VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

ŘÍZENÍ AUTONOMNÍ FORMULE

AUTONOMOUS FORMULA CONTROLLING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MÁRIO HARVAN**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JAROSLAV ROZMAN, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Harvan Mário**
Program: Informační technologie
Název: **Řízení autonomní formule**
Autonomous Formula Controlling
Kategorie: Umělá inteligence

Zadání:

1. Nastudujte plánování cesty a zaměřte se na plánování cesty vozidel s Ackermannovým podvozkem. Nastudujte metody pro jízdu robota (formule) po čáře (např. implementace v ROS, PID regulátor, atd.).
2. Navrhněte program, který v zadané cestě určí co nejvhodnější cestu pro vozidlo s Ackermannovým podvozkem. Zároveň navrhněte způsob, jak vozidlo na navržené cestě řídit. Uvažujte i o postupné úpravě cesty a parametrů řídicího algoritmu při jednotlivých průjezdech cestou.
3. Navržené programy implementujte.
4. Proveďte otestování buď v simulaci nebo na reálné formuli. Výsledky zhodnoťte a navrhněte případná vylepšení.

Literatura:

- Howie Choset et al., Principles of Robot Motion, 2005, ISN 0-262-03327-5

Pro udělení zápočtu za první semestr je požadováno:

- první dva body zadání

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Rozman Jaroslav, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 3. listopadu 2021

Abstrakt

Cieľom tejto práce je rozobrať problématiku riadenia autonómnej formule v súťaži Formula Student. Práca sa zaoberá návrhom a implementáciou systému, ktorý dokáže identifikovať trať, vypočítať trasu, ktorú ma formula nasledovať a riadiť formulu po vypočítanej trase. Úlohou systému je riadiť formulu tak, aby zvládla trať prejsť v najkratšom možnom čase. Súčasťou práce je teoretický model vozidla, pomocou ktorého dokáže algoritmus vypočítať maximálnu možnú rýchlosť formule v každom úseku trate. V závere práce je vyhodnotenie testovania systému v simulátore a porovnanie rôznych prístupov k riadeniu autonómneho vozidla.

Abstract

The aim of this paper is to analyze the problems of autonomous car system in the Formula Student competition. The paper focuses on design and implementation of a system that can identify race track, calculate the best route that car can follow, and control car to follow given route. The objective of the system is to control car so that it can go around the track in the shortest possible time. Part of the paper is a theoretical model of the vehicle, which allows an algorithm to calculate the maximum possible speed of the formula in each section of the track. Last section focuses on system testing inside simulator and comparison of different path finding algorithms.

Kľúčové slová

riadenie autonómnej formule, riadenie vozidla, PID regulátor, detekcia trate

Keywords

Formula student driverless, driverless pipeline, autonomous driving, PID regulator, track detection

Citácia

HARVAN, Mário. *Řízení autonomní formule*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.

Řízení autonomní formule

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracovával samostatne pod vedením pána Ing. Jaroslava Rozmana, PhD. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Mário Harvan
11. mája 2022

Poďakovanie

Týmto ďakujem vedúcemu mojej práce Ing. Jaroslavovi Rozmanovi, PhD., za voľnosť počas realizácie tejto práce a odborné vedenie. Taktiež by som chcel poďakovať tímu TU Brno Racing za poskytnutie možnosti vývoja systému pre ich závodnú formulu a odbornú pomoc kolegov tímu poskytnutú pri riešení mojej práce. V neposlednom rade chcem poďakovať mojim rodičom za podporu počas štúdia.

Obsah

Kapitola 1

Úvod

Riadenie autonómneho vozidla je téma, ktorá je v posledných rokoch často rozoberaná. Väčšinou sa spája s riadením osobných automobilov v premávke alebo rôznych robotov. Táto práca je zameraná na problematiku riadenia autonómneho vozidla z pohľadu závodnej formule. Pre riadenie osobného automobilu je priorita, aby bola jazda bezpečná a v súlade s pravidlami cestnej premávky. Pre bezpilotný závodný monopost[1] je priorita prejsť trať v najkratšom možnom čase tak, aby boli naplno využité všetky možnosti monopostu. Cieľom tejto práce je navrhnúť a implementovať program na riadenie závodného monopostu Dragon E2 tímu TU Brno Racing. Monopost sa bude zúčastňovať na súťažiach Formula Student, kde je od roku 2022 podmienka, aby monopost odjazdil určité disciplíny bez pilota.

Formula Student je súťaž študentských tímov z technických univerzít po celom svete. Cieľom súťaže je postaviť závodný monopost, s ktorým sa študenti zúčastňujú na súťažiach a pretekajú s tímami z iných univerzít. Súťaž je kópiou americkej súťaže Formula SAE, ktorá funguje od roku 1981 a do Európy prišla v roku 1998. Formula Student vždy sleduje trendy v automobilovom priemysle, podľa ktorých upravuje pravidlá, aby študenti vždy pracovali s najmodernejšími technológiami. V začiatkoch pretekali monoposty výhradne so spaľovacími motormi, neskôr sa začalo prechádzať na elektrický pohon. Od roku 2022 musí monopost zvládnuť vybrané disciplíny autonómne, teda bez pilota. Pokiaľ tím nezvládne postaviť monopost, ktorý bude schopný zvládnuť určené disciplíny autonómne, dostane bodovú zrážku kvôli ktorej bude prakticky nemožné umiestniť sa na vrchných priečkach. [10]

Tím TU Brno Racing funguje už od roku 2010 pod záštitou Fakulty strojného inžinierstva VUT a každý rok od svojho vzniku postaví nový monopost pre túto súťaž. Tím patrí medzi svetovú špičku. V sezóne 2019 sa v svetovom rebríčku umiestnil na 9 mieste. Prvých 10 monopostov s názvom Dragon 1 až 10 malo spaľovací pohon. Pre sezónu 2021 tím postavil prvý monopost s elektrickým pohonom Dragon eD1. Pre sezónu 2022 je vo vývoji elektrický monopost Dragon E2, ktorý bude po technickej stránke pripravený na bezpilotnú jazdu. [10]

[1] Jednomiestny závodný automobil s otvorenou kabínou

Obr. 1.1: Tím TU Brno Racing pri predstavení monopostov Dragon X a Dragon eD1.

Kapitola 2

Problematika riadenia autonómnej formule

Pri návrhu riadenia autonómnej formule potrebné zohľadniť, akým spôsobom bude na súťažiach definovaná trať, aké rôzne disciplíny musí monopost zvládnuť a ako sa dokáže navigovať po trati. V prvej časti tejto kapitoly sú popísané pravidlá Formula Student a jednotlivé časti autonómneho systému, aby bolo jasné, s akými dátami môžme pri navigácii a riadení pracovať. V druhej časti sú analyzované rôzne prístupy riadenia autonómneho vozidla.

2.1 Bezpilotné disciplíny na súťaži

Pravidlá Formula Student nám definujú tri disciplíny v ktorých súťažia autonómne formule:

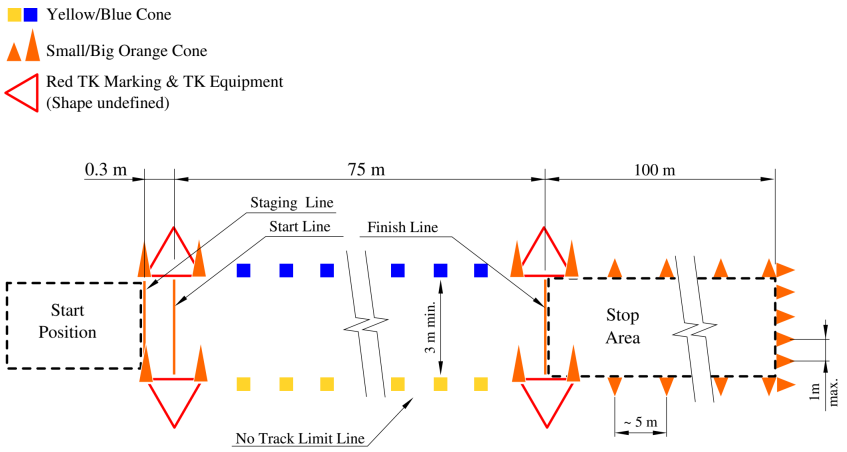
- Bezpilotná akcelerácia
- Bezpilotný skidpad
- Bezpilotná jazda traťou

Pre každú z disciplín platia rôzne pravidlá. Pre súťaž v roku 2022 je potrebné, aby formula zvládla akceleráciu a skidpad. Jazda traťou je zatiaľ nepovinná, ale v našej práci sa budeme sústrediť na všetky disciplíny. [5]

Vo všetkých disciplínach je trať definovaná pomocou modrých, žltých a oranžových kužeľov. Žlté kužele definujú ľavý limit trate, modré kužele pravý limit. Význam oranžových kužeľov sa líši v závislosti na disciplíne. V prípade, že formula zrazí kužeľ dostane penalizáciu vo forme pripočítania 2 sekúnd k celkovému času. Program riadenia musí počítať s tým, že kužeľ, ktorý formula zrazila sa bude nachádzať na inom mieste. Okrem toho hrozí, že zrazený kužeľ poškodí predné krídlo alebo sa zasekne niekde v podvozku a zhorší jazdné vlastnosti formule. Preto je ideálne kužele vôbec nezrážať.

Bezpilotná akcelerácia

V tejto disciplíne sa hodnotí čas, za ktorý dokáže formula prejsť vzdialenosť 75 metrov. Formula je na začiatku postavená na štartovaciu pozíciu 2.1, štartovacia čiara je určená jedným oranžovým kužeľom na každej strane trate. Po štartovacej čiare nasleduje 75 metrov dlhá rovinka ohraničená modrými a žltými kužeľmi. Po nich nasleduje cieľová zóna označená oranžovými kužeľmi, v ktorej musí formula bezpečne zastať. [5]



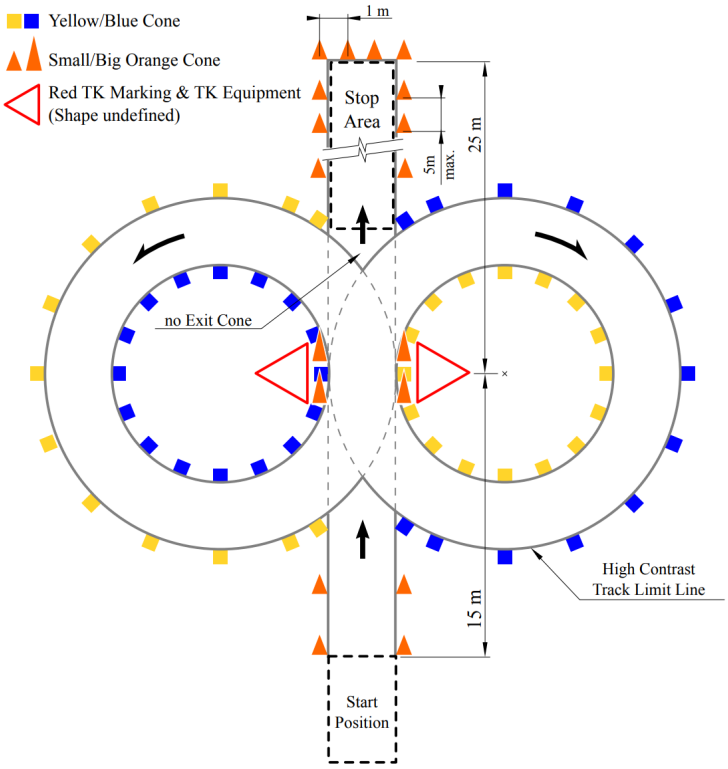
Obr. 2.1: Trať na disciplínu bezpilotná akcelerácia. Prevzaté z: [4]

Bezpilotný skidpad

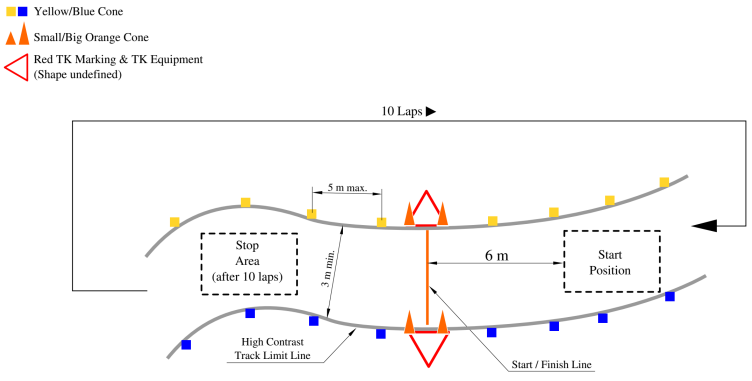
Úlohou skidpadu je preveriť akú má formula mechanickú trakciu. Z toho dôvodu sa jazdia dve kolá okolo kužeľov 2.2 v pomerne nízkej rýchlosti, kedy aerodynamické zariadenia nevytvárajú žiaden prítlak. Konfigurácia kužeľov na skidpad je vopred známa, takže sa môže manuálne nastaviť v programe, ktorý už bude následne len overovať svoju pozíciu na trati. Formula je postavená na štartovaciu pozíciu. Najskôr musí spraviť dva okruhy na pravú stranu, následne dva okruhy na ľavú stranu. Hneď po poslednom okruhu musí formula vojsť do cieľového priestoru, kde musí sama zastať. [5]

Bezpilotná jazda traťou

Špecialitou jazdy po trati je, že dopredu nie je známe, ako bude trať vyzerať. Formula je na začiatku postavená medzi modré a žlté kužele. Trať tvorí okruh, ktorého začiatok je označený oranžovými kužeľmi. Formula musí prejsť 10 okruhov po trati a na konci posledného bezpečné zastaviť za cieľovou čiarou. Tím ma možnosť si pred jazdou prejsť trať pešo, ale je zakázané mať so sebou elektronické zariadenie na mapovanie trate. [5]



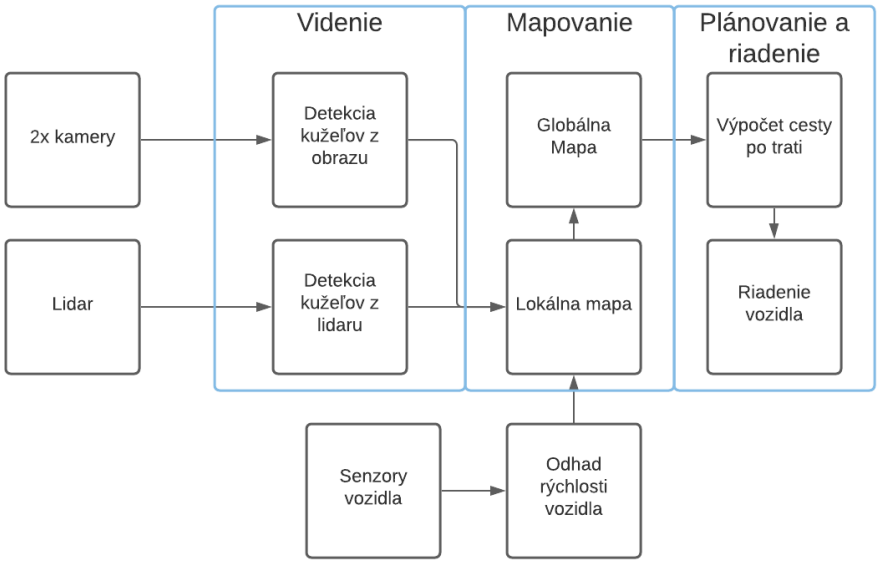
Obr. 2.2: Trať na disciplínu bezpilotný skidpad. Prevzaté z: [4]



Obr. 2.3: Pravidlá pre trať na disciplínu jazda po trati. Prevzaté z: [4]

2.2 Moduly bezpilotného riadenia

Riadenie autonómnej formule po trati je netriviálna úloha, ktorá vyžaduje komplexne a dobre navrhnutý program. Program na riadenie sa skladá z viacerých modulov. Táto práca sa zaoberá modulmi na identifikáciu trate, výpočet ideálnej stopy a rýchlosti vozidla a modul riadenia vozidla. Pre pochopenie celého bezpilotného systému je potrebné poznať základnú funkciu všetkých modulov, ktorá je popísaná v nasledujúcej sekcii. Na obrázku 2.4 je znázornená architektúra tzv. Driverless pipeline, čiže zreťazené jednotlivé moduly programu a komunikácia medzi nimi. [6]



Obr. 2.4: Architektúra programu na riadenie formule.

Počítačové videnie

Úlohou tohto module je správne rozlíšiť objekty reálneho sveta a určiť ich polohu vzhľadom k vozidlu. V prípade bezpilotnej formule je potrebné rozlíšiť kužele rôznych farieb. V autonómnych osobných vozidlách sa najčastejšie používajú len kamery na detekciu objektov. Je to jednak z dôvodu, že kamery sú najlacnejšie a iné senzory (napríklad Lidar[1]) sú nespoľahlivé v horšom počasí. Väčšina tímov Formula Student používa kombináciu Lidaru a kamier. Lidar je senzor, ktorý meria vzdialenosť od okolitých objektov na základe merania doby šírenia laserového lúča odrazeného od objektu. Problém lidaru je, že získava iba informáciu o vzdialenosti objektu. Lidar nedokáže rozoznať jeho farbu a tvar. Z tohto dôvodu sa lidar využíva v kombinácii s kamerami, ktoré dokážu identifikovať objekty aj na základe farieb. Pre účely Študentskej formule je Lidar ideálny, keďže na trati nebudú okrem kužeľov žiadne iné objekty. [1]

Detekcia kužeľov z kamier prebieha väčšinou pomocou neurónových sietí, ktoré je potrebné najskôr natrénovať na obrázkoch, ktoré majú manuálne určené pozície kužeľov. Najväčšou nevýhodou neurónových sietí je výpočtová náročnosť. Pre úspešnú jazdu vozidla

[1] Senzor na meranie vzdialenosti

je nutné spracovávať cca 10 až 20 snímkov za sekundu. Na to je potrebný vysoký výpočtový výkon, väčšinou v podobe nejakého akcelerátora (napríklad Cuda jadrá na grafických kartách Nvidia). To znamená, že počítač, ktorý bude na formuly musí byť dostatočne výkonný. Pokiaľ je identifikovaný rovnaký kužeľ na oboch kamerách, je možné vypočítať jeho vzdialenosť k vozidlu. [1]

Výstup z lidaru je zoznam bodov v priestore so súradnicami x a y. Ako prvé je potrebné odstrániť body, ktoré opisujú zem. Na odstránenie zeme sa používajú rôzne algoritmy. Po odstránení zeme sa všetky body roztriedia do zhlukov. Kvôli vysokému rozlíšeniu moderných lidarov je očakávané, že každý kužeľ bude reprezentovaný viacerými bodmi. Každý zhluk predstavuje jeden kužeľ. [15]

Odhad rýchlosti a smeru jazdy

Odhad rýchlosti a smeru jazdy je netriviálny problém, ktorý je možné riešiť rôznymi spôsobmi. Poznať presnú rýchlosť a smer, ktorým sa vozidlo pohybuje je veľmi dôležité pre modul Mapovanie, ktorý na základe rýchlosti a smeru dokáže určiť či kužeľ, ktorý bol nasnímaný pri pohybe vozidla v rôznych časoch je ten istý kužeľ alebo sa jedná o kužeľ nový. Na odhadnutie rýchlosti vozidla sa používajú senzory otáčok umiestnené v kolesách. Pokiaľ je známy priemer kolesa, je možné jednoducho vypočítať rýchlosť vozidla. Problém však nastáva napríklad, keď vozidlo zatáča. V tom momente opisujú vnútorné kolesá vozidla s Ackermannovým podvozkom [16] kružnicu s menším polomerom ako kolesá vonkajšie. To znamená, že vnútorné kolesá prejdú menšiu vzdialenosť ako vonkajšie. Pri závodných vozidlách a formuliach vznikajú ešte ďalšie problémy.Formula sa môže nachádzať v pretáčavom alebo nedotáčavom šmyku, keď jazdí na limite priľnavosti pneumatík. V prípade vozidiel, ktoré majú len jednu hnanú nápravu sa môže často stať, že zadné kolesá pri výjazde zo zákrut stratia trakciu a vykonajú viac otáčok ako predné. Pri brzdení môže nastať situácia, že sa všetky kolesá zablokujú a vozidlo sa stále ďalej pohybuje. Z tohto dôvodu sa vykonáva fúzia dát z rôznych senzorov. Formula zúčastňujúca sa súťaže Formula študent je typicky vybavená nasledujúcimi senzormi:

- Inerciálna meracia jednotka
- Senzory otáčok kolies
- GPS
- Senzor otočenia volantu

Na výpočet rýchlosti a smeru jazdy sa používa fúzia dát z jednotlivých senzorov. Niektoré tímy využívajú kalmanové filtre, iné zase neurónové siete. Ďalšou možnosťou je použiť tzv. True ground speed senzor, ktorý pracuje na princípe optického snímania zemského povrchu. Takéto senzory sú však pre väčšinu tímov finančne nedostupné a potrebujú byť v konštantnej výške od zeme, čo komplikuje ich uchytenie na vozidle. [14]

Mapovanie

Mapovanie dostáva na vstupe lokálnu pozíciu kužeľov, teda súradnice od aktuálnej pozície auta. Taktiež má k dispozícii odhad rýchlosti a smeru jazdy z príslušnej časti programu. Mapovanie využíva tieto informácie na poskladanie globálnej mapy. Globálna mapa zobrazuje pozíciu všetkých kužeľov, aj tých, ktoré vozidlo identifikovalo v minulosti a pozíciu auta

v jednom súradnicovom systéme. Z globálnej mapy je možné určiť kde na trati sa vozidlo nachádza. Mapovanie musí správne identifikovať ktoré kužele vozidlo videlo pri ďalšom meraní znovu a asociovať ich ku rovnakému kužeľu z predchádzajúcich meraní. Nové kužele musí pridať do globálnej mapy. To je možné docieliť vďaka informácii akú vzdialenosť a akým smerom sa vozidlo pohybovalo medzi dvomi meraniami. Preto je veľmi potrebný modul na odhad smeru jazdy a rýchlosti vozidla. Algoritmus taktiež musí počítať so zvyšujúcou sa presnosťou merania vzdialenosti kužeľov, čím bližšie sa vozidlo pri kuželi nachádza. Pozícia kužeľov sa medzi jednotlivými meraniami upravuje tak, aby súradnice v globálnej mape presne popisovali realitu. [6]

2.3 Prieskumné kolo po trati

Prvou úlohou modulu plánovania trasy a riadenia vozidla, je správne identifikovať, kde sa nachádza trať. Modul získava globálnu mapu so súradnicami kužeľov z modulu Mapovanie. Pre disciplíny akcelerácia a skidpad je identifikácia trate veľmi jednoduchá. Obe trate sú aj s presnými rozmermi definované v pravidlách. Modul mapovanie načíta očakávané súradnice kužeľov vopred, následne už len overuje či sa všetky kužele nachádzajú na správnom mieste. Do manuálne vytvorenej mapy je možné pridať trasu, ktorú má formula nasledovať. Pre tieto disciplíny nie je potrebné prejsť prieskumné kolo po trati. [13]

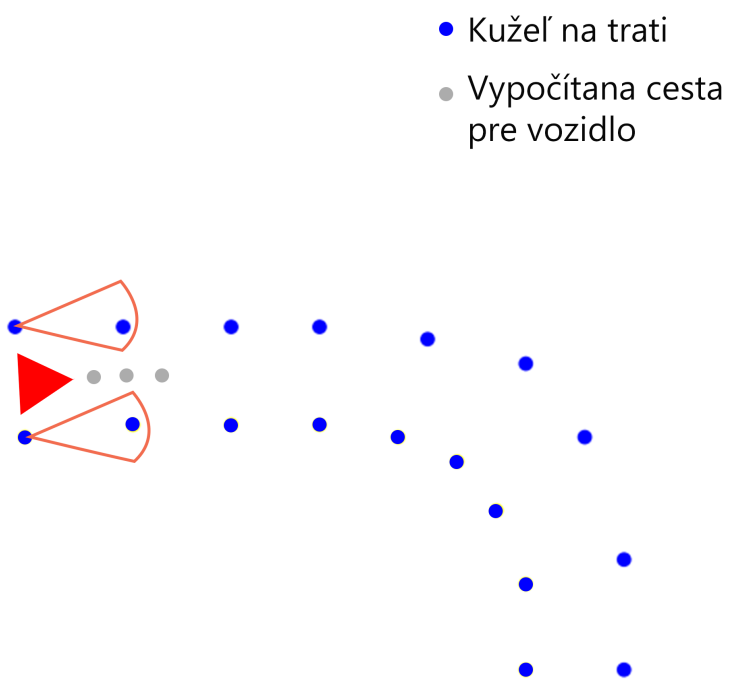
Problém nastáva s disciplínou bezpilotná jazda traťou. Trať vopred nie je známa. Väčšina tímov používa prejazd prvého kola trate na detekciu všetkých kužeľov. Po prejdení celej trate prvý krát sú v globálnej mape uložené súradnice všetkých kužeľov. Pomocou globálnej mapy sa vypočíta ideálna cesta traťou a rýchlosť prejazdu trate. Prvé kolo sa spravidla jazdí pomalou rýchlosťou, formula sa vždy snaží držať stredu trate a cestu po trati upravuje podľa toho kde boli identifikované nové kužele. Formula sa pohybuje po trase, ktorá je definovaná pomocou bodov, ktoré sú vypočítane z pozície kužeľov pred formulou. Postupnou jazdou po trati sú identifikované nové kužele a prepočítava sa nová trasa, po ktorej má formula ísť tak, aby sa držala stredu trate. [13]

Pred začiatkom jazdy sú známe len dve informácie o trati. Formula je vždy postavená v strede trate, tesne pred cieľovou čiarou. Na ľavej strane formule sa nachádzajú žlté kužele a na pravej kužele modré. Modul identifikácie trate má za úlohu aproximovať smer jazdy pre formulu, na základe ktorého dokáže formula úspešne prejsť prvé kolo. Druhou úlohou je vytvoriť usporiadaný zoznam bodov, ktorý definuje trasu po trati. Spôsoby identifikácie trate sa delia na dve hlavné kategórie. Prvá kategória je identifikácia trate len podľa pozície kužeľov, bez využitia znalosti o farbe kužeľov. Druhá metóda je identifikácia pomocou farby aj pozície kužeľov. Každá kategória má svoje výhody aj nevýhody. Z rôznych dôvod môže nastať situácia, že budú známe súradnice kužeľov, ale nie ich farba. Môže to byť spôsobené napríklad mechanickou poruchou kamier, nesprávnou identifikáciou kužeľov pomocou neurónových sietí alebo zlými svetelnými podmienkami (napríklad zapadajúce slnko svietiace priamo do objektívu kamery). Vo všetkých týchto prípadoch sa formula dokáže navigovať pomocou Lidaru. Preto je ideálne implementovať kombináciu týchto dvoch algoritmov tak, aby pri výpadku jedného systému bolo stále možné pokračovať v jazde. [13]

Identifikácia trate bez použitia farieb kužeľov

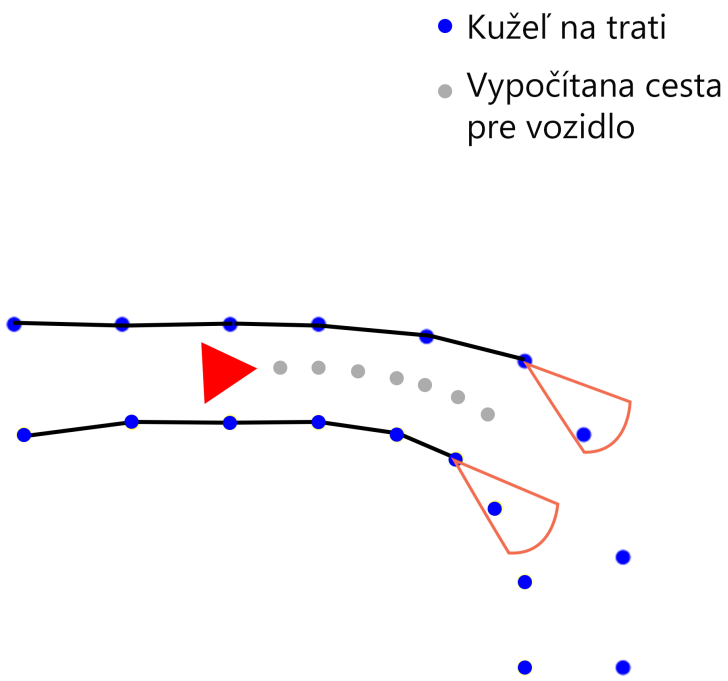
Pred štartom autonómnej jazdy je formula umiestnená do stredu trate, pred štartovaciu čiaru. Nasleduje asi minúta čakania, kým všetci ľudia odídu z miesta trate za ochranné bariéry. Vďaka tomu má formula dostatok času identifikovať prvých pár kužeľov, ktoré sú

v jej blízkosti. O túto skutočnosť sa opiera prvý algoritmus. V prvom kroku identifikuje najbližší pravý a ľavý kužeľ od formule. Tieto kužele sú priradené k ľavému a pravému limitu trate. Algoritmus následne vytvorí kruhový výsek okolo každého kužeľa. Pokiaľ sa nachádza ďalší kužeľ vnútri tohto výseku, je priradený k príslušnému limitu 2.5 trate. Algoritmus rekurzívne priraďuje kužele, pokiaľ existujú nové kužele, ktoré je možné priradiť. Po priradení všetkých nových kužeľov vypočíta stredovú líniu po novej časti trate. Po tejto ceste následne prechádza formula. Vždy, keď sú identifikované nové kužele, tak sa algoritmus spúšťa znova a stredová línia trate je prepočítaná. V prípade, že nie sú identifikované žiadne nové kužele, tak je vygenerovaná rovná trasa so smerom rovnakým, akým smeruje formula. To dáva šancu ostatným modulom identifikovať nové kužele pred tým, ako formula prejde mimo trať. Problém algoritmu nastáva v ostrých zákrutách, kedy je potrebné rotovať kruhové výseky 2.6, ktoré sa používajú na detekciu nových kužeľov. Kruhové výseky sú otočené o uhol, ktorí zvierajú posledné tri kužele na trati. Táto úprava pomáha identifikovať kužele aj počas ostrých zákrut. Samozrejme je možné meniť parametre kruhového výseku tak, aby najlepšie identifikoval správne kužele. [13]



Obr. 2.5: Detekcia kužeľov, pomocou kruhových výsekov a plánovanie trasy.

V prípade, že sú identifikované aj farby kužeľov, je stále možné použiť tento algoritmus. Vždy po klasifikácii kužeľa nastane kontrola, či jeho farba prislúcha traťovému limitu ku ktorému je priradený. Pokiaľ nie, algoritmus hľadá ďalší kužeľ, ktorý vyhovuje tejto podmienke. Vďaka tomu dokáže algoritmus obmedziť počet nesprávnych detekcií kužeľov hlavne v ostrých zákrutách a stáva sa ešte robustnejší.



Obr. 2.6: Rotácia kruhových výsekov v zákrute.

Identifikácia trate s použitím farieb kužeľov

Algoritmus začína rovnako ako algoritmus bez použitia farieb. Identifikuje najbližší ľavý a pravý kužeľ. Algoritmus má však výhodu, že pri hľadaní ľavého kužeľa prechádza iba kuželi so žltou farbou a opačne. Vypočíta sa prvý stredový bod trate. Nasleduje rekurzívne hľadanie ďalších kužeľov. Nový kužeľ musí spĺňať nasledujúce podmienky:

- Nový kužeľ musí byť v oblasti označenej žltou farbou. Oblasť je tvorená aktuálnym ľavým a pravým kužeľom
- Kužeľ musí byť najbližším kužeľom od aktuálneho stredu trate

Nový nájdený kužeľ sa podľa farby klasifikuje a nastaví sa ako aktuálny kužeľ danej farby. Vypočíta sa nový stred trate z aktuálnych kužeľov. Algoritmus končí, keď neexistuje nový kužeľ, ktorý by splnil podmienky. Vždy, keď sa identifikujú nové kužele, tak sa algoritmus spustí znovu. Najväčšia výhoda algoritmu je, že mu nerobia problém ostré zákruty. [8]

2.4 Výpočet ideálnej cesty po trati

Najlepšia cesta pre závodné vozidlo po trati, je cesta, ktorej prejdenie trvá najmenej času. Najviac času je možné ušetriť v zákrutách. Závodní piloti sa od začiatku svojej kariéry učia, ako prechádzať cez zákruty najrýchlejšie. Najlepšia cesta cez zákrutu je taká, pri ktorej musí vozidlo najmenej spomaliť. Mať vyššiu priemernú rýchlosť pri prejazde zákrutou je oveľa dôležitejšie, ako ísť najkratšou možnou trasou. Po zákrute sa väčšinou nachádza

rovinka, na ktorej vozidlo dosiahne väčšiu priemernú rýchlosť, ak bude aj výjazdová rýchlosť zo zákruty vyššia. Vďaka tomu bude celkový čas jazdy po trati menší aj keď vozidlo prejde väčšiu vzdialenosť. Každé závodné vozidlo je limitované maximálnou trakciou, ktorú dokážu pneumatiky poskytnúť. [12]

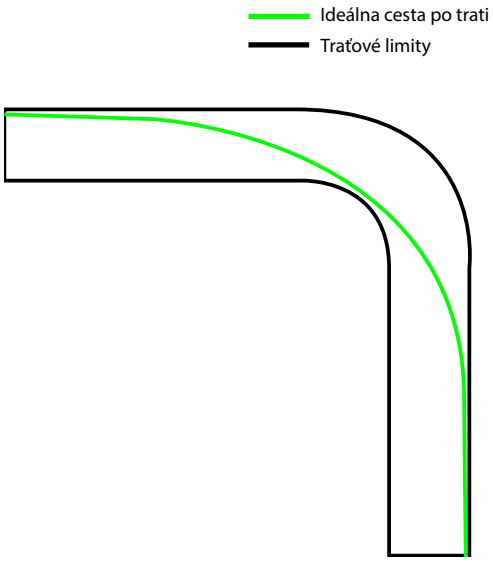
$$v = \sqrt{\frac{F * r}{m}} \tag{2.1}$$

F je maximálna sila ktorú dokážu pneumatiky preniesť na vozovku

r je polomer zákruty

m je hmotnosť vozidla

Zo vzorca 2.1 vyplýva, že na dosiahnutie maximálnej rýchlosti vozidla je potrebné zväčšiť polomer zákruty. Závodní piloti, pri snahe dosiahnuť najlepší čas na jedno kolo, využívajú celú šírku trate aby pre každú zákrutu čo najviac zväčšili jej polomer. Ideálna cesta pre jednoduchú zákrutu je znázornená na obrázku 2.7.



Obr. 2.7: Ideálna stopa pri prejazde zákruty.

Nájsť najrýchlejšiu cestu cez jednu zákrutu je pomerne jednoduché. Problém nastáva, keď je na trati viacero zákrut za sebou. Vtedy nie je možné prejsť prvú zákrutu najrýchlejšou cestou, lebo sa na konci prvej zákruty dostaneme do nevhodnej polohy pre prejazd druhej zákruty. Vtedy je potrebné nájsť taký kompromis, aby bolo celkové prejdenie všetkých zákrut najrýchlejšie. Ak po kombinácii viacerých zákrut nasleduje dlhá rovinka, zvykne sa uprednostňovať prejazd poslednej zákruty maximálnou rýchlosťou, aby sa zvýšila priemerná rýchlosť na rovine. [12]

Diskretizácia trate

Výpočet ideálnej cesty po trati nastáva až v momente, keď formula dokončila prvé kolo po trati. Z modulu mapovania získava globálnu mapu so súradnicami všetkých kužeľov a pomocou identifikácie trate má určenú stredovú líniu pozdĺž celej trate. Trať je potrebné

diskretizovať na body, ktoré opisujú plynulú cestu po trati. Modul pomocou stredovej línie vypočíta priečne úsečky. Na každej takejto priečnej úsečke definuje bod. Všetky body sú preložené kubickou krivkou, ktorá definuje plynulú trasu po trati. Na definovanie úsečiek sa väčšinou využíva parametrické vyjadrenie priamky:

$$\begin{aligned} x &= x_0 + a * t \\ y &= y_0 + b * t \end{aligned} \tag{2.2}$$

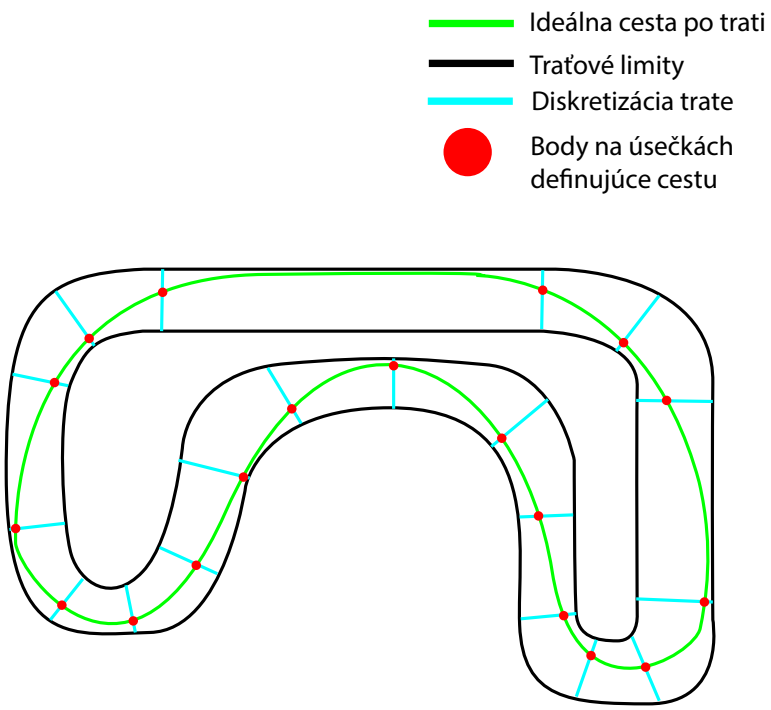
x a y sú súradnice výsledného bodu na priamke

x_0 a y_0 sú súradnice bodu v strede trate

t určuje posun výsledného bodu na priamke

Pre každú priečnu úsečku je potrebné zistiť minimálne a maximálne t, pre ktoré platí, že výsledný bod leží vo vnútri trate. Minimálne a maximálne t určuje limity trate pre každú priečnu úsečku. Po vytvorení priečnych úsečiek pozdĺž celej trate stačí meniť parameter t pre každú úsečku a upravuje sa celá cesta po trati. Ideálne je trať definovať najmenším možným počtom úsečiek tak, aby existovalo čo najmenej parametrov, ktoré je potrebné pri optimalizácii meniť. V zákrutách je potrebné mať viac priečnych úsečiek, aby bola zákruta kubickou krivkou správne popísaná, ale na rovinkách stačí mať úsečiek menej. Diskretizácia trate je znázornená na obrázku 2.8.

Cestu po trati, ktorá je popísaná kubickou krivkou je potrebné disrketizovať do bodov tak, aby sa jednoduchšie počítala rýchlosť vozidla po trati. Algoritmus prechádza po kubickej krivke s konštantným krokom a jednotlivé body ukladá. Veľkosť kroku je potrebné vhodne zvoliť, tak, aby bol zachovaný kompromis medzi výpočtovou náročnosťou a presným popísaním trate. [12]



Obr. 2.8: Trať s vyznačenými priečnymi úsečkami a bodmi na nich, ktorými je definovaná cesta po trati.

Výpočet trvania jedného kola

Ako je spomenuté na začiatku kapitoly, kvalita cesty po trati je hodnotená podľa toho, ako dlho bude formuly trvať, kým prejde jedno kolo. Pre optimalizáciu cesty po trati je potrebné vedieť vypočítať, koľko bude po danej ceste jedno kolo trvať.

V prvom kroku je vypočítaná teoretická maximálna rýchlosť v zákrute podľa vzorca 2.1. Polomer zákruty v danom mieste je vypočítaný z viacerých bodov tak, že sú preložené kružnicou. Každý bod na trase má priradenú teoretickú maximálnu rýchlosť. Ak vozidlo nevyužíva maximálnu možnú trakciu na zatáčanie malo by zvyšnú trakciu využiť na akceleráciu. Na výpočet maximálnej možnej akcelerácie je potrebné zistiť maximálnu silu, ktorou dokáže motor pôsobiť cez pneumatiky na cestu. Maximálna sila sa počíta podľa vzorca:

$$F_a = \frac{P}{v} \tag{2.3}$$

P je výkon motoru

v je aktuálna rýchlosť vozidla

Rýchlosť vozidla v ďalšom bode počas zrýchľovania je vypočítaná pomocou vzorca:

$$v = \sqrt{v_0^2 + 2s * \frac{F_a}{v_0}} \tag{2.4}$$

v je rýchlosť v ďalšom bode

v_0 je aktuálna rýchlosť vozidla

s je vzdialenosť medzi aktuálnym bodom a ďalším bodom

V ďalšom kroku algoritmus začne prechádzať všetky body cesty. Pozerá sa na N bod. Ak má bod $N + 1$ väčšiu maximálnu rýchlosť ako bod N, vypočíta sa sila pôsobiacu na pneumatiky v bode $N + 1$ pri aktuálnej rýchlosti. Rozdiel medzi maximálnou silou, ktorú dokážu pneumatiky preniesť a silou v bode $N+1$ udáva silu, ktorou môže vozidlo zrýchľovať. Podľa vzorca 2.4 je vypočítaná nová rýchlosť v bode $N + 1$ a tá je uložená. Pokiaľ má bod $N + 1$ maximálnu rýchlosť nižšiu ako bod N, algoritmus nevykoná žiadnu akciu.

V poslednom kroku je potrebné zistiť, kedy pred zákrutou je potrebné začať brzdiť, aby vozidlo na začiatku zákruty malo požadovanú rýchlosť. Pri brzdení sa predpokladá, že vozidlo má dostatočne silné brzdy, aby v každej rýchlosti dokázali vyvinúť väčšiu silu, ako je maximálna sila, ktorú dokážu pneumatiky preniesť na zem. Algoritmus pracuje rovnako ako algoritmus pre zrýchlenie, akurát sa body prechádzajú v opačnom poradí.

Po vypočítaní rýchlosti v každom bode trasy je jednoduché vypočítať čas potrebný na prejdenie celej trate. Vypočíta sa čas potrebný na prejedenie každého segmentu trate a následne sa časy sčítajú. Výsledok je celkový čas potrebný na prejdenie jedného kola. [12]

Optimalizácia cesty po trati

Úlohou optimalizácie je nájsť cestu, ktorá bude trvať najmenej času. Hľadá sa konfigurácia parametrov t pre každú priečnu úsečku tak, aby vozidlo prešlo trať najrýchlejšie. Jedná sa o zložitú optimalizačnú úlohu, keďže existuje viac kombinácií ako dokážu bežné počítače vypočítať v rozumnom čase. Výberom algoritmu, ktorý sa na túto úlohu hodí sa budeme zaoberať v praktickej časti. [9]

2.5 Riadenie vozidla po definovanej trati

Z predchádzajúcich modulov sú dostupné informácie o momentálnom stave vozidla, globálna mapa trate, pozícia vozidla na trati a trasa, ktorú ma vozidlo nasledovať. Formula a taktiež väčšina závodných vozidiel využíva Ackermannov podvozok [16]. Pre obsah tejto práce je potrebné vedieť, že podvozok má typicky dve nápravy. Predná náprava má dve kolesá, ktoré je možné natáčať do strán a tým meniť smer jazdy vozidla. Zadné kolesa sú väčšinou pevne uchytené a nemožno s nimi zatáčať. Z toho vyplýva, že vozidlo nedokáže meniť smer jazdy náhodne, ale má maximálny rádius, ktorým vie zatáčať. Podľa toho koľko zatočia predné kolesá, takú kružnicu bude vozidlo opisovať. Zadná náprava vozidla je väčšinou pripojená k motoru, ktorý roztáča zadné kolesá a uvádza vozidlo do pohybu. Pre dosiahnutie vypočítanej rýchlosti vozidla je potrebné ovládať výkon motorov, alebo brzdnú silu vozidla.

Závodné vozidlo je dynamický systém, na ktorí vplývajú rôzne fyzikálne javy. Ovládanie takého vozidla je netriviálna úloha, na ktorú sa väčšinou používajú rôzne typy regulátorov, prípadne neurónové siete. Najjednoduchšia možnosť je použiť PID regulátor.

PID regulátor

PID alebo proportional integral derivative regulátor sa používa na riadenie dynamických systémov so spätnou väzbou tak, aby sa systém nachádzal v požadovanom stave. Na dynamické systémy môžu vplývať rôzne fyzikálne javy, ktoré menia ich reakciu na vstupy. Preto je veľmi zložité popísať ich modelom, podľa ktorého by ich bolo možné regulovať. PID regulátor nepozná parametre systému, reguluje ho na základne spätnej väzby. Na vstupe regulátoru je regulačná odchýlka, čo je rozdiel medzi požadovaným a aktuálnym stavom systému. Na výstupe regulátoru je akčná veličina, ktorá popisuje, čo máme so systémom urobiť, aby sa dostal do požadovaného stavu. Regulátor sa skladá z troch častí P, I, D. Z každej časti je vypočítaná čiastková akčná veličina, ktorá sa sčíta do výslednej. Výstup každej časti upravujeme pomocou konštánt K_P, K_I a K_D. Keďže regulátor nepozná systém ktorý reguluje, je potrebné upraviť jednotlivé konštanty tak, aby akčná veličina presne riadila systém a nerozkmital sa. [2]

Pre prispôsobenie PID regulátora pre daný systém je najdôležitejšie správne vypočítať regulačnú odchýlku. Pre vozidlo, ktoré má nasledovať vyznačenú trasu existuje viacero spôsobov na výpočet odchýlky. Prvý spôsob je vypočítať uhol dvoch vektorov 2.5. Prvý je smerový vektor, ktorý má smer rovnaký ako smer jazdy vozidla. Druhý vektor je vypočítaný medzi pozíciou vozidla a ďalším bodom na trase. Uhol medzi týmito vektormi popisuje, ako je vozidlo odchýlené od požadovanej trasy.

$$\alpha = \arccos \frac{a * b}{\mid a \mid * \mid b \mid} \tag{2.5}$$

a je vektor smeru vozidla

b je vektor vypočítaný z pozície vozidla a ďalšieho bodu trasy

α je uhol medzi dvoma vektormi

Druhá možnosť výpočtu regulačnej odchýlky je vypočítať tzv. cross track error 2.5, čiže vzdialenosť vozidla od úsečky popisujúcej trasu, po ktorej má vozidlo ísť. Ku vzdialenosti je potrebné pripočítať uhol medzi vektormi dotyčnice k trase a smeru aktuálnej dráhy vozidla. [3]

$$CTE = \frac{\mid (x_2 - x_1) * (y_1 - y) - (x_1 - x) * (y_2 - y_1) \mid}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \tag{2.6}$$

x a y sú súradnice vozidla

x_1, y_1, x_2, y_2 sú súradnice definujúce úsečku

CTE je cross track error, čiže vzdialenosť od úsečky

Je ťažké povedať, ktorý spôsob na výpočet regulačnej odchýlky je lepší pre účely tejto práce. Porovnaním oboch spôsobov sa zaoberá sekcia testovania riadenia vozidla.

2.6 Robot operating system

Robot operating system (ROS) je open-source operačný systém, ktorý obsahuje nástroje pre jednoduchší vývoj systému pre robotov rôzneho druhu. Nejedná sa o plnohodnotný operačný systém, je to súbor programov pracujúcich na operačnom systéme Linux. ROS umožňuje jednoduchú komunikáciu jednotlivých častí systému (tzv. Nodes), distribúciu rôznych

vopred vytvorených programov, nástroje na ladenie a testovanie programov, a umožňuje vytvárať jednotlivé časti systému v rôznych programovacích jazykoch. Každý ROS systém sa skladá z ROS Workspace, čo je pracovná plocha, ktorá obsahuje jednotlivé moduly nášho systému, konfiguračný súbor so všetkými konštantami a spustiteľný súbor, ktorý je možné nakonfigurovať tak, aby spustil všetky moduly, ktoré systém k behu potrebuje. [7]

ROS Node

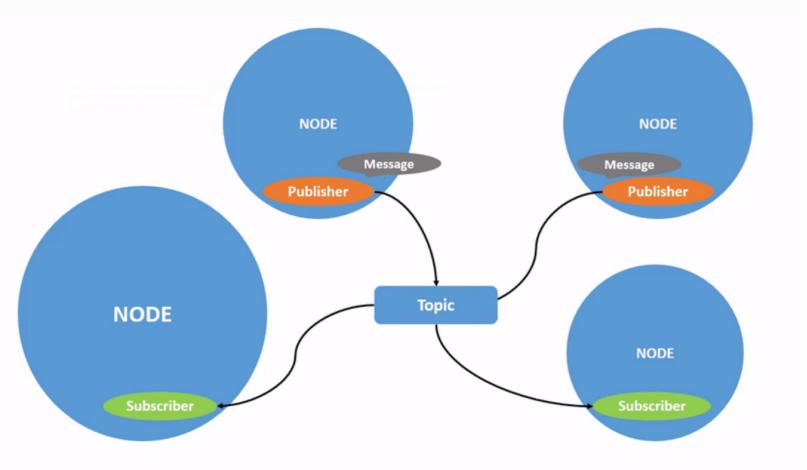
Každý modul ROS systému je nazývaný node. Node môže prijímať správy z iných modulov, je podporovaná synchrónna aj asynchrónna komunikácia. Dôležitou vlastnosťou ROS nodes je, že každý node pracuje vo vlastnom procese operačného systému. Vďaka tomu môže viacero modulov pracovať paralelne. To je dôležitá vlastnosť pre systém riadenia vozidla. Je potrebné, aby riadenie formule fungovalo nezávisle, aj keď sa budú spracovávať nové dáta zo senzorov, alebo keď sa bude počítať ideálna trasa po trati. Keďže je každý node samostatný proces, využitím viac jadrového procesoru sa zvýši výkon celého systému. Program jednotlivých modulov je možné písať v jazyku Python, ale aj v jazyku C++ v prípade, že procesy potrebujú vysokú rýchlosť spracovania. V každom module sa definuje časť kódu, ktorá sa bude vykonávať periodicky. Taktiež je možné definovať obslužné funkcie na spracovanie dát, ktoré sa zavolajú vždy, keď budú pre daný modul dostupné nové správy. [7]é

ROS Topics

ROS Topics je systém komunikácie jednotlivých modulov medzi sebou. Každý modul, ktorý chce posielať dáta, definuje v inicializácii modulu meno a typ správy, ktorú bude odosielať. Ostatné moduly sa následne môžu prihlásiť na prijímanie týchto správ. Každý modul môže nezávisle prijímať správy a zároveň vykonávať inú časť kódu, čo je ideálne pre riadenie vozidla. Formát správ je možné definovať v špeciálnom súbore s príponou *.msg*. ROS ponúka rôzne dátové typy, ktoré sú kompatibilné medzi rôznymi programovacími jazykmi. To umožňuje použiť rôzne jazyky pre jednotlivé moduly. Vďaka systému správ v ROS je možný jednoduchý vývoj jedného modulu, bez ovplyvňovania ostatných modulov. Taktiež je jednoduché celý systém testovať najskôr v simulácii, ktorá bude odosielať a prijímať rovnaké správy ako reálna formula, následne len zmeniť zdroj správ zo simulátora na senzory vo vozidle a systém bude fungovať aj v realite. [7]

2.7 FS Driverless Simulator

FS Driverless Simulator je simulátor určený na testovanie autonómnych systémov pre súťaž Formula student. Simulátor nám ponúka fyzikálny model formule, ktorá má podobné jazdné vlastnosti ako reálna formula, ktorú má systém riadiť na súťažiach. V simulátore sú pripravené trate s rôznou zložitosťou na disciplínu bezpilotná jazda traťou. To je ideálne na postupné testovanie celého systému, ako si poradí aj so zložitejšou traťou. Formulu v simulátore je možné vybaviť rovnakými senzormi, aké budeme mať v realite. Každý senzor má simulovaný šum a rozptyl meraní tak, aby čo najlepšie predstavoval realitu. Formula používa jeden lidar a dve kamery aj v simulátore, aj v realite. Pozícia senzorov je nastavená podobne ako v realite. Formulu je možné ovládať aj manuálne, čo je možné využiť pri zbere dát.



Obr. 2.9: Graf komunikácie medzi ROS Nodes pomocou ROS Topics.
Prevzaté z: [11]

Pripojenie simulátora k autonómnemu systému

Súčasťou FS simulátora je ROS Bridge, ktorý zbiera dáta zo Simulátora a odosiela ich pomocou ROS topics. Simulátor ponúka viacero ROS topic správ, ktoré vie náš systém pomocou ROS prijímať. Každý modul sa prihlási na odoberanie správ ktoré potrebuje. Vďaka tomu dostáva dáta zo simulovaných senzorov. Zoznam správ z ROS Bridge:

- */fsds/imu* - zrýchlenie, rýchlosť a orientácia auta z inerciálnej jednotky
- */fsds/camera/CAMERANAME* - obraz z kamery
- */fsds/lidar/LIDARNAME* - dáta z lidaru

Na ovládanie formule v simulátore je potrebné odosielať dáta na správu: */fsds/control-command*. Táto správa umožňuje nastaviť požadované zatočenie volantu, pozíciu plynového a brzdového pedálu.

Kapitola 3

Návrh systému

Pri návrhu systému na riadenie bezpilotnej formule som mal jasné požiadavky. Systém je potrebné navrhnúť tak, aby bol modulárny a ľahko rozšíriteľný. Systém vyvíjam pre tím TU Brno racing. Ten ho bude používať viac rokov. Systém plánujeme v ďalších sezónach vylepšovať, aby sme sme sa dostali v súťažiach na najvyššie priečky. Preto som sa rozhodol systém vyvíjať nad operačným systémom ROS. Ten poskytuje výbornú modularitu v podobe ROS nodes. Každý node bude mať definovaný jeho vstup a výstup. Preto nebude problém jednotlivé moduly postupne vylepšovať bez zmeny ostatných modulov. Pre ROS som sa rozhodol aj z dôvodu, že uľahčuje komunikáciu medzi jednotlivými modulmi vďaka konceptu ROS Topics, ktoré sú popísané v teoretickej časti. Ďalšou výhodou je jednoduchá paralelizácia celého systému, keďže každý node pracuje v separátnom procese. Pre jednoduchý vývoj a testovanie celého systému je potrebné, aby ho bolo možné testovať v FS Simulátore a následne ho nasadiť na reálnu formulu bez väčších zásahov do kódu. Preto je taktiež ideálny ROS, keďže pri nasadení systému na formulu stačí pridať moduly, ktoré budú spracovávať dáta zo senzorov a odosielať ich rovnakými správami, aké odosiela FS Simulátor. To zaručí, že zbytok systému môže pracovať rovnako keď bude pracovať so simulátorom, aj keď bude nasadený na formuly.

Na obrázku B.1 je možné vidieť rozdelenie navrhnutého systému do jednotlivých modulov a správy pomocou ktorých medzi sebou komunikujú. V nasledujúcej časti je vysvetlené, ktorý modul vykonáva akú úlohu, aké správy prijíma a aké odosiela.

Kamera

Modul kamery je zodpovedný za spracovanie dát z kamier. Modul je prihlásený na odoberanie správ z ROS topicu */fsds/camera/CAMERANAME*. Vždy keď je dostupná nová snímka z kamier, modul sa spustí a začne fotku spracovávať. Na formuly sú umiestnené dve kamery. Poloha oboch kamier je známa, čo umožňuje vypočítať pozíciu kužeľa, ak je identifikovaný na oboch snímkoch. V prvom kroku je potrebné identifikovať všetky kužele na snímku z jednej kamery. Na identifikáciu kužeľov je použitá konvolučná neurónová sieť. Na neurónovú sieť sú kladené viaceré požiadavky. Výpočet pre každú snímku musí byť čo najrýchlejší, keďže každé oneskorenie v jednotlivých moduloch vedie k horšej funkčnosti celého systému. Neurónová sieť musí vedieť rozoznať aj farbu kužeľov pre správnu klasifikáciu modrých, žltých a oranžových kužeľov. Po identifikácii kužeľov v snímke z jednej kamery je vypočítané, kde približne by sa mal nachádzať kužeľ v snímku z druhej kamery. Snímok je následne rozdelený len na časti, ktoré obsahujú kužele. Je to výhodnejšie, pretože neurónová sieť nemusí prehľadávať celú snímku. Po identifikácii všetkých kužeľov na oboch snímkach

sa vypočíta ich poloha. K súradniciam je pridaná informácia o farbe kužeľu. Pole súradníc je odosielané na topic: */camera/cones/*. Ku každej správe je taktiež potrebné priradiť časovú známku, kedy boli snímky zachytené. Je to z dôvodu, že spracovanie snímkov trvá nejaký čas a vozidlo sa väčšinu času pohybuje dopredu. Keďže pozícia kužeľov sa vzťahuje k pozícii auta, je tento pohyb potrebné zahrnúť do výpočtov.

Lidar

Modul Lidar má podobnú funkciu ako modul Kamera, akurát spracováva dáta zo senzoru Lidar. Tento senzor meria vzdialenosť okolitých objektov na základe výpočtu doby šírenia laserového lúča od odrazeného objektu. Modul získava dáta z topicu */fsds/lidar/Lidar1*. Formát dát je tzv. pointcloud. Pointcloud je pole bodov v priestore. Lidar je možné nakonfigurovať, aby mal požadované zorné pole a vyhovujúci počet snímaných bodov. Algoritmus na identifikovanie kužeľov z Lidaru je odlišný od toho použitého v kamerách, keďže sú známe iba súradnice bodov, nie ich farby. Situácia je ešte zložitejšia, keďže vozidlo používa odpružený podvozok, takže uhol lidaru od zeme sa počas jazdy môže meniť. Preto je ako prvé potrebné odfiltrovať všetky body, ktoré snímali zem. Filtrované dáta je následne potrebné rozdeliť do skupín na body, ktoré sa nachádzajú blízko seba. Vďaka pravidlám súťaže Formula student je známe, že okrem kužeľov sa na trati nebudú nachádzať žiadne iné objekty. Vďaka tomu je možné predpokladať, že každá skupina bude predstavovať jeden kužeľ. Z každej skupiny bodov je vypočítaný ich stred. Súradnice kužeľov sú odosielané na topic: */lidar/cones/* spolu s časovou známkou kedy bolo meranie uskutočnené, z rovnakého dôvodu ako v module kamera.

Mapovanie

Hlavnou úlohou mapovania je zostaviť mapu všetkých kužeľov, ktoré vozidlo zatiaľ identifikovalo v globálnom súradnicovom systéme. Modul odoberá správy z topicu */lidar/cones/* a */camera/cones/*. Keďže senzory vozidla nie sú presné, modul musí správne priradiť nové kužele ku kužeľom, ktoré boli detekované v predošlích meraniach. Pokiaľ je nový kužeľ v blízkosti starého kužeľa na globálnej mape, musí ho k nemu asociovať a upresniť jeho pozíciu. V opačnom prípade pridá na globálnu mapu nový kužeľ. Modul ďalej využíva informácie o pohybe vozidla, aby zistil, o koľko sa vozidlo posunulo medzi jednotlivými meraniami. Následne aktualizuje pozíciu vozidla na globálnej mape tak, aby odpovedala realite. Informácie o polohe kužeľov sú presnejšie z lidaru, ale farbu kužeľov dokáže získať iba z kamier. Využíva sa váhovanie tak, že pozícia rovnakého kužeľa z lidaru je uprednostnená pred pozíciou kužeľa z kamier. Vždy, keď modul spracuje nové dáta zo senzorov a upraví globálnu mapu, odošle pole súradníc a farieb kužeľov pomocou správy */mapping/globalMap/*. Počas disciplíny akcelerácia a skidpad sú známe pozície kužeľov dopredu. V tomto prípade sa modul snaží podľa prvých nájdených kužeľov správne prispôsobiť pozíciu celej pred pripravenej mapy. Vďaka tomu môže vozidlo hneď od začiatku ísť plnou rýchlosťou, keďže pozná celú trať.

Výpočet ideálnej trasy

Výpočet ideálnej trasy sa rozdeľuje na dva režimy operácie. Na začiatku disciplíny jazda po trati je potrebné prejsť celú trať tak, aby bola uložená v globálnej mape. Modul prijíma pozície identifikovaných kužeľov pomocou správy */mapping/globalMap/* a v reálnom čase sa snaží vypočítať trasu, po ktorej má vozidlo ďalej ísť. Využíva sa jeden z algoritmov

na detekciu trate 2.3, podľa toho či je dostupná farba kužeľov z kamier. Algoritmus sa snaží nájsť cestu po strede trate, aby sa minimalizovala šanca na zrazenie kužeľa. Vždy, keď sa vypočíta aktualizovaná trasa, po ktorej má vozidlo ísť, spustí sa algoritmus, ktorý detekuje či vozidlo neprešlo celé kolo po trati. Ak áno, znamená to, že je načítaná celá trasa v globálnej mape a môže sa začať počítať ideálna trasa po trati. Modul odosiela správu: */racingLine/waypoints/*, čiže usporiadaný zoznam súradníc bodov v globálnej mape, po ktorých má formula ísť. V momente kedy je detekovaný koniec kola, modul začne počítať ideálnu trasu po trati. Ku každému bodu je následne vypočítaná aj maximálna rýchlosť, ktorou vozidlo dokáže danú časť trate prejsť. Po vypočítaní sú tieto body odoslané aj s rýchlosťami v rovnakej správe. V tomto momente prestane modul riadenia vozidla jazdiť konštantou rýchlosť a začne sa rýchle kolo. Situácia sa mení pri disciplíne akcelerácia, kedy je trasa definovaná dopredu, takže ju nie je potrebné počítať. V tejto disciplíne je cieľom dosiahnuť maximálnu rýchlosť vozidla, takže nie je potrebné počítať ani rýchlosť. Modul v tomto režime len preposiela body na trase modulu riadenia vozidla. V disciplíne skidpad je definovaná trasa, po ktorej má formula ísť, stále je však potrebné vypočítať maximálnu možnú rýchlosť v zákrutách a taktiež brzdné dráhy tak, aby formula zastavila v požadovanom mieste.

Riadenie vozidla

Modul riadenia vozidla je zodpovedný za ovládanie motorov, bŕzd a motor riadenia tak, aby sa formula držala na vopred určenej trati. Modul je prihlásený na odber správy */racingLine-/waypoints/*, z nej získava body, po ktorých sa má formula pohybovať. Modul získava dáta o polohe vozidla zo správy */fsds/imu*. V každom cykle sa pozrie aké je momentálne vychýlenie formule od požadovanej dráhy. Z odchýlky je vypočítané zatočenie kolies, ktoré je potrebné pre pohyb po definovanej trase. Počas prieskumného kola je nastavená konštantná rýchlosť vozidla, ktorú udržiava PID regulátor. Po ukončení tohto kola získava aktualizovanú trasu z modulu výpočet ideálnej trasy, ktorá obsahuje taktiež teoretickú maximálnu rýchlosť v každom bode. PID regulátoru pre plyn a brzdu začneme dávať ako cieľ rýchlosť v nasledujúcom bode trasy. Vozidlo začne jazdiť rýchlosťou, aká bola vypočítaná modulom výpočet ideálnej trasy. Počas ďalších kôl získava dáta o zrýchlení vozidla z inerciálnej jednotky. Počas rýchlych kôl je úlohou modulu využívať maximálny možný potenciál vozidla, a teda jazdiť čo najrýchlejšie. Modul pozná maximálne možné zrýchlenie, ktoré dokážu pneumatiky preniesť na vozovku. V každom úseku trate monitoruje aktuálne zrýchlenie. Ak je v niektorých momentoch aktuálne zrýchlenie menšie ako maximálne, algoritmus zvýši maximálnu možnú rýchlosť a vozidlo prejde v ďalšom kole túto časť trate rýchlejšie. Algoritmus taktiež monitoruje vychýlenie vozidla od vypočítanej trasy. Ak sa vozidlo vychýli, znamená to, že boli presiahnuté limity a vozidlo sa nachádza v nedotáčavom alebo pretáčavom šmyku. V danom úseku algoritmus zníži maximálnu možnú rýchlosť v danom úseku a vozidlo pôjde v ďalšom kole pomalšie. Modul odosiela požiadavky na ovládanie kolies, motorov a bŕzd do simulátora pomocou správy */fsds/controlcommand*. V realite bude túto správu prijímať riadiaca jednotka vozidla, ktorá motory ovláda.

Kapitola 4

Implementácia a testovanie

Na implementáciu systému som sa rozhodol použiť programovací jazyk Python a operačný systém ROS. Python chcem použiť hlavne kvôli knižniciam na spracovanie obrazu ktoré ponúka a taktiež sa v ňom programuje jednoduchšie ako v iných programovacích jazykoch. ROS podporuje programovanie aj v jazyku C++. Programy napísané v jazyku C++ sú síce menej náročné na hardware oproti jazyku Python, ale rozhodol som sa uprednostniť menej náročné programovanie. Operačný systém ROS umožní hociktorý modul prepísať do jazyka C++, ak by bolo potrebné rýchlejšie spracovanie danej časti systému. Testovanie bude prebiehať v prvej časti hlavne v simulátore FS Driverless simulátor, ktorý poskytuje všetky potrebné nástroje na vývoj systému. Neskôr, keď bude systém fungovať v simulátore, začne sa pripravovať na vozidle tímu TU Brno racing, Dragon E2. Toto vozidlo bude ako prvé pripravené na jazdu bez pilota. V čase písania tejto práce je však vozidlo stále vo vývoji a zatiaľ na ňom testovanie bezpilotného systému nie je možné. Implementácia sa zameriava na modul výpočtu ideálnej trasy a modulu riadenia vozidla. Ostatné moduly boli vytvorené kolegami v tíme TU Brno Racing, keďže ich funkcia nie je obsahom zadania tejto práce.

4.1 Výpočet ideálnej trasy

V nasledujúcej sekcii práce sa zameriavam na implementáciu jednotlivých algoritmov pre výpočet trasy formule počas prieskumného kola a následne aj trasu pre rýchle kolo. Pri implementácii som sa snažil dbať na čo najnižšiu výpočtovú náročnosť všetkých algoritmov, keďže do formule je ideálne dať počítač čo najmenších rozmerov, z dôvodu šetrenia hmotnosti.

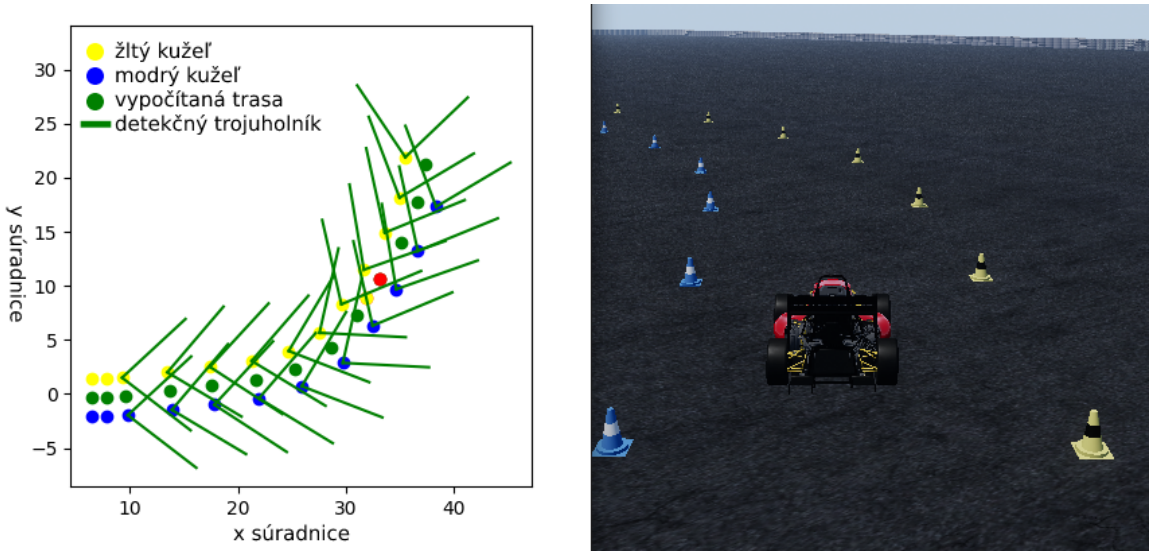
Prieskumné kolo

Hľadanie trasy počas prieskumného kola je prvá časť každej jazdy vozidla v disciplíne jazda traťou. Pre výpočet ideálnej trasy a rýchlosti vozidla, je najskôr potrebné prejsť celú trať a uložiť ju do globálnej mapy. Úlohou tejto časti je v reálnom čase počítať trasu vozidla tak, aby nevyšlo mimo trať pokiaľ budú senzory detekovať trať pred vozidlom. Na výpočet trasy počas prieskumného kola som sa rozhodol implementovať dva algoritmy. Prvý algoritmus nepoužíva informáciu o farbe kužeľov. Tento algoritmus je vhodné použiť v prípade, že nastane výpadok kamerového systému. Prípadne sa môže stať, že za nepriaznivého počasia (napr. silný dážď alebo ostré slnko) nebudú kamery schopné identifikovať kužele a vozdilo sa bude musieť spoliehať iba na dáta z lidaru. Lidar nedokáže identifikovať farbu kužeľov. Druhý algoritmus využíva informáciu o farbe kužeľov, aby určil trasu po trati. Informácia o

farbe kužeľov je veľmi výhodná, keďže v pravidlách súťaže je definované, že kužele na pravej strane budú modrej farby a na strane ľavej žltej farby. Tento algoritmus sa bude využívať v prípade, že budú fungovať kamery a lidar, a aj pri výpadku lidaru.

Algoritmus bez použitia farieb kužeľov

Podľa pravidiel súťaže sa vozidlo pri štarte autonómneho systému nachádza vnútri trate tak, že po jeho pravej strane sa nachádzajú modré kužele a po ľavej strane žlté. Na začiatku algoritmu využije túto skutočnosť a začne hľadať najbližší kužeľ na pravej a ľavej strane od vozidla. Tieto kužele sú uložené ako prvý modrý a žltý kužeľ. Na súradniciach kužeľa je vytvorená detekčná zóna v tvare trojuholníka, ktorá je zobrazená na grafe 4.1. Smer detekčného trojuholníka je v prvom kroku rovnaký ako smer, v ktorom stojí vozidlo. Následne je upravený podľa toho, ako sa zatáča trať. Algoritmus následne prechádza cez všetky kužele ktoré ešte neboli priradené na ľavú alebo pravú stranu, a pomocou algoritmu na zistenie či sa bod nachádza vnútri trojuholníka hľadá kužeľ, ktorý definuje ďalšiu časť trate. Ak leží vnútri trojuholníka viacero kužeľov, vyberie sa ten, ktorý leží najbližšie k pôvodnému kužeľu. Tvar trojuholníka, pomocou ktorého je identifikovaný nový kužeľ môžem upravovať. Zmenou dĺžky strán a uhlu, ktorí zvierajú je možné detekciu prispôsobiť na rôzne typy tratí.

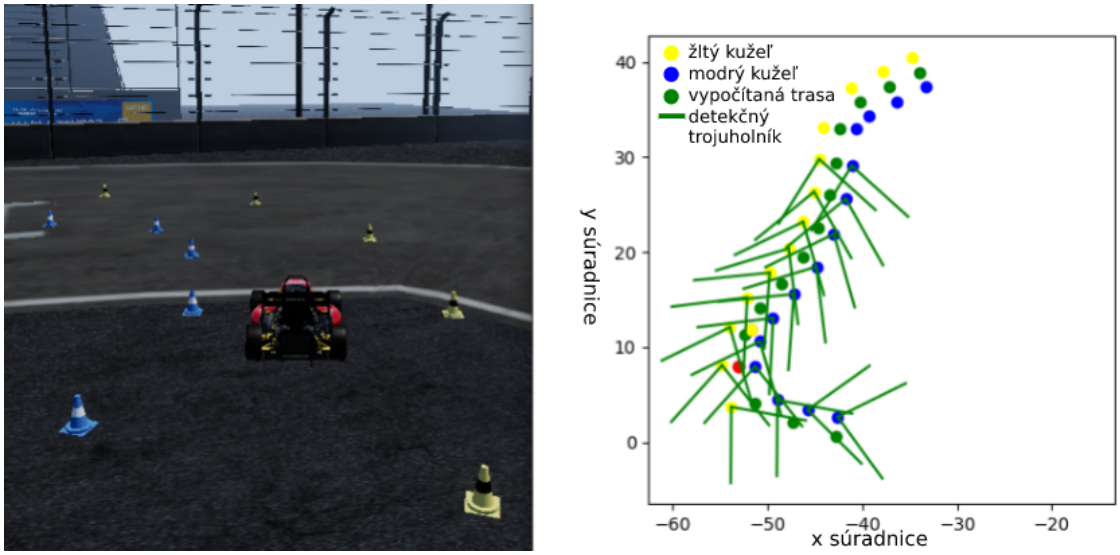


Obr. 4.1: Detekcia kužeľov počas jazdy traťou.

Problém nastáva v ostrých zákrutách, ďalší kužeľ sa väčšinou nachádza mimo detekčného trojuholníka. Trojuholník je potrebné rotovať podľa zakrivenia zákruty tak, aby nasledoval zákrutu. Trojuholník je rotovaný pomocou rotačnej matice. Trojuholník je rotovaný o uhol, ktorí zvierajú úsečky posledných troch bodov vypočítanej trasy. Rotácia trojuholníkov je znázornená na grafe 4.2.

Počas prieskumného kola sa môže stať, že senzory neidentifikujú niektorý z kužeľov na jednej strane trate. Ak sa podarilo identifikovať kužeľ na strane druhej, chýbajúci kužeľ je doplnený na predpokladané miesto, kde by sa mal nachádzať. To umožňuje identifikovať ďalšie kužele na trati rovnakým spôsobom.

Na úspešné riadenie vozidla nie je potrebné identifikovať viac, ako 4 nové kužele na každej strane pred vozidlom. Ak algoritmus identifikuje aspoň 4 kužele, tak sa prehľadávanie ukončí kvôli šetreniu výpočtového výkonu. Po rozdelení nových kužeľov do dvoch



Obr. 4.2: Detekcia trate s ostrou zákrutou.

skupín algoritmus vypočíta stredové body medzi nimi. Tieto stredové body určujú novú trasu pre vozidlo. Následne sú body odoslané modulu riadenie vozidla pomocou správy: */racingLine/waypoints/*.

Algoritmus je zavolaný vždy, keď je aktualizovaná globálna mapa. Po aktualizácii je potrebné prepočítať trasu pred autom, pretože mohli byť identifikované nové kužele, ktoré budú trať upravovať. Trasu, ktorú vozidlo už prešlo nie je potrebné meniť. Môžeme predpokladať, že do tohto momentu išlo vozidlo po správnej trase. Ďalší cyklus algoritmu začína v mieste, kde sa momentálne nachádza vozidlo. Algoritmus sa snažíme znovu vypočítať trasu, po ktorej má vozidlo pokračovať.

Algoritmus s použitím farieb kužeľov

Pokiaľ funguje rozpoznávanie kužeľov pomocou kamier, je možné využiť informáciu o farbe kužeľa na zlepšenie hľadania trasy po trati. Na začiatku algoritmus hľadá najbližší modrý a žltý kužeľ k vozidlu. Využíva sa predpoklad, že vozidlo stojí vo vnútri trate. Vypočítaný stred týchto dvoch kužeľov určuje začiatok trasy. Kužele, z ktorých sú vypočítané body na trase označíme ako už spracované, aby neboli použité znovu. Ak je nájdený prvý bod trasy, hľadá sa nový, ešte nespracovaný kužeľ, ktorý je najbližšie k momentálnej trase vozidla. Po nájdení sa skontroluje farbu kužeľu, ak je nový kužeľ modrý, algoritmus vypočíta nový bod trasy medzi posledným nájdeným žltým kužeľom a novým modrým.

Tento algoritmus je cyklicky volaný pokiaľ nie sú spracované všetky kužele alebo, aby bol ušetrený výpočtový výkon, pokiaľ neexistujú aspoň 4 nové traťové body pred vozidlom. Vždy, keď príde aktualizácia globálnej mapy, odstránia sa traťové body, ktoré sa nachádzajú pred vozidlom a algoritmus sa spustí znovu. Trasu je potrebné vždy prepočítať, pretože mohli byť identifikovať nové kužele, ktoré menia trasu, po ktorej má vozidlo ísť.

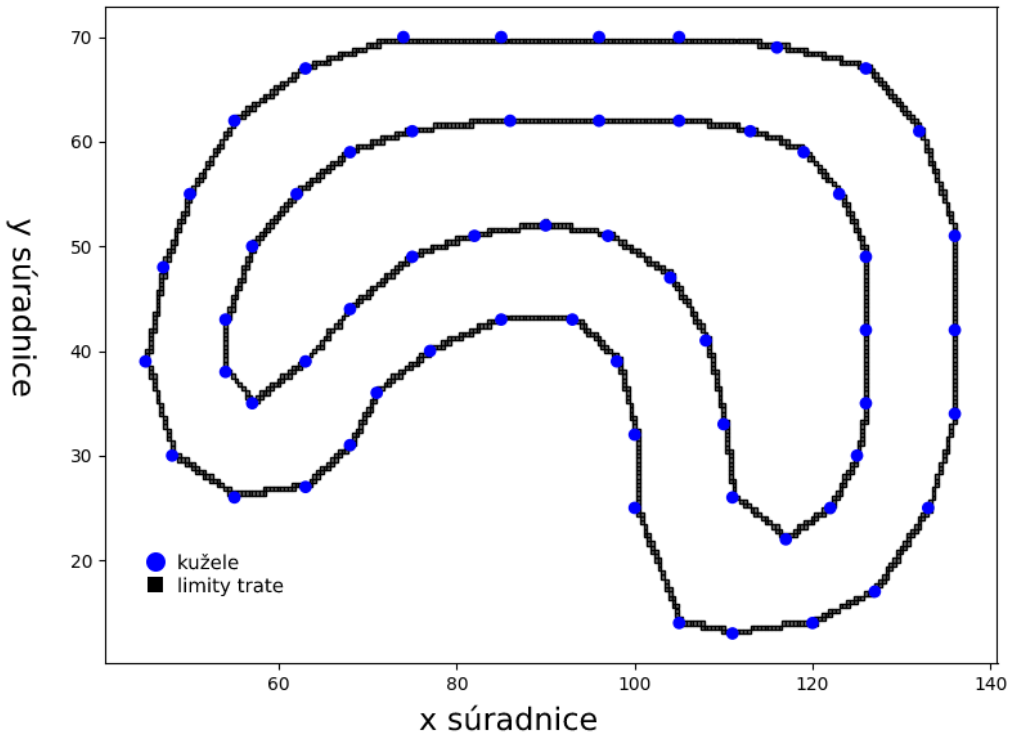
Algoritmus s použitím farieb kužeľov je oveľa jednoduchší na implementáciu, aj na výpočtový výkon. Oproti algoritmu bez použitia farieb odpadá povinnosť ladiť veľkosť detekčných trojuholníkov v závislosti od trate. Najväčšou nevýhodou algoritmu je, že sa spolieha na to, že všetky kužele budú mať korektne detekovanú farbu. Jeho úspešnosť je teda závislá na kvalite neurónovej siete použitej na rozpoznanie kužeľov z kamier. Taktiež je jeho

úspešnosť závislá na počasí. V prípade ostrého slnka svietiaceho do kamery alebo v prípade dažďa nemôžme očakávať úspešné rozpoznanie kužeľov z kamier.

Hľadanie traťových limitov

Nájdenie traťových limitov je dôležité pre detekciu či vozidlo nevyšlo mimo trate, a taktiež pri počítaní závodnej stopy po trati. Závodná stopa musí využívať maximálnu možnú šírku trate, aby sme dokázali optimalizovať prejazd zákrutami.

Po prejdení prvého kola po trati je postavená globálna mapa so súradnicami všetkých kužeľov. Na základe globálnej mapy je vypočítaný minimálny rámček, ktorý ohraničuje všetky kužele. V rámčeku je vytvorená 2D mriežka, ktorá je v programe reprezentovaná pomocou 2D poľa. Vďaka algoritmom na nájdenie trasy počas prieskumného kola je dostupný zoznam modrých a žltých kužeľov usporiadaných podľa smeru trate. Z usporiadaného zoznamu algoritmus zoberie vždy dva susedné body a vyfarbí všetky štvorčeky v mriežke, ktoré ležia na úsečke medzi týmito bodmi. Po prejdení všetkých bodov v zozname sú v mriežke 4.3 vyznačené limity trate.



Obr. 4.3: Testovacia trať s vyznačenými limitmi.

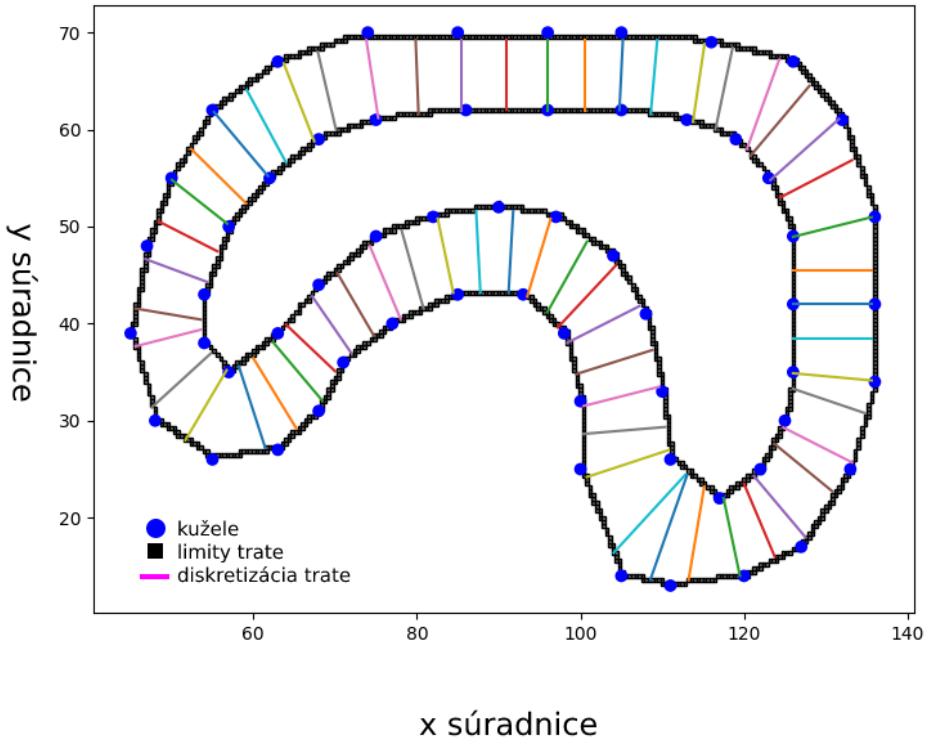
Na vyznačenie celého vnútra trate je možné použiť seed fill[1] algoritmus. Po vyznačení celej trate je jednoduché určiť či sa vozidlo nachádza na trati alebo mimo trate. Pomocou mapovacej funkcie je vypočítaný z globálnej pozície vozidla korešpondujúci index v poli. Podľa hodnoty prvku sa určí či je vozidlo na trati, alebo mimo trate. Algoritmus je veľmi

[1] Algoritmus na identifikáciu ohraničenej plochy vo viac rozmernom poli

efektívny, na overenie či je vozidlo na trati je potrebný len jeden prístup do pamäte a porovnanie jednej podmienky. Vďaka tomu je možné overiť či je vozidlo na trati pri každej aktualizácii polohy vozidla.

Diskretizácia trate

Po úspešnom nájdení traťových limitov sa trať parametrizuje tak, aby bolo možné úpravou parametrov meniť trasu po trati. Na trati je potrebné vytvoriť priečne úsečky definované parametrickým vyjadrením priamky 2.2. Ohraničenie trate algoritmus hľadá tak, že postupne mení parameter t priamky, až kým bod nebude ležať na vyfarbenom políčku v mriežke limitov trate. Týmto spôsobom je vypočítané minimálne a maximálne t pre ktoré bude bod na priamke ležať na trati. Po definovaní všetkých priečnych úsečiek sa začína počítať optimálna trasa po trati.



Obr. 4.4: Diskretizácia testovacej trate.

Výpočet rýchlosti vozidla

Ďalšou úlohou modulu výpočet trasy je vypočítať rýchlosť, ktorou môže vozidlo po trati prejsť. Po prejdení prvého kola, v ktorom jazdí vozidlo konštantou rýchlosťou pozná celú trať. Vďaka globálnej mape je možné vypočítať teoretickú maximálnu rýchlosť, ktorou môže vozidlo ísť v každom bode trasy. Táto časť práce sa zaoberá hlavne výpočtom maximálnej rýchlosti, ktorú dokáže vozidlo udržať v zákrute. Zrýchľovaním na rovinkách a brzdením pred zákrutami sa bude zaoberať modul riadenia vozidla.

Teoretickú maximálnu rýchlosť je potrebné vypočítať pre každý bod trasy. Ako prvý sa zistí polomer zákruty v každom bode na trase. Polomer je vypočítaný zo skupiny bodov na trase, ktoré sú preložené kružnicou. Na nájdenie kružnice, ktorá najlepšie definuje body je použitá metóda najmenších štvorcov. Polomer zistenej kružnice je priradený k prostrednému bodu zo skupiny. Tento postup sa opakuje pre všetky body trasy. Po zistení polomerov je vypočítaná teoretická maximálna rýchlosť pre každý bod.

V zákrutách funguje táto metóda správne, na rovinkách je polomer kružnice nekonečný, to znamená, že aj vypočítaná maximálna rýchlosť bude nekonečná. To nevadí, keďže rýchlosť na rovinkách upravuje modul riadenia vozidla.

Optimálna trasa po trati

Nájdenie optimálnej trasy je zložitá optimalizačná úloha z dôvodu, že existuje veľa parametrov, ktoré je potrebné brať do úvahy. Vďaka diskretizácii trate je možné jednoducho meniť trasu po trati zmenou jedného parametru pre každú úsečku. Pri hľadaní optimálnej trasy je nutné zohľadniť kompromis medzi bezpečnosťou a rýchlosťou vozidla. Čím bližšie ku kužeľom bude trasa smerovať, tým sa zväčší šanca, že vozidlo do kužeľov nabúra, čo by znamenalo časovú penalizáciu na pretekoch. Z dôvodu, že tento systém riadenia nie je dokonalý, rozhodol som sa uprednostniť bezpečnú vzdialenosť medzi trasou a kužeľmi.

$$
\begin{aligned}
a_x &= p1_x - p2_x \\
a_y &= p1_y - p2_y \\
b_x &= p2_x - p3_x \\
b_y &= p2_y - p3_y \\
y &= \arccos \frac{a_x * b_x + a_y * b_y}{\sqrt{a_x^2 + a_y^2} * \sqrt{b_x^2 + b_y^2}} * \alpha + (\sqrt{a_x^2 + a_y^2} + \sqrt{b_x^2 + b_y^2}) * \beta
\end{aligned}
\tag{4.1}
$$

$p1$, $p2$, $p3$ sú body na priečnych úsečkách

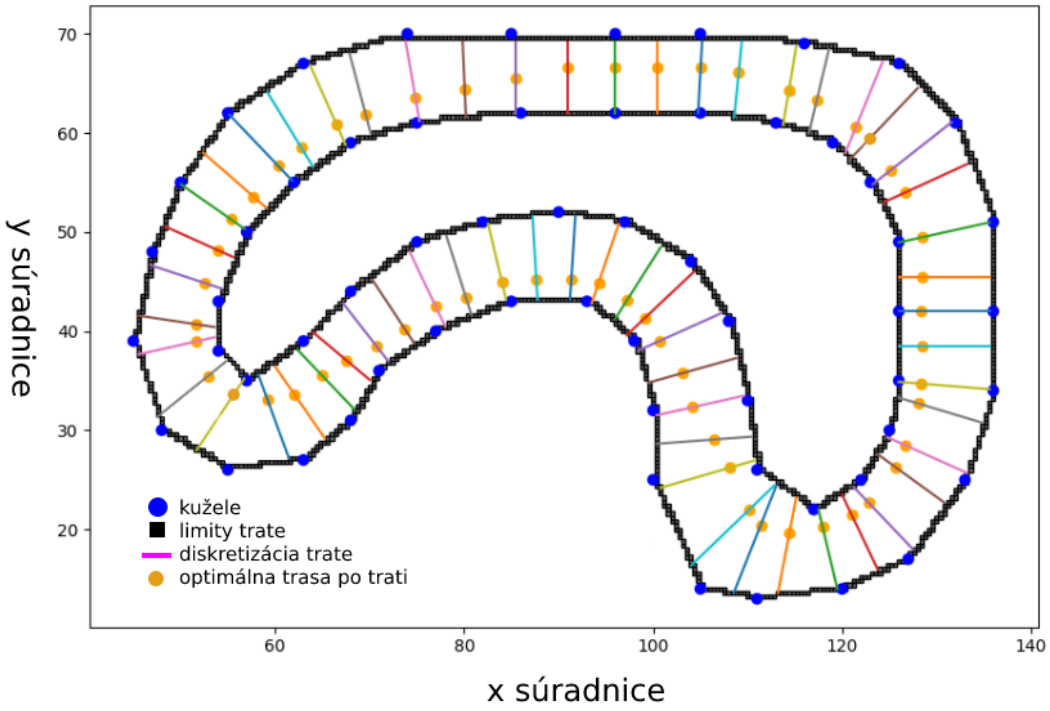
α a β sú ladiace parametre funkcie

y je výsledná cena segmentu trate

Algoritmus na hľadanie ideálnej trasy sa vždy pozerá na tri po sebe idúce priečne úsečky, mení ich parameter t až kým nebude mať výsledok vzorca 4.1 na výpočet ceny segmentu trate najmenšiu hodnotu. Po zistení parametrov pre všetky priečne úsečky je vypočítaný teoretický čas potrebný na prejdenie kola. Následne sú upravené parametre vo vzorci na určenie ceny segmentu trate a algoritmus sa spúšťa znovu. Trať, ktorá bude mať najmenší čas na prejdenie jedného kola je následne použitá.

4.2 Riadenie vozidla

Modul riadenie vozidla je zodpovedný za fyzické ovládanie motorov vozidla, aby vozidlo nasledovalo vopred vypočítanú trasu. Modul musí ovládať servo-motor otáčania kolies, motor na ovládanie bŕzd a posielať dáta riadiacej jednotke pohonnej sústavy. Modul prijíma správu */racingLine/waypoints*, z ktorej získava trasu pre vozidlo. Ku každému bodu je priradená teoretická maximálna rýchlosť, ktorou môže vozidlo v danom bode ísť. Informácie o pozícii vozidla a rýchlosti modul získava zo správy */fsds/imu*.



Obr. 4.5: Testovacia trať s vyznačenou optimálnou trasou.

PID regulátor

Existuje viacero spôsob na reguláciu rýchlosti a ovládania kolies vozidla. V tejto práci som sa rozhodol použiť PID regulátor z dôvodu jednoduchosti implementácie. Pri iných formách riadenia je potrebné mať komplexný fyzikálny model vozidla.

Riadenie zatáčania

Na riadenie zatáčania je použitý osobitný PID regulátor. Regulátoru vyžaduje na vstupe odchýlku od požadovaného stavu. V teoretickej časti sú popísané dve možnosti na výpočet odchýlky vozidla od žiadanej trasy. Počas testovania som vyskúšal a porovnal obe možnosti na výpočet odchýlky.

Prvý spôsob výpočtu odchýlky je počítať uhol medzi vektorom smeru jazdy vozidla a vektorom medzi polohou vozidla a nasledujúcim bodom na trase. Uhol vypočítame pomocou funkcie:

$$\alpha = arctg2(A_y, B_x) - arctg2(A_y, B_x) \tag{4.2}$$

A je vektor smeru jazdy vozidla

B je vektor medzi pozíciou vozidla a ďalším bodom na trase

α je uhol medzi dvomi vektormi

Funkciu arctg2 používame, lebo nám vracia kladný uhol ak je vozidlo vychýlené od bodu v smere hodinových ručičiek a záporný uhol ak je vychýlené v protismere. To je pre vstup do PID regulátora ideálne, podľa znamienka vie určiť, na ktorú stranu má kolesá zatočiť.

Druhý spôsob na určenie odchýlky je vypočítať tzv. cross track error 2.5. CTE nám určuje vzdialenosť od úsečky medzi dvoma bodmi na trati. Problém tejto metódy je, že CTE nám určuje iba vzdialenosť od úsečky. Na úspešné riadenie je potrebné vedieť, na ktorej strane od úsečky sa nachádzame. Vzájomnú polohu úsečky a vozidla zistíme podľa vzorca:

$$\beta = (C_x - A_x) * (B_y - A_y) - (C_y - A_y) * (B_x - A_x) \tag{4.3}$$

C sú súradnice vozidla

A sú súradnice bodu A úsečky

B sú súradnice bodu B úsečky

β kladný výsledok - vozidlo na jednej strane úsečky, záporný výsledok - vozidlo na druhej strane úsečky

Úlohou PID regulátora riadenia je udržiavať odchýlku vždy čo najbližšie k nule. Vtedy sa bude vozidlo pohybovať po zadanej trase. Problémom PID regulátora je potreba nastaviť konštanty K_P, K_I a K_D. Tieto konštanty sú špecifické pre každý systém, ktorý sa pokúša riadiť. Definujú, ako bude regulátor reagovať na zmenu výchylky. Existuje viacero metód na zjednodušenie ladenia týchto konštánt, ale žiadna z nich nie je spoľahlivá pre všetky systémy. Konštanty som ladil počas testovania na základe zozbieraných dát z telemetrie.

Riadenie rýchlosti

Riadenie rýchlosti pracuje na podobnom princípe ako riadenie zatáčania. Úlohou PID regulátora je, aby vozidlo išlo požadovanou rýchlosťou. Rozdiel oproti PID regulátoru riadenia je, že na ovládanie rýchlosti je potrebné nastavovať výkon motorov pre zrýchlenie a brzdnú silu pre spomalenie. Vozidlo sa správa rozdielne pri zrýchľovaní a spomaľovaní, preto sú použité dva rôzne PID regulátory, jeden na riadenie výkonu motor a druhý na riadenie brzdnej sily. Pri použití dvoch PID regulátorov môže nastať nežiadúci jav, že sa vozidlo snaží zrýchľovať a brzdiť zároveň. Tento jav som pozoroval hlavne keď sa regulátory snažili udržať konštantnú rýchlosť. Je to nechcený jav, v praxi by to znamenalo preťaženie bŕzd a následnú nemožnosť zabrzdiť. Tento jav sa mi podarilo obmedziť lepším nastavením oboch PID regulátorov a zároveň som pridal podmienku, aby vozidlo nikdy nemohlo brzdiť a pridávať plyn naraz. Modul vozidlu odosiela požiadavku na nastavenie brzdnej sily a výkonu motorov pomocou správy: *fsds/controller*.

Prispôsobenie rýchlosti aktuálnemu stavu vozidla

Z modulu optimálnej trasy po trati je dostupná informácia o maximálnej možnej teoretickej rýchlosti vozidla v každej časti trate. Túto rýchlosť treba ďalej upraviť, aby zodpovedala reálnemu stavu vozidla. V prípade, že vozidlo môže v zákrute ísť podstatne vyššou rýchlosťou ako je jeho aktuálna rýchlosť nemôžeme dovoliť, aby PID regulátor nastavil plný výkon na oba motory, lebo by to mohlo znamenať preťaženie pneumatík a vozidlo by sa mohlo dostať mimo trať. Požadovanú rýchlosť vozidla, ktorú dostáva PID regulátor na vstupe je potrebné upraviť tak, aby bolo vozidlo pod maximálnym limitom, ktorí zvládnu pneumatiky.

V prvej časti sa algoritmus pozerá na body na trati, ktoré sú pred vozidlom. V prvom rade sa kontroluje minimálna rýchlosť v nasledujúcich bodov trate. Algoritmus musí overiť, že túto rýchlosť vie vozidlo dosiahnuť tesne pred tým než sa dostane k danému bodu. Algoritmus má nastavenú maximálnu možnú deceleráciu vozidla pri brzdení. Ak by bola aktuálna rýchlosť vozidla vyššia ako rýchlosť v bode X a dráha od aktuálnej polohy vozidla k bodu X menšia ako maximálna možná decelerácia vozidla, vozidlo by nedobrzdilo do zákruty a vyletelo by z trate. Preto algoritmus prechádza cez body na trase, ktoré sú pred vozidlom, nájde bod s minimálnou rýchlosťou a overí, či vozidlo dokáže na danej vzdialenosti dostatočne spomaliť. Ak nie, zníži rýchlosť na všetkých predchádzajúcich bodoch trasy tak, aby vozidlo zabrzdiť zvládlo. Keď sú rýchlosti vo všetkých bodoch pred vozidlom upravené tak, že vozidlo zvládne spomaliť pred zákrutou, spustí sa podobný algoritmus pre zrýchľovanie. Ak je maximálna rýchlosť v bodoch pred vozidlom vyššia ako aktuálna rýchlosť, upravia sa maximálne rýchlosti na hodnoty, ktoré dokáže vozidlo dosiahnuť pri plnej akcelerácii. Úlohou algoritmu je nastaviť rýchlosti tak, aby vozidlo zrýchľovalo na všetkých miestach, kde zrýchľovať môže a aby stihlo spomaliť pred zákrutou.

Druhá časť systému sa zaoberá maximálnou silou, ktorú pneumatiky dokážu preniesť na cestu. Môže sa stať, že vozidlo vychádza zo zákruty a v ďalšom bode na trase chce zrýchľovať, pretože za zákrutou je rovinka. Ak by vozidlo začalo pridávať plyn príliš skoro, môže sa stať, že vozidlo ešte neprestalo úplne zatáčať a preťažia sa pneumatiky. Systém sa pozerá na dáta z inerciálnej jednotky a preťaženie pôsobiace na vozidlo prepočítava na silu pôsobiacu na pneumatiky. Ak je výstup PID regulátora pre ovládanie výkonu motorov vyšší ako maximálna sila, ktorú dokážu pneumatiky preniesť, obmedzí výstup na bezpečnú hodnotu. Jedná sa o veľmi jednoduchý systém trakčnej kontroly, ktorý sa používa v moderných vozidlách.

Úprava rýchlosti na trati

Model, ktorý je použitý na výpočet rýchlosti vozidla nie je ideálny. Taktiež nie je presne známy povrch v každom bode trate. Preto sa môže maximálna sila, ktorú dokáže vozidlo preniesť pneumatikami na trať v rôznych častiach trate meniť. Pre prvé rýchle kolo po trati je vhodné vypočítať maximálnu rýchlosť vozidla konzervatívne tak, aby vozidlo nevyšlo mimo trate. Počas rýchleho kola sú sledované dáta z inerciálnej jednotky vozidla. Ak sa vozidlo dokáže udržať na definovanej trase bez väčších problémov, môže v danom bode trasy maximálnu rýchlosť mierne zvýšiť a ďalšie kolo pôjde vozidlo v danom mieste rýchlejšie. Naopak, ak sa vozidlo v určitom bode trasy príliš odchýli od definovanej trasy, znamená to, že bol prekročený limit pneumatík. V danom mieste sa mierne zníži maximálna rýchlosť vozidla. To zabráni prípadnému vypadnutiu vozidla z trate počas ďalšieho kola.

Detekcia prejdeného kola

Pre správnu funkčnosť všetkých systémov je potrebné presne vedieť kedy bolo ukončené celé kolo na trati a začína nové kolo. Po ukončení prvého prieskumného kola musí modul optimálnej trasy po trati vypočítať trasu a rýchlosť, ktorou vozidlo môže po trati ísť. Po každom ďalšom rýchlom kole musí prepočítať maximálne rýchlosti v každom bode trasy na základe úprav z minulého kola. V neposlednom rade je potrebné počítať počet prejdených kôl, aby vozidlo mohlo zastaviť po nastavenom počte kôl. Podmienkou na súťaži je, aby vozidlo zastavilo maximálne 20 metrov za cieľovou čiarou.

Algoritmus musí správne fungovať aj v prípade, že bude používať len dáta z lidaru. Ak by sa spoliehal na dáta z kamier, bola by detekcia prejdeného kola jednoduchšia. Cieľová čiara

je definovaná oranžovými kužeľmi na oboch stranách trate. Pri použití lidaru nie je dostupná informácia o farbe kužeľov. Pred štartom vozidla sú preto uložené prvé dva kužele na každej strane. Algoritmus následne kontroluje či boli pretnuté úsečky medzi poslednou pozíciou vozidla, aktuálnou pozíciou vozidla a úsečkou medzi dvomi kužeľmi. Ak áno, znamená to, že vozidlo prešlo jedno kolo. Túto informáciu model riadenia vozidla posiela pomocou správy *car-controller/lapCounter*. Ostatné moduly ktoré odoberajú túto správu môžu vďaka tomu vykonať potrebné operácie.

Po prejdení požadovaného počtu kôl je vozidlu nastavená cieľová rýchlosť na *0m/s* a vozidlo by malo bezpečne zastaviť bez toho, aby vyšlo mimo trať.

4.3 Testovanie

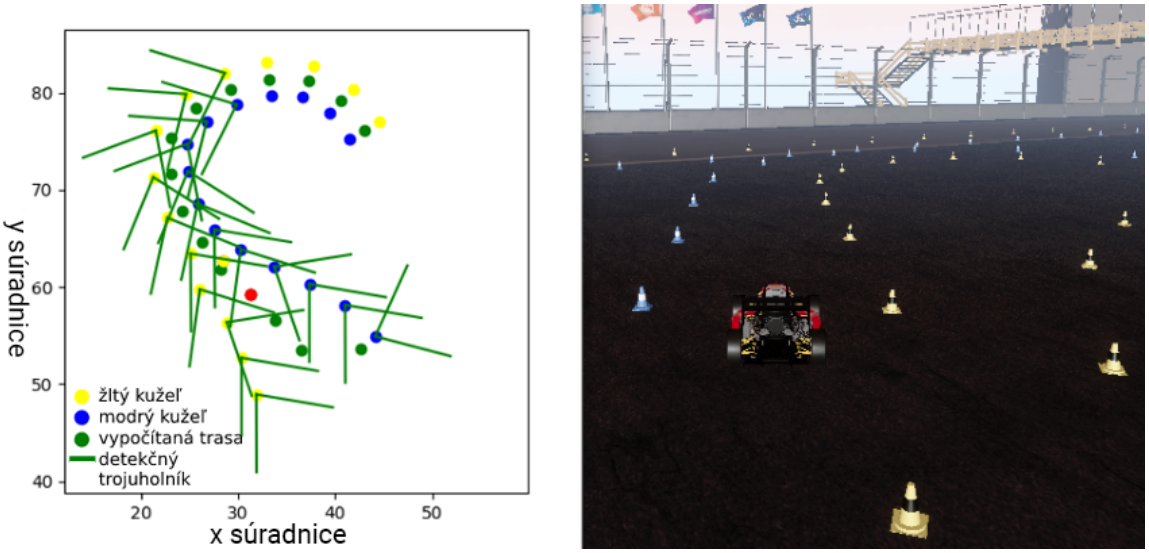
Testovanie všetkých algoritmov som uskutočnil v simulátore Formula student Driverless simulator. Ako prvé bolo potrebné otestovať algoritmy na hľadanie trasy počas prieskumného kola. V nasledujúcej časti som porovnal oba algoritmy, aby som zistil aké výhody a nevýhody má každý algoritmus. V časti riadenie vozidla som sa najskôr pokúsil odladiť PID regulátor pre riadenie tak, aby vozidlo dokázalo prejsť po trati. Následne som porovnal obe možnosti pre výpočet odchýlky pre regulátor. Po odladení regulátora riadenia je potrebné nastaviť regulátor pre zrýchľovanie a brzdenie vozidla. Keď vozidlo začalo jazdiť podľa mojich predstáv, porovnal som teoretický model vozidla s reálnymi časmi v simulátore.

Detekcia trate

Ako bolo spomenuté v minulých kapitolách, sú použité dva rôzne algoritmy na detekciu trate počas prieskumného kola. Ako prvý som vyskúšal algoritmus na detekciu trate bez použitia farieb kužeľov na rôznych tratiach a porovnal som výsledky s algoritmom, ktorý používa farby kužeľov.

Algoritmus bez použitia farieb kužeľov je potrebné pred použitím najskôr odladiť. Nastavil som veľkosti detekčných trojuholníkov tak, aby detekovali len tie kužele, ktoré majú a zároveň, aby fungovali dobre aj v ostrých zákrutách. Na väčšine tratí fungoval algoritmus bez problémov. Otestoval som aj situáciu v ktorej sa lidaru nepodarí rozpoznať jeden z kužeľov. Algoritmu to nerobí problém, ďalej bezproblémovo detekoval trať. Najväčší problém pre algoritmus je typ trate, kde sú dva rozdielne úseky trate veľmi blízko pri sebe. Vtedy bez použitia farieb kužeľov nie je možné zistiť, ktorý kužeľ je ten správny pre pokračovanie na trati. Táto situácia je zobrazená na obrázku 4.6. Algoritmus by bolo potrebné nejakým spôsobom upraviť aby takúto situáciu zvládol.

Algoritmus s použitím farieb kužeľov je jednoduchší na použitie, keďže nie je potrebné odladiť žiadne jeho konštanty. Funguje perfektne na všetkých typoch tratí, nerobí mu problém ani trať s dvoma úsekmi blízko pri sebe. Problémom tohto algoritmu je, že sa spolieha na kvalitu neurónovej siete použitej na detekciu kužeľov z obrazu kamier. V dobrom počasí funguje všetko spoľahlivo. V prípade zložitejších podmienok, napríklad nízke ostré slnko, ktoré spôsobuje, že kužele vrhajú dlhé tiene alebo v daždi nie je naša aktuálna neurónová sieť dostatočne spoľahlivá, aby správne identifikovala farbu kužeľa. Pre lepšiu neurónovú sieť je potrebné zozbierať oveľa viac dát z testovania v rôznych svetelných podmienkach, aby sme mohli neurónovú sieť lepšie natrénovať. V prípade, že neurónová sieť nesprávne identifikuje farbu viacerých kužeľov, prestane môj algoritmus fungovať. Trasa, ktorú vypočíta je nesprávna. Ideálne by bolo skombinovať oba algoritmy, aby sa v prípade neistoty neurónovej siete začal používať algoritmus bez použitia farieb kužeľov.



Obr. 4.6: Nesprávna identifikácia trate algoritmom bez použitia farieb kužeľov.

Trať	Bez farieb kužeľov	S farbami kužeľov	S farbami kužeľov, zlé svetelné podmienky
Trať 1 A.1	✓	✓	×
Trať 2 A.2	✓	✓	✓
Trať 3 A.3	×	✓	✓
Trať 4 A.4	✓	✓	×
Trať 5 A.5	✓	✓	×

Tabuľka 4.1: Porovnanie algoritmov na detekciu trate na rôznych tratiach, ktoré sú vizualizované v prílohe A.

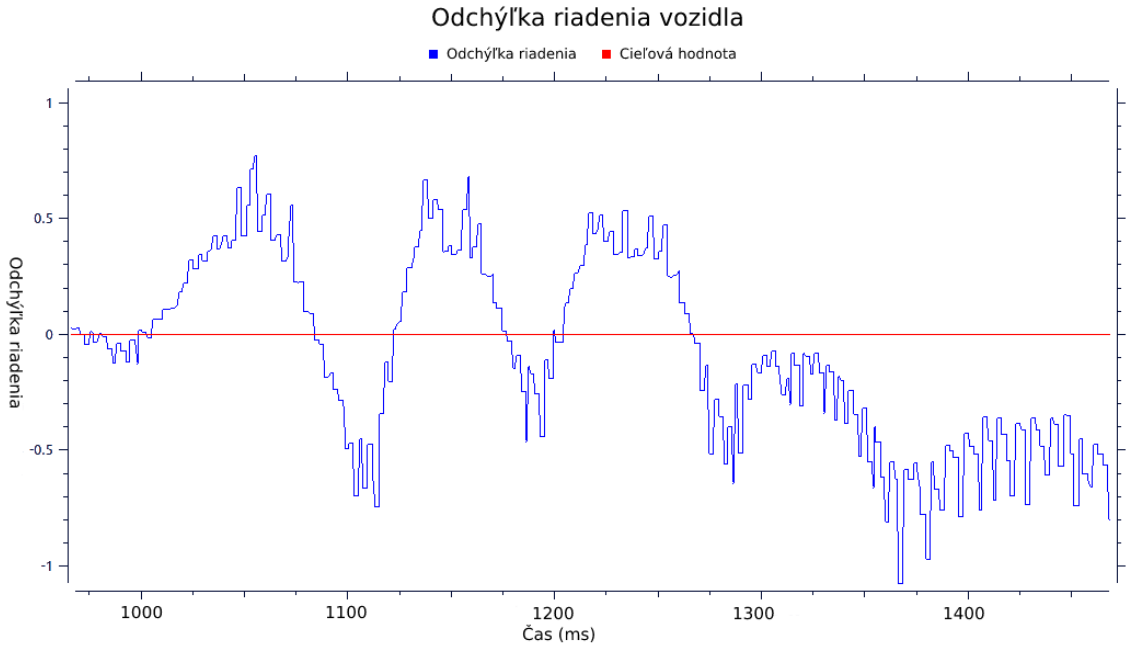
Výsledky testovania oboch algoritmov na rôznych tratiach, a za rôznych svetelných podmienok, sú zobrazené v tabuľke 4.1.

Riadenie vozidla

Mojou prvou úlohou pri testovaní riadenia vozidla bolo nastaviť PID regulátor riadenia tak, aby udržal vozidlo na definovanej trase aspoň v nízkych rýchlostiach. To mi umožnilo odladiť ostatné moduly systému. Keď ostatné moduly začali spoľahlivo fungovať, začal som ladiť PID regulátor. Na nastavenie regulačných konštánt existuje viacero metód, ale žiadna z nich nezaručuje presné nastavenie. Preto som sa rozhodol ísť cestou pokus, omyl. Pomocou telemetrie som monitoroval momentálnu odchýlku vozidla od čiary a ako sa regulátor správa. Podľa toho som nastavil proporčnú, integračnú a derivačnú konštantu.

V predošlých kapitolách sú vysvetlené dva spôsoby na výpočet odchýlky vozidla od trate. Ako prvý som sa rozhodol otestovať spôsob, kedy ako odchýlku beriem uhol medzi vektorom vozidla a ďalším bodom na trase. Pri pomalých rýchlostiach riadenie vozidla fungovalo dobre, našiel som konštanty, pri ktorých vozidlo neoscilovalo. Pri nastavovaní konštánt je cieľom mať od vozidla čo najrýchlejšiu odozvu, napríklad, keď sa dostane do zákruty, a zároveň, aby sa systém následne nerozkmital a bol stabilný. V simulátore som vybral oválnu trať, kde som testovali stabilitu systému na rovinkách a správnu odozvu v

zákrutách. Na grafe 4.7 je možné vidieť reakciu systému na prichádzajúcu zákrutu a jeho následné rozkmitanie, ktoré som nedokázal zastaviť zmenou parametrov.



Obr. 4.7: Rozkmitaný PID regulátor riadenia.

Rozhodol som sa preto otestovať druhý spôsob výpočtu odchýlky, tzv. Cross track error 2.5. Hneď po prvom teste som videl, že je vozidlo oveľa stabilnejšie na rovinkách. Následne bolo potrebné naladiť parametre tak, aby bola odozva systému dostatočne rýchla a rozkmitanie sa ustálilo. Výsledok v grafe 4.8 ukazuje prvotnú odchýlku pri nájazde do zákruty, následnú korekciu s miernym rozkmitaním, ktoré sa po chvíli ustálilo.

Z testovania je evidentné, že spôsob výpočtu odchýlky vozidla pomocou cross track error je oveľa lepší hlavne kvôli stabilite vo vyšších rýchlostiach. Tento spôsob taktiež lepšie popisuje stav vychýlenia vozidla, takže s ním PID regulátor vie lepšie pracovať.

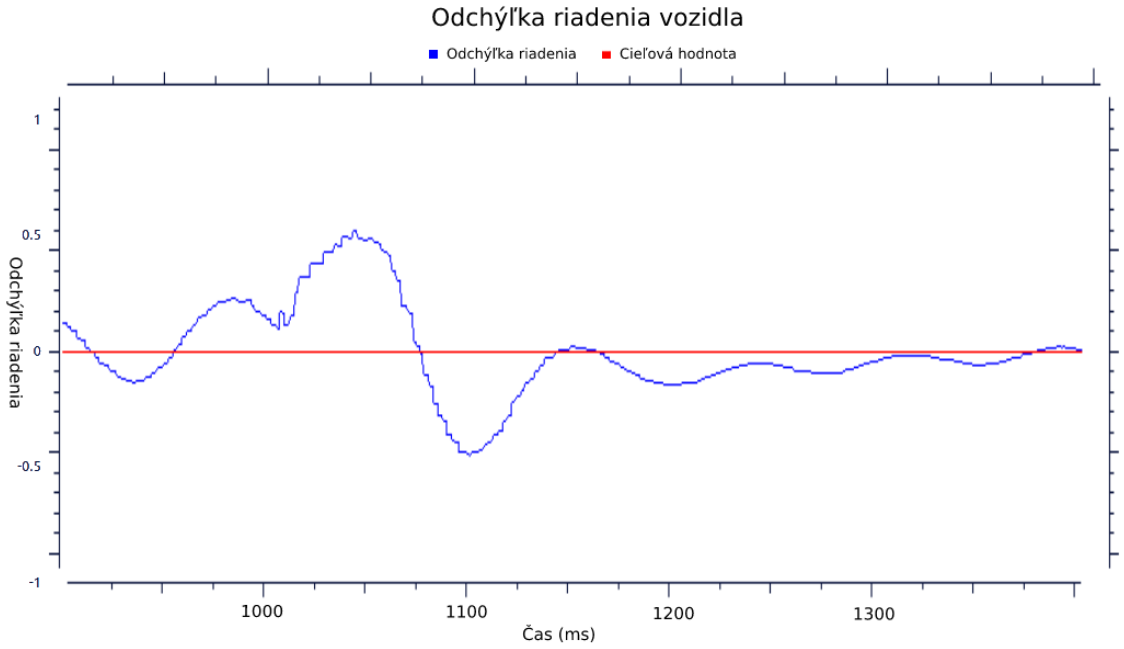
Po úspešnom nastavení parametrov pre riadenie bolo potrebné nastaviť parametre zrýchľovania a spomaľovania vozidla. Pomocou telemetrie som porovnával aktuálnu rýchlosť vozidla a požadovanú rýchlosť. 4.9. PID regulátory som nastavil tak, aby mali dostatočne rýchlu odozvu na zmeny rýchlosti a zároveň, aby cieľovú rýchlosť neprekročili.

Riadenie vozidla pomocou PID regulátora nikdy nebude ideálne. PID regulátor sa najviac hodí na lineárny systém, a to vozidlo nie je. Aj po úplnom odladení všetkých parametrov nie som s výsledkom spokojný. Kvôli zlepšeniu riadenia vozidla by bolo v budúcnosti vhodné prejsť na zložitejší spôsob regulácie, napríklad model predictive controller.

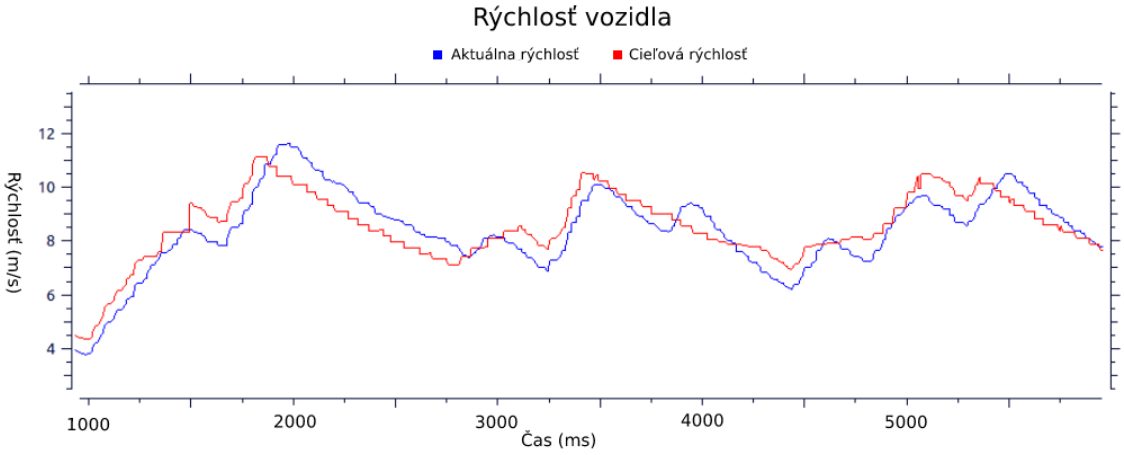
Overenie modelu vozidla

Po nastavení parametrov riadenia vozidla prišiel čas na testovanie algoritmu na úpravu rýchlosti vozidla. V teórii by vozidlo po každom prejdenom kole malo ísť rýchlejšie. Na grafe 4.10 je zobrazený čas potrebný na prejdenie každého kola. Vozidlo postupne zrýchľuje každé kolo až sa dostane k limitu, kedy sa už výrazne nezlepšuje.

Počas tohto experimentu som sa rozhodol overiť teoretický model vozidla. Z maximálnych rýchlostí, ktoré sú vypočítané pomocou modelu bol zistený teoretický čas, koľko by



Obr. 4.8: PID regulátor riadenia s odchýlkou vypočítanou pomocou cross track error.

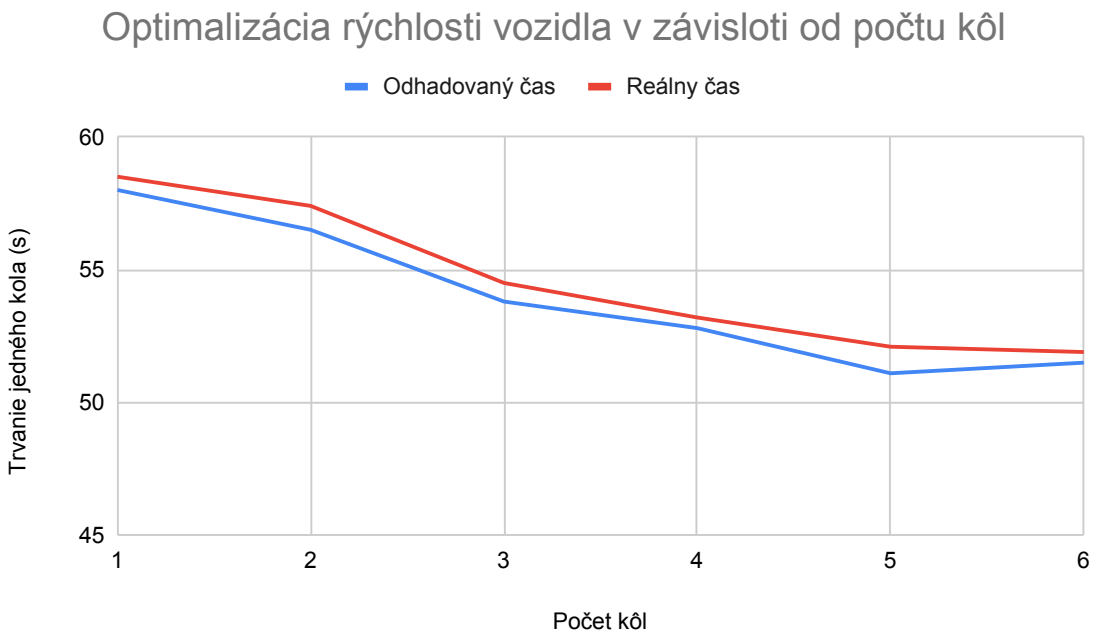


Obr. 4.9: PID regulátor na riadenie rýchlosti.

malo trvať jedno kolo. Na grafe 4.10 je porovnanie medzi teoreticky vypočítaným časom a reálnym časom v simulátore. Z tohto experimentu hodnotím, že pre účely tejto práce je model dostatočne presný.

Zhodnotenie výsledkov

Počas testovania som porovnal oba prístupy k detekcii trate. Z výsledkov vyplýva, že algoritmus, ktorý používa farby kužeľov je lepší na detekciu rôznych typov trate, ale záleží od kvality použitej neurónovej siete na detekciu kužeľov. Algoritmus bez použitia farieb kužeľov je spoľahlivejší, keď sa mení počasie na trati.



Obr. 4.10: Graf zlepšenia času potrebného na prejdenie jedného kola pri jednotlivých prejazdoch trate.

Počas testovania PID regulátora som našiel ideálne hodnoty konštánt pre PID regulátor takých, aby regulátor správne fungoval. Taktiež som porovnal rôzne možnosti na výpočet odchýlky pre PID regulátor. Z výsledkov som zistil, že pre stabilné riadenie vozidla je lepšie použiť počítanie odchýlky pomocou Cross track error.

V poslednej časti testovanie som overil teoretický model vozidla použitý na výpočet rýchlosti vozidla tak, že som porovnal teoreticky vypočítane časy potrebné na prejdenie jedného kola po trati s reálnymi časmi, ktoré boli dosiahnuté v simulátore. Z experimentu som zistil, že teoretický model je dostatočne presný pre využitie v tejto práci.

V čase písania tejto práce je formula Dragon E2 ešte vo fáze vývoja, preto nebolo možné uskutočniť testovanie na reálnej formuly, iba v simulátore. V budúcnosti bude priorita otestovať celý systém na formuly Dragon E2, prípadne upraviť jeho časti tak, aby systém vyhovoval aplikácii na reálnej formuly.

Kapitola 5

Záver

Cieľom tejto práce bolo preskúmať problematiku riadenia autonómnej formule, navrhnúť a implementovať systém na autonómne riadenie formule tímu TU Brno Racing, ktoré bude použité na súťažiach Formula Student.

Prvým krokom riešenia práce bolo preskúmať problematiku riadenia autonómneho vozidla. Keďže sa aplikácia tejto práce značne líši, oproti systémom použitých v osobných automobiloch, začal som študovať, ako k systému riadenia formule pristupujú konkurenčné tímy, ktoré sa zúčastňujú súťaže Formula Student. V teoretickej časti práce som priblížil problematiku detekcie trate, ktorá je definovaná kužeľmi. Popísal som jednotlivé spôsoby detekcie trate, porovnal ich výhody a nevýhody. Následne som popísal problematiku riadenia závodnej formulu po definovanej trati, a ako dosiahnuť najnižší možný čas potrebný na prejdenie jedného kola trate. Teoretická časť sa zameriava aj na rôzne spôsoby riadenia závodného vozidla, ktorého cieľom bolo vozidlo riadiť tak, aby bol využitý maximálny potenciál vozidla.

Výsledkom tejto práce je navrhnutý systém riadenia autonómneho vozidla, ktorý je implementovaný pomocou Robot Operating system. Systém riadenia vozidla dokáže správne identifikovať trate rôznych tvarov a naplánovať trasu po trati tak, aby formula dokázala úspešne prejsť kolo po trati. Po prejdení trate systém trasu upraví tak, aby po nej formula dokázala prejsť čo najrýchlejšie. Pomocou teoretického modelu formule systém počíta teoretickú maximálnu rýchlosťou, ktorou dokáže ísť formula po trati. Tieto dáta využíva modul riadenia vozidla a riadi formulu po vopred vypočítanej trase. Modul kontroluje dáta z formule a riadenie upravuje tak, aby sa formula držala trasy a išla určenou rýchlosťou.

Testovanie simulátoru prebiehalo v simulátore, kde som porovnal rôzne spôsoby detekcie trate pri rôznych podmienkach. Systém dokáže identifikovať väčšinu typov tratí aj pri rôznych podmienkach. V simulátore som taktiež testoval riadenie vozidla po vypočítanej trase, ktoré po správnom nastavení funguje veľmi dobre. Formula sa dokáže udržať na trati a nedostáva sa do nestabilného stavu. Počas testovania som taktiež porovnal čas potrebný na prejdenie jedného kola medzi viacerými prejazdmi trate. Algoritmus dokáže postupne upravovať rýchlosť formule na trati, aby formula dokázal trať prejsť rýchlejšie.

Po dokončení formule Dragon E2 chcem v práci pokračovať testovaním systému riadenia autonómneho vozidla v realite. Pre úspešné riadenie reálnej formule bude potrebné upraviť niektoré časti systému. Taktiež by som chcel vylepšiť reguláciu riadenia, keďže sa v testovaní potvrdilo, že použitý regulátor nie je ideálny na riadenie formule.

Literatúra

[1] Dhall, A., Dai, D. a Gool, L. V. Real-time 3D Traffic Cone Detection for Autonomous Driving. *CoRR*. 2019, abs/1902.02394. Dostupné z: http://arxiv.org/abs/1902.02394.

[2] Farag, W. Track Maneuvering using PID Control for Self-Driving Cars. *Recent Advances in Electrical Electronic Engineering (Formerly Recent Patents on Electrical Electronic Engineering)*. Január 2019, zv. 12. DOI: 10.2174/2352096512666190118161122.

[3] Farag, W. Track Maneuvering using PID Control for Self-Driving Cars. *Recent Advances in Electrical Electronic Engineering (Formerly Recent Patents on Electrical Electronic Engineering)*. Január 2019, zv. 12. DOI: 10.2174/2352096512666190118161122.

[4] Germany, F. S. *FSG Competition Handbook 2018* [online]. Pravidlá súťaže. 2018 [cit. 2021-12-28]. Dostupné z: https://www.formulastudent.de/fileadmin/user_upload/all/2018/rules/FSG2018_Competition_Handbook_V1.0.pdf.

[5] Germany, F. S. *Formula Student Rules 2022* [online]. Pravidlá súťaže. 2021 [cit. 2021-12-28]. Dostupné z: https://www.formulastudent.de/fileadmin/user_upload/all/2022/rules/FS-Rules_2022_v1.0.pdf.

[6] Kabzan, J., Iglesia Valls, M. de la, Reijgwart, V., Hendrikx, H. F. C., Ehmke, C. et al. AMZ Driverless: The Full Autonomous Racing System. *CoRR*. 2019, abs/1905.05150. Dostupné z: http://arxiv.org/abs/1905.05150.

[7] Koubaa, A. *Robot Operating System (ROS): The Complete Reference (Volume 2)*. Cham: Springer International Publishing AG, 2017. ISBN 9783319549262.

[8] Mattiske, M. *Simultaneous Localisation and Mapping using LiDAR for Autonomous Racing*. [cit. 2022-05-08]. Bakalárska práca. Dostupné z: https://www.monashmotorsport.com/blog/slamfyt.

[9] Poel, B. V. D. *Raceline Optimization BA Thesis (Afstudeerscriptie)* [online]. [cit. 2022-01-01]. Bakalárska práca. Dostupné z: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.224.1573&rep=rep1&type=pdf.

[10] Racing, T. B. *Formula Student* [online]. 2021 [cit. 2021-12-28]. Dostupné z: https://tubrnoracing.cz/o-nas/formula-student/.

[11] Robotics, O. *Understanding ROS 2 topics* [online]. 2022 [cit. 2022-05-08]. Dostupné z: http://docs.ros.org/en/galactic/Tutorials/Topics/Understanding-ROS2-Topics.html.

[12] Slomoi, A. Lap Time Simulation. [online]. 2000, [cit. 2022-01-01]. Dostupné z: http://www.jameshakewill.com/Lap_Time_Simulation.pdf.

[13] Slomoi, A. Path Planning and Control in an Autonomous Formula Student Vehicle. 2021, [cit. 2021-12-31]. Dostupné z: https://www.monashmotorsport.com/blog/rth91nkpjy508gvadw0n2suukwkkqi.

[14] Srinivasan, S., Sa, I., Zyner, A., Reijgwart, V., Iglesia Valls, M. de la et al. End-to-End Velocity Estimation For Autonomous Racing. *CoRR*. 2020, abs/2003.06917. Dostupné z: https://arxiv.org/abs/2003.06917.

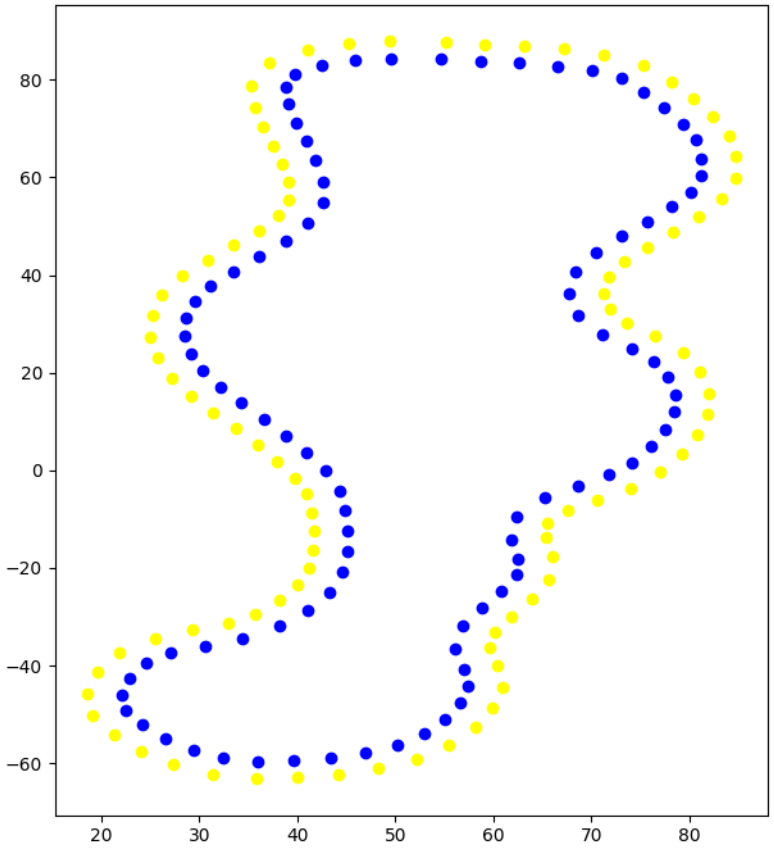
[15] Tian, H., Ni, J. a Hu, J. Autonomous Driving System Design for Formula Student Driverless Racecar. *CoRR*. 2018, abs/1809.07636. Dostupné z: http://arxiv.org/abs/1809.07636.

[16] Zhao, J.-S., Liu, Z.-J. a Dai, J. Design of an Ackermann Type Steering Mechanism. *Journal of Mechanical Engineering Science*. November 2013, zv. 227. DOI: 10.1177/0954406213475980.

Príloha A

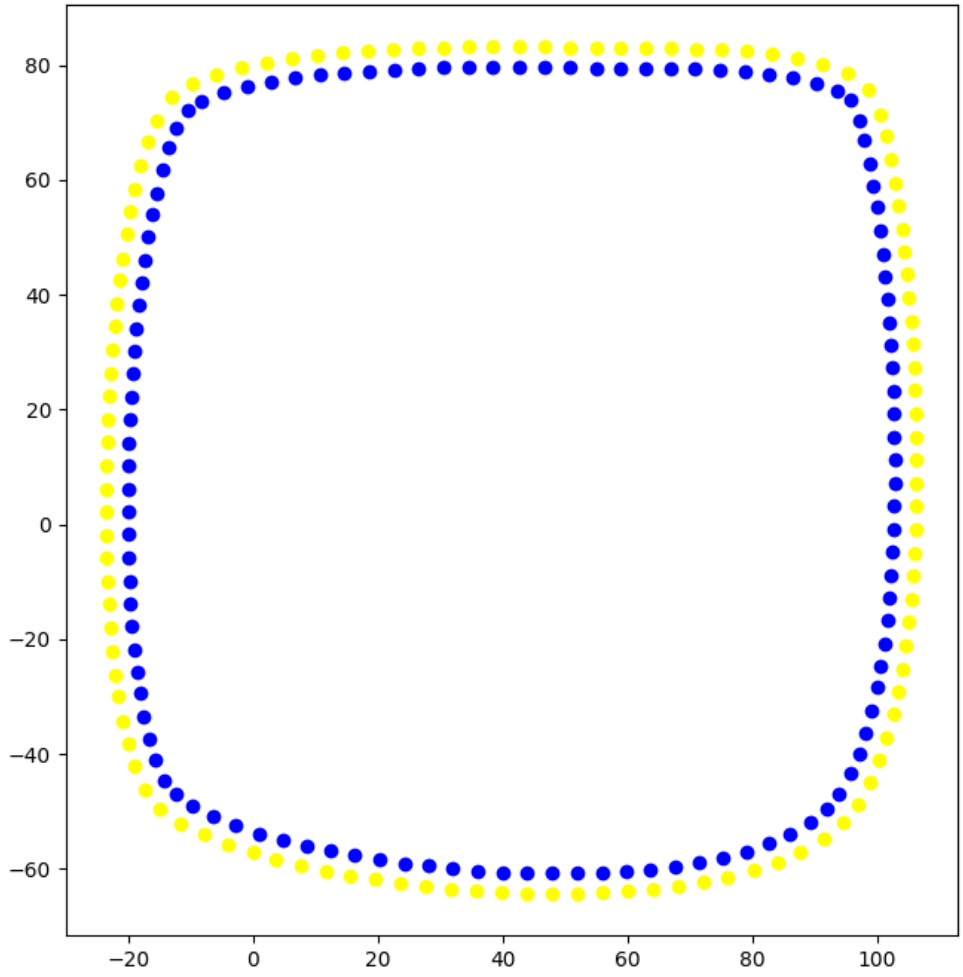
Testovacie trate v simulátore

A.1 Trať 1



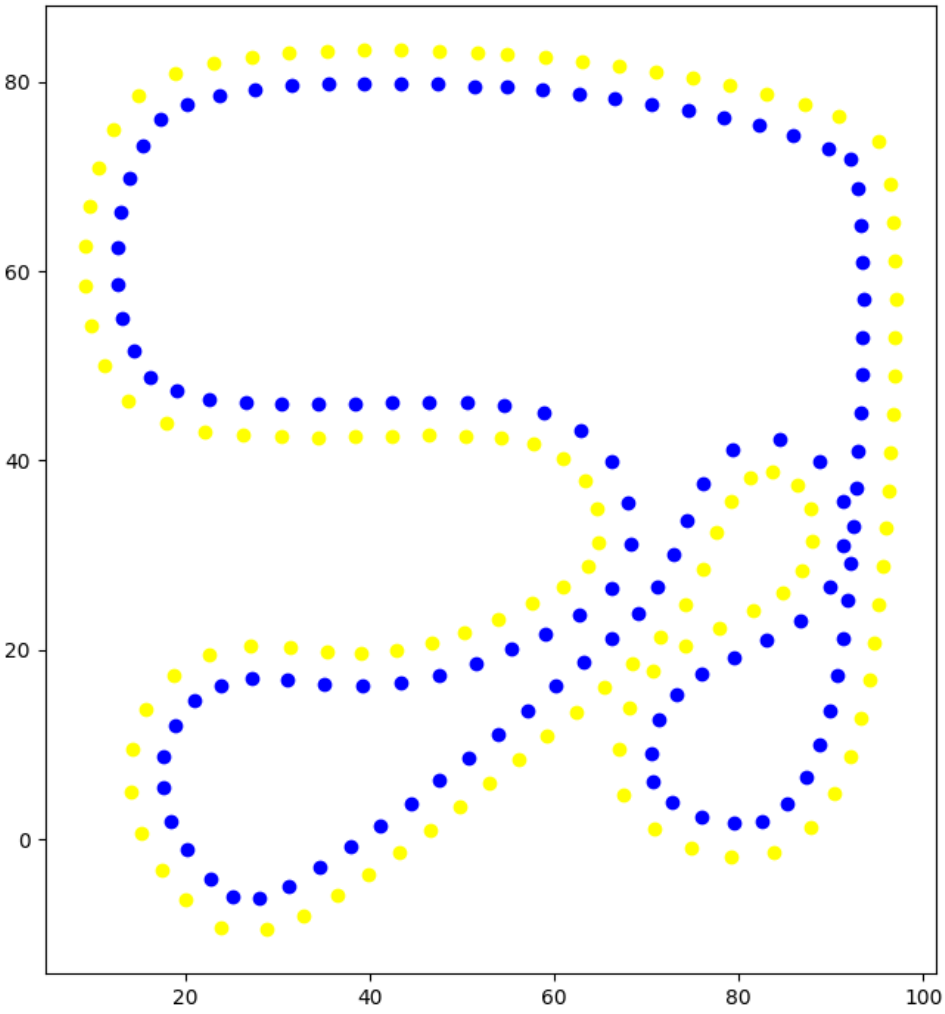
Obr. A.1: Testovacia trať 1

A.2 Trať 2



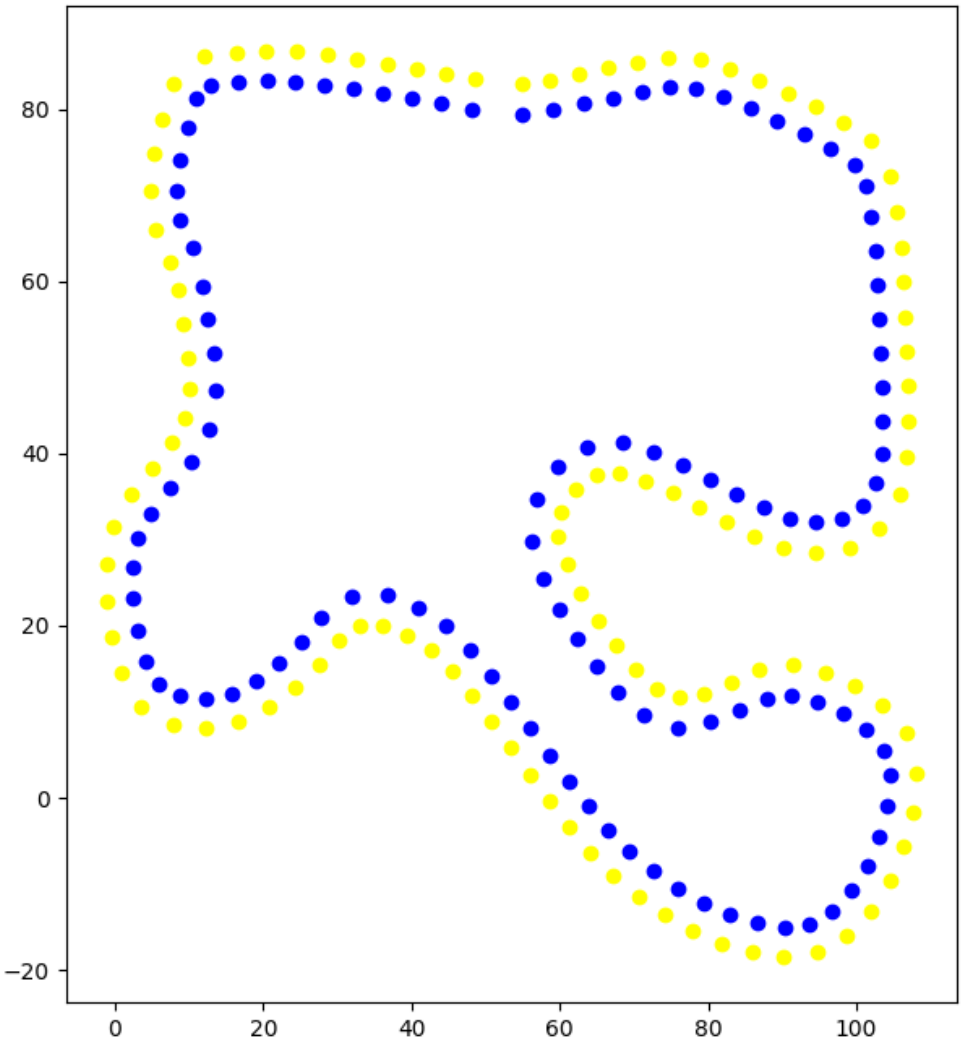
Obr. A.2: Testovacia trať 2

A.3 Trať 3



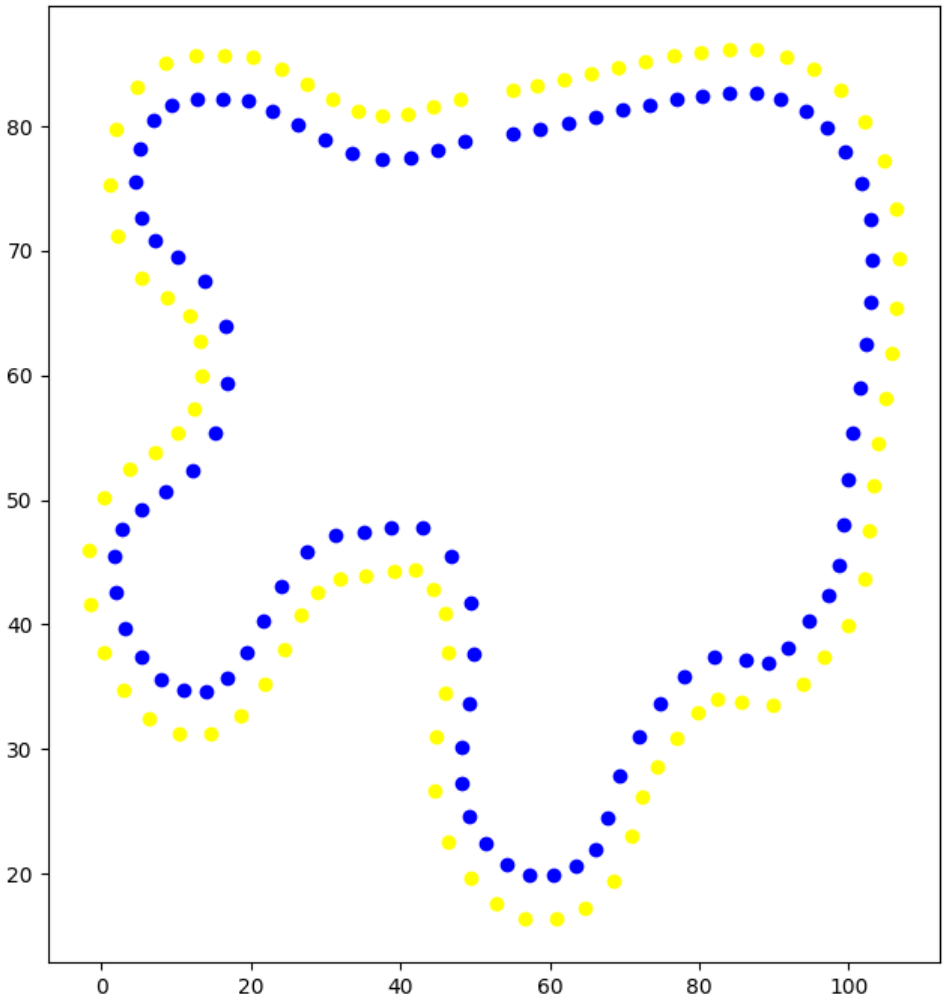
Obr. A.3: Testovacia trať 3

A.4 Trať 4



Obr. A.4: Testovacia trať 4

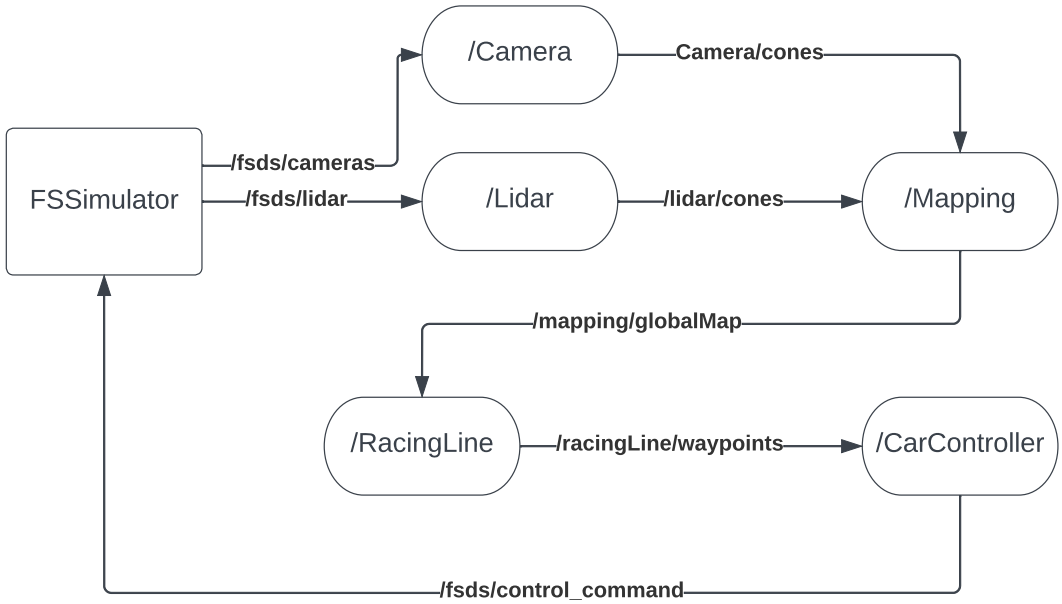
A.5 Trať 5



Obr. A.5: Testovacia trať 5

Príloha B

Návrh riadiaceho systému v ROS



Obr. B.1: Graf modulov(ROS Nodes), a komunikácia medzi nimi (ROS Topics)

Príloha C

Obsah SD karty

Priložená SD karta obsahuje:

- zdrojové kódy programu riadenia vozidla
- text bakalárskej práce vo formáte PDF
- zdrojové súbory textu bakalárskej práce
- súbor README.txt obsahujúci postup pre spustenie simulácie
- video z jazdy vozidla v simulátore