhly otočenia predných kolies vzhladom na aktuálne natočenie vozidla.



Obr. 6.1: Geometria Ackermannovho riadenia

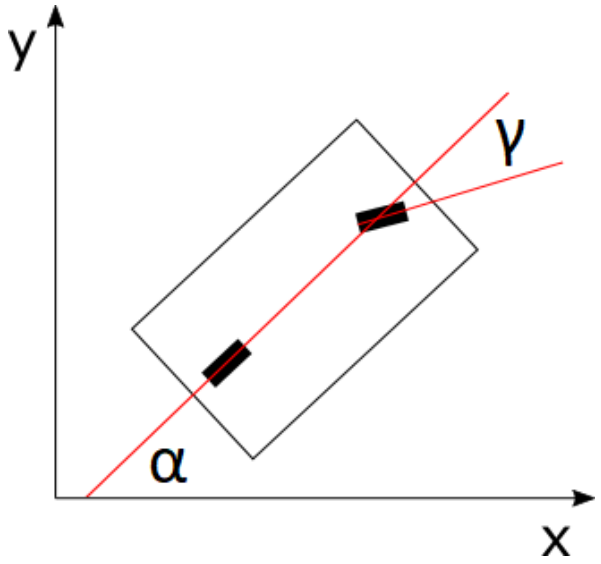
6.2 Simulácia riadenia

Vozidlo má v prednej časti namontovaný senzor, ktorý zachytáva profil vozovky. Aby senzor zaznamenal všetky potrebné informácie, je potrebné, aby stred prednej nápravy bol vždy čo najbližšie k stredu plánovanej dráhy. Podľa Ackermanovho modelu riadenia zadná náprava v zákrute vždy smeruje priamo do stredu otáčania. Ak zadná náprava pôjde po dráhe, tak predná náprava pôjde po kružnici väčšej ako samotná dráha. Stred prednej nápravy bude príliš ďaleko od stredu dráhy a senzor nebude môcť zachytiť všetky informácie, ktoré potrebujeme. Riešením je posunúť celé vozidlo bližšie k stredu otáčania tak, aby predná náprava išla po kružnici dráhy. Aby sa vozidlo pri prechode z rovnej dráhy na zakrivenú dostalo do požadovanej pozície bolo by potrebné spraviť manéver v tvare písmena S:



Obr. 6.2: Manéver v tvare písmena S

Pre lepšie pochopenie správania auta v zákrute bola vytvorená simulácia pohybu vozdila riadeného Ackermannovým riadením. Pre jednoduchosť sa simuluje len zjednodušená verzia vozidla, ktorá má dve kolesá namiesto štyroch. Jedno koleso sa nachádza v strede prednej nápravy a druhé v strede zadnej nápravy. Geometriu tejto verzie popisuje obrázok 6.3.



Obr. 6.3:

α je uhol, ktorý zviera smer vozidla s počiatkom otáčania. γ hovorí to, o koľko je predné koleso otočené vzhľadom na smer vozidla.

Podstatou simulácie je vytvoriť sériu natočení predných kolies v čase tak, aby vozidlo plynule prešlo po určitej dráhe, a vypočítať pozíciu predného a zadného kolesa v jednotlivých časových úsekoch. Pri simulácii sa používajú optimálne natočenia predného kolesa, ktoré vznikli ako výstup evolučného algoritmu popísaného v sekcii 6.3. Na vstupe simulácie je pozícia predného kolesa a orientácia vozidla v štartovacej pozícii, pole uhlov zatočení, rýchlosť a rozmery vozidla. Každá nasledujúca pozícia vozidla sa počíta nasledovne:

```
def getNextPoint(frontPos, backPos, alpha0, gama0, beta, t):
    if(beta + gama0 == 0): # vozidlo ide rovno
        # smer vozidla
        direction = getVector(backPos, frontPos)

        # nová pozícia stredu prednej nápravy
        xf = frontPos[0] + speed*t * direction[0]
        yf = frontPos[1] + speed*t * direction[1]

        # nová pozícia stredu zadnej nápravy
        xb = backPos[0] + speed*t * direction[0]
        yb = backPos[1] + speed*t * direction[1]

        return {"position": [xf, yf], "alpha": alpha0,
                "beta": 0, "gama": 0, "backWheelPos": [xb, yb] }
    else: # vozidlo je v zákrute
        # polomer otáčania predného kolesa
        radiusFront = wheelDistance / math.sin(math.fabs(beta + gama))
        # polomer otáčania zadného kolesa
        radiusBack = radiusFront * math.cos(math.fabs(beta + gama))

        # smer otočenia (kladný / záporný)
        orientation = (beta/math.fabs(beta)) if (beta != 0) else 1
        # rýchlosť zadného kolesa
        omegaBack = speed/radiusBack
        # uhol, o ktorý budú kolesá posunuté v ďalšom kroku
        fi = omegaBack*t*orientation
        # nový uhol otočenia zadneho kolesa (celého vozidla)
        alpha = alpha0 + fi

        # stred otáčania
        cx = backPos[0] + radiusBack*math.cos(alpha0+math.pi/2*orientation)
        cy = backPos[1] + radiusBack*math.sin(alpha0+math.pi/2*orientation)

        # nová pozícia stredu prednej nápravy
        x = ((frontPos[0] - cx)*math.cos(fi)-(frontPos[1]-cy)*math.sin(fi))+cx
        y = ((frontPos[0] - cx)*math.sin(fi)+(frontPos[1]-cy)*math.cos(fi))+cy

        return {"position": [x, y], "alpha": alpha, "gama": gama + beta,
                "beta": beta, "backWheelPos": getBackWheelPos([x, y], alpha)}

def getBackWheelPos(frontPos, alpha):
    x = frontPos[0] + wheelDistance*math.cos(alpha+math.pi)
    y = frontPos[1] + wheelDistance*math.sin(alpha+math.pi)
    return [x, y]
```

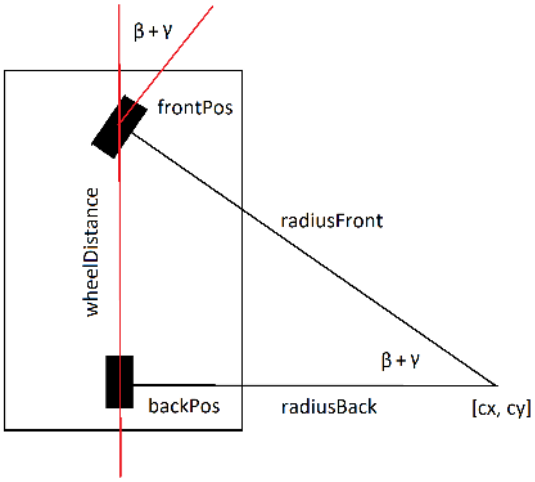
Parametre na vstupe funkcie:

- frontPos - pozícia stredu prednej nápravy,
- backPos - pozícia stredu zadnej nápravy,
- alpha0 - aktuálny uhol, ktorý zviera smer vozidla s počiatkom otáčania,
- gama0 - aktuálny uhol, o ktorý je predné koleso otočené vzhľadom na smer vozidla,
- beta - zmena otočenia predného kolesa,

- t - delta času,
- wheelDistance - vzdialenosť prednej a zadnej nápravy.

Objekt na výstupe funkcie obsahuje dáta o pozícii vozidla v danom čase:

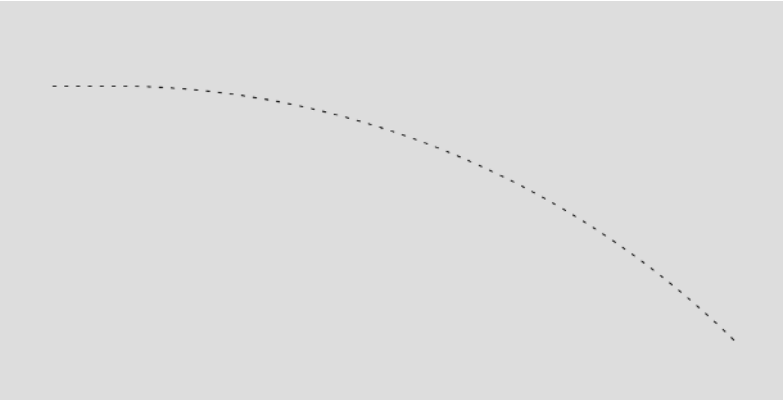
- position - pozícia stredu prednej nápravy,
- backWheelPos - pozícia stredu zadnej nápravy,
- alpha - nový uhol, ktorý zviera smer vozidla s počiatkom otáčania,
- gama - nový uhol, o ktorý je predné koleso otočené vzhľadom na smer vozidla,
- beta - zmena otočenia predného kolesa.



Obr. 6.4: Parametre funkcie

6.3 Evolučný algoritmus pre nájdenie ideálnej trajektórie auta

Problém hľadania správneho natáčania volantu pri prechode z rovného do zakriveného úseku budeme študovať na modelovej situácii zobrazenej na obrázku 6.5. Dráha je zložená z jednej úsečky a z jedného kružnicového oblúku. Spojenie týchto dvoch objektov bude tvoriť prechod medzi rovným úsekom a zákrutou.



Obr. 6.5: Tvar dráhy, na ktorej testujem evolučný algoritmus

Parametre dráhy sú popísané v JSONe nasledovne:

```
[
        {
            "type": "line",
            "start": [50, 100],
            "end": [70, 100]
        },
        {
            "type": "circle",
            "start": -1.5705963267949,
            "end": -0.785398163397448,
            "radius": 300,
            "center": [600, 400]
        }
    ]
```

Úsečka je určená začiatočným a koncovým bodom. Kružnicový oblúk je daný stredom, polomerom a uhlom začiatku a konca oblúku v radiánoch.

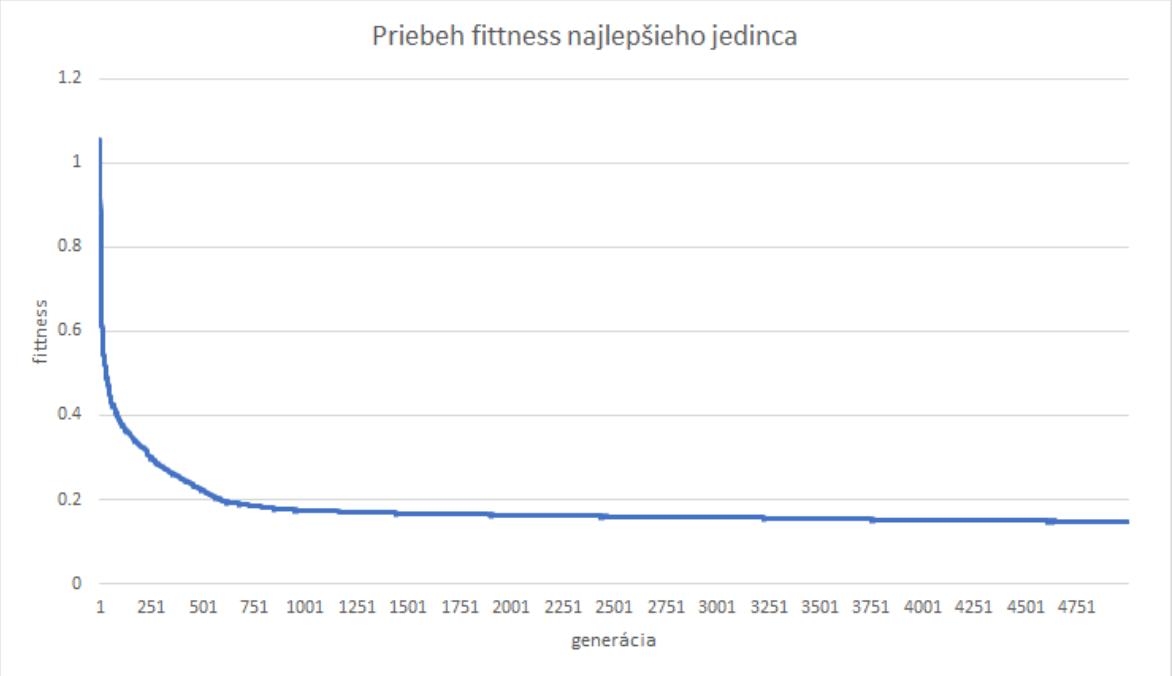
Na nájdenie najlepšej trajektórie pri prechode z rovnej dráhy do zákruty som sa pokúsila použiť evolučný algoritmus. Evolučné algoritmy sa používajú pri riešení optimalizačných úloh v prípadoch, kedy nie je podstatné nájsť najoptimálnejšie riešenie, ale stačí sa mu čo najviac priblížiť, a hlavne, keď nepoznáme analytické, resp. deterministické riešenie problému. Jedným jedincom populácie evolúcie bude séria náhodných zatočení kolies prednej nápravy. Aby vzniknuté trajektórie neišli úple mimo našej dráhy, v náhodnosti zatočení bude hrať rolu aj pravdepodobnosť. Pre generovanie uhlov zatočenia v rozmedzí -4° až +4° sa používa trojuholníkové rozdelenie. Ak sa trajektória vozidla príliš vzdaľuje od dráhy vozidlo sa s väčšou pravdepodobnosťou natočí smerom k dráhe cesty. Maximálne natočenie prednej nápravy oproti zvyšku vozidla je 22°.

Takto sa vygeneruje 300 trajektórií, ktoré sa budú evolúciou postupne upravovať do želanej podoby. Pre každú trajektóriu sa vypočíta jej chyba oproti ideálnej trajektórii.

Evolúcia pozostáva z 5000 generácií. Základný krok evolúcie:

deNájdu sa dve náhodné trajektórie. Vybrie sa z nich tá, ktorá má menšiu chybu. Tento proces sa zopakuje ešte raz, čím sa získajú dve trajektórie do ďalšieho spracovania. (štandardná metóda výberu tournament selection [5]) Dve vybrané trajektórie sa skrížia tak, že každá z nich rozdelí na dve časti (one-point crossover), resp. na tri časti v dvoch náhodných bodoch (two-point crossover) v náhodnom bode a časti populácie, ktoré sa nachádzajú za daným bodom, resp. medzi dvoma bodmi sa vymenia. V ďalšom kroku sa aplikuje mutácia na obidve trajektórie. S určitou pravdepodobnosťou sa vyberie jedno, prípadne viacero otočení volantom pozdĺž jazdy po trajektórii a náhodne sa pozmení ich uhol. Získané trajektórie sa uložia do novej generácie.

Krok evolúcie sa opakuje až kým sa nezíska N nových jedincov a nenaplní sa celá populácia ďalšej generácie. Hodnota N je veľkosť populácie. Pri vyberaní jedincov do ďalšej generácie sa aplikuje elitizmus. Dva jedince s najmenšou chybou sa automaticky presunú do novej generácie.



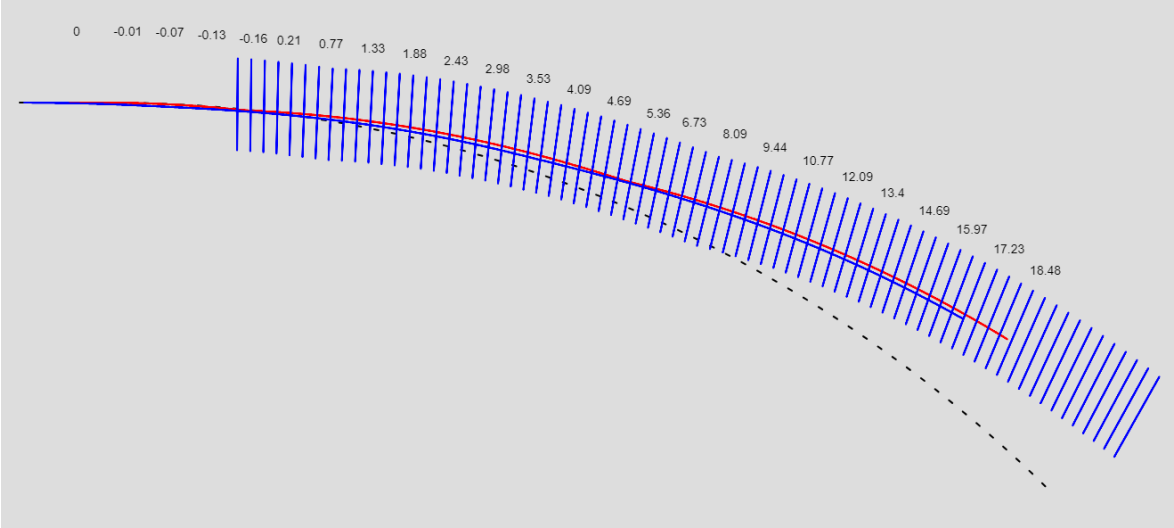
Obr. 6.6: Zmenšujúca sa chyba pre 5000 generácií

Experimentovala som s rôznymi verziami účelovej funkcie i rôznymi verziami operátora mutácie. Prvou verziou bolo porovnávanie trajektórií po bodoch - ako chyba v

každom bode sa použila vzdialenosť tohto bodu k najbližšiemu bodu želanej trajektórie. Problémom tohto riešenia bolo, že zmena zatočenia volantu v jednom časovom okamihu ovplyvňuje celý zvyšok trajektórie, čím znehodnotí ohodnotenie všetkých nasledujúcich zatočení. Evolučný algoritmus funguje správne iba vtedy, ak sa (takmer) z ľubovoľného iniciálneho náhodného riešenia postupným aplikovaním evolučných operátorov (mutácia, crossover) riešenie postupne zlepšuje až k optimálnemu (alebo takmer optimálnemu) riešeniu, ktoré algoritmus hľadá. Preto som hľadala takú kombináciu účelovej funkcie a operátora mutácie, ktorá by túto požiadavku spĺňala. Ako sľubná sa ukazovala nasledujúca myšlienka: Nebudeme si všímať kadiaľ trajektória prechádza a akú veľkú absolútnu chybu voči želanej trajektórii robí, ale iba lokálnu chybu, ktorú spôsobuje zatočenie volantu v každom jednom mieste. Zmena natočenia volantu v jednom mieste spôsobí zmenu tvaru trajektórie len lokálne, v ďalších bodoch trajektórie bude vozidlo pokračovať rovnakým spôsobom ako pred vykonaním zmeny (mutácie), pretože natočenia volantu budú identické: tvar trajektórie z každého bodu ďalej nezávisí od natočení volantu pred týmto bodom, ale iba od natočenia v tomto a ďalších časových okamihoch. Budeme si teda všímať každé tri za sebou idúce body trajektórie, ktorú vytvorila predná náprava. Uhol, ktorý tieto tri body vytvárajú, porovnáme s uhlom, ktoré vytvárajú tri body v zodpovedajúcich časových okamihoch na želanej trajektórii. Zodpovedajúci operátor mutácie teda v náhodnom bode trajektórie otočí volantom o určitý malý uhol a v nasledujúcom bode trajektórie otočí volantom o rovnaký uhol späť, čím spôsobí, že v nasledujúcich bodoch trajektórie bude natočenie volantu rovnaké ako pred mutáciou a tvar trajektórie v nasledujúcich bodoch bude rovnaký. Ako sa však ukázalo, ani táto verzia algoritmu nevyhovovala podmienkam, aby algoritmus konvergoval k správnej trajektórii. Dôvodom je, že zmena natočenia volantu len v jednom bode B_i spôsobí, že sa tvar vytvorenej trajektórie zmení nielen v zodpovedajúcom bode B_i trajektórie, ale aj v bode B_{i+1} tesne za ním nasledujúcom Hoci vo všetkých ostatných bodoch $B_{i+2} \ldots B_{max}$ sa už tvar trajektórie nezmení, táto situácia spôsobuje závislosť, ktorá sa propaguje až do konca trajektórie. Opravenie chyby, ktorú spôsobíme v nasledujúcom bode B_{i+1} si totiž vyžiada otočenie volantom v bode B_{i+1}, čo spôsobí chybu v bode B_{i+1} atď. až po B_{max}. Ukazuje sa, že problém hľadania natočení volantu pre jazdu po želanej trajektórii nie je možné hľadať evolučným algoritmom, ktorý by v genotype reprezentoval celú trajektóriu a na celom takomto vektore aplikoval štandardné operátory mutácie a kríženia.

Riešením by mohlo byť použiť optimalizačný algoritmus postupne - pre každý časový krok najskôr nájsť vhodné natočenie volantu, ktoré vozidlo (stred jeho prednej nápravy) dostane do správnej polohy v nasledujúcom časovom kroku a potom toto natočenie ďalej nemeniť. Namiesto celého vektora $\{\beta_i\}_1^n$ naraz budeme teda vždy hľadať len jedno nasledujúce β_i pre už zafixované hodnoty $\{\beta_j\}_1^{i-1}$, až kým presnosť nebude vyhovujúca.

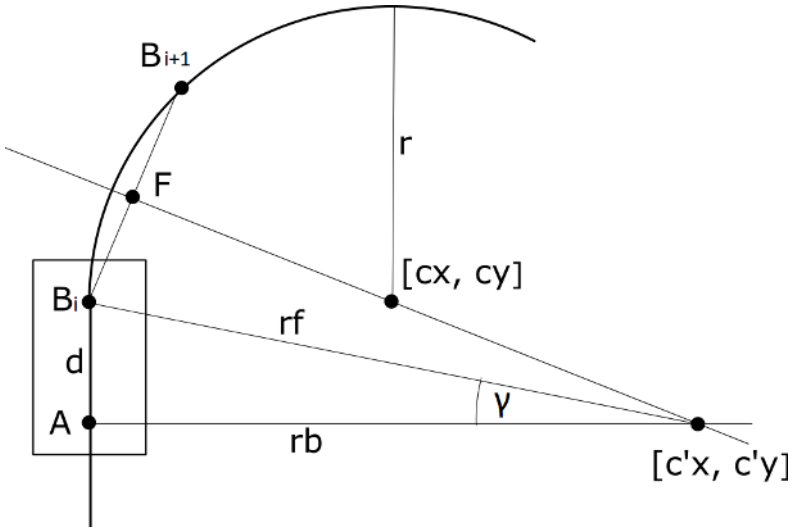
Na obrázku 6.7 je znázornený výsledok pokusu o nájdenie ideálnej trajektórie. Čiarkovaná čiara je dráha po ktorej má vozidlo prejsť. Červená čiara znázorňuje stred prednej nápravy, modrá čiara stred zadnej nápravy a priečne modré čiary sú laserové čiary senzora. Čísla na obvode sú vzdialenosti nájdenej trajektórie a dráhy v rôznych bodoch.



Obr. 6.7: Pokus o evolúciu trajektórie

6.4 Analytické riešenie

Podrobné úvahy nad riešením pomocou evolučného algoritmu ma nakoniec doviedli k zisteniu, že problém hľadania optimálneho scenára zatáčania volantom, pre požadovanú trajektoriu prednej nápravy má aj pomerne jednoduché analytické riešenie. Všetky kolesá vozidla sa budú pohybovať po sústredných kružniciach s tým istým stredom otáčania. Tento stred otáčania leží na priamke prechádzajúcej stredmi zadných kolies - priamka je kolmá na os vozidla. Aby sa vozidlo za príslušný časový krok dostalo do želaného bodu B_{i+1}, musí sa v pohybovať po kružnici s oveľa väčším polomerom ako je polomer kružnice dráhy. Potrebujeme, aby stred kružnice dráhy ležal na priamke ktorá prechádza stredmi kolies prednej nápravy. Výsledkom úsudku je, že stred otáčania musí ležať na osi tetivy kružnice dráhy vytvorenej bodmi B_i a B_{i+1}. Situácia je znázornená na obrázku 6.8



Obr. 6.8: Analytické riešenie

- A - pozícia stredu zadnej nápravy,
- B_i - pozícia stredu prednej nápravy,
- B_{i+1} - pozícia, do ktorej sa musí predná náprava dostať v ďalšom časovom kroku,
- d - vzdialenosť prednej a zadnej nápravy,
- rf - polomer otáčania prednej nápravy,
- rb - polomer otáčania zadnej nápravy,
- F - stred tetivy na krunici dráhy. Začiatočným bodom tetivy je aktuálna poloha prednej nápravy a koncovým bodom je želaná poloha prednej nápravy v ďalšom časovom kroku,
- r - polomer kružnice dráhy,
- [cx, cy] - stred kružnice dráhy,
- [c'x, c'y] - stred kružnice dráhy,
- γ - uhol, ktorý sa rovná uhlu natočenia prednej nápravy v ďaľšom časovom kroku (vychádzam z geometrie ackermannovho riadenia popísanej v sekcii 6.2).

Vstupom našich výpočtov sú hodnoty A, B, d, r a [cx, cy]. Výstupom je uhol γ. Prvým krokom je nájsť stred otáčania [c'x, c'y]. Tento bod nájdeme ako priesečník dvoch priamok. Jednou je priamka vedená bodmi F a [cx, cy] a druhou je priamka určená bodom A a vektorom kolmým na smer vozidla. Následne nájdem polomer otáčania zadného kolesa rb ako vzdialenosť bodov A a [c'x, c'y].

7. Záver

V rámci projektu venujúcemu sa autonómnej mobilite vzniklo množstvo malých podúloh a táto práca sa venuje práve dvom z nich. Prvou úlohou bolo vytvoriť spoľahlivé lokálne navigovanie vozidla po vozovke. Pre účely tejto úlohy bola vytvorená séria 3D vodiacich značiek, ktorých úlohou bolo usmerňovať vozidlo na vozovke do správnej pozície v rovine aj v zákrute. Ideou bolo vyrobiť vozidlo s namontovaným profilometrom, ktorý bude pri pohybe vozidla snímať profil vozovky pred vozidlom a vyhľadávať značky. Po prečítaní značky bude vozidlo schopné spoľahlivo vyhodnotiť situáciu a vhodne zabočiť. Prvý naimplementovaný algoritmus rozpoznávania využíval systém porovnávania bitmáp. Na rovných dráhach tento spôsob fungoval dobre, ale akonáhle vozidlo vošlo do zákruty algoritmus narazil na svoje hranice. Dáta získané z profilometra boli natoľko deformované, že algoritmus ich nevedel zrekonšruovať a rozpoznať. Následne bol naimplementovaný pokus o robustnejší algoritmus, ktorý pri rozpoznávaní využíval hrany značiek. V tomto prípade deformácia nespôsobovala problémy, pretože akokoľvek pretiahnutá hrana stále obsahuje informáciu potrebnú pre rozpoznanie. Tento spôsob je momentálne označený za konečný. Miesto pre vylepšovanie je stále v spoľahlivosti rozpoznávania, systém je schopný rozoznať značku v približne 95% prípadov.

Druhou úlohou bolo nájsť optimálnu trajektóriu pre vozidlo tak, aby profilometer skenoval vozovku čo najefektívnejšie a ziskal čo najviac čistých dát. Ideou bolo použiť evolučný algoritmus, ktorý by nás hrubou silou priviedol k čo naoptimálnejšiemu riešeniu. Táto úloha sa ukázala ako náročná. Nájsť optimálnu dráhu evolúciou je komplikované vzhľadom na to, že aj malá zmena zatočenia v trajektorii môže razantne zmeniť celú výslednú trajektóriu. Dlhé úvahy a pokusy viedli k nájdeniu analytického riešenia problému.

7. Literatúra

[1] V. Milanés, D. F. Llorca, B. M. Vinagre, C. González and M. A. Sotelo, "Clavileño: Evolution of an autonomous car,"13th International IEEE Conference on Intelligent Transportation Systems, Funchal, 2010

[2] Discover SEDRIC @ Volkswagen Group, (2018, May 03)

[3] Autopilot, Tesla, Inc, (2018, May 03)

[4] Waymo, (2018, May 03)

[5] Agoston E. Eiben, J. E. Smith: Introduction to Evolutionary Computing, Springer-Verlag, 2003