Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

BAKALÁŘSKÁ PRÁCE

Plzeň, 2021

Filip Duda

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Filip DUDA**
Osobní číslo:	**A18B0502P**
Studijní program:	**B3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Téma práce:	**Návrh řízení RC modelu auta**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1. Seznamte se s problematikou modelování pohybu RC modelu auta.
2. Navrhněte řídicí algoritmus mobilního robotu.
3. Experimentálně ověřte kvalitu navrženého řídicího algoritmu.

Rozsah bakalářské práce: **30-40 stránek A4**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

Klančar G., Zdešar A., Blažič S. and I. Škrjanc, „Wheeled Mobile Robotics: From Fundamentals Towards Autonomous Systems", ISBN 978-0-12-804204-5, Butterworth-Heinemann, 2017
Spyros G. Tzafestas. „Introduction to Mobile Robot Control", ISBN 978-0-12-417049-0, Elsevier, 2014

Vedoucí bakalářské práce: **Ing. Miroslav Flídr, Ph.D.**
Katedra kybernetiky

Datum zadání bakalářské práce: **15. října 2020**
Termín odevzdání bakalářské práce: **24. května 2021**

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
L.S.

Doc. Dr. Ing. Vlasta Radová
děkanka

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 15. října 2020

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne .....

vlastnoruční podpis

Poděkování

Rád bych poděkoval Ing. Miroslavu Flídrovi Ph.D. za přípravu systému Vicon pro měření, RC modelu a pomoc při vedení samotné práce i předchozích projektů.

Abstrakt

Tato práce se zabývá návrhem řízení RC modelu auta. V této práci je shrnuta problematika modelování pohybu kolového vozidla a různé metody jeho lokalizace. Dále je součástí práce navržení algoritmů umožňujících řízení mobilního kolového robota za účelem sledování určité trasy a rychlosti, algoritmů umožňujících automatické plánování této trasy tak, aby zohledňovaly různá kritéria, algoritmů navrhujících vhodnou rychlost průjezdu těchto tras. Tato práce dále obsahuje navržení softwarového rozhraní, které umožňuje efektivní použití zmíněných algoritmů a ověření funkčnosti a kvality těchto algoritmů.

Klíčová slova: RC model auta, Vicon, řídící systémy

Abstract

This thesis deals with control design for an RC car model. This thesis summarizes the problematics of modelling the motion of the wheeled vehicle and different methods of its localisation. Another part of the work is the design of mobile robot control algorithms to follow certain path and velocity, path planning algorithms that take into account various criteria and velocity planning algorithms. This work also contains the design of a software interface that allows effective use and verification of the functionality and quality of these algorithms.

Keywords: RC car model, Vicon, control systems

Obsah

Zkratky

DARPA Defense Advanced Research Projects Agency

FPS Frames Per Second

GLONASS Globalnaja navigacionnaja sputnikovaja sistěma

GPS Global Positioning System

IMU Inertial Measurement Unit

LIDAR Light Detection And Ranging

ORCA Optimal RC Autonomous Racing

RADAR Radio Detection and Ranging

RC Remote Control

RNDF Route Network Definition File

ROS Robot Operating System

USB Universal Serial Bus

Kapitola 1

Úvod

Autonomní řízení vozidel je v posledních letech často skloňované téma. Řada společností se předhání ve vývoji v tomto odvětví a přestože plná autonomie zatím k dispozici není, nějakou formu asistence řízení již nabízí většina automobilek. Ačkoliv k zásadnímu pokroku došlo v posledních dvou desetiletích, první myšlenky na vytvoření automaticky řízeného vozidla jsou mnohem starší.

Již na přelomu 15. a 16. století přišel Leonardo da Vinci s návrhem samořídícího mechanizmu. K výraznějšímu posunu v této oblasti ale došlo až po vynálezu a rozšíření automobilů. V roce 1939 společnost General Motors představila na veletrhu Futurama svoji vizi světa za 20 let. Součástí této vize byl systém automatických dálnic, které by naváděly samořídící automobily [8]. Přes to, že se jim tento koncept podařilo zprovoznit, potrvá nejspíše ještě několik let, než komerčně dostupné automobily budou disponovat plnou autonomií.

Mimo osobní dopravu však existují mobilní roboti pohybující se po kolech, nohách, vzduchem nebo vodou bez zásahu člověka. Tito roboti zastávají řadu funkcí, například monitorování různých oblastí, logistické práce ve skladech, nebo i autonomní vysávání a sekání trávy v domácnostech. Algoritmy používané k řízení těchto robotů se liší podle funkce daného robota, ale některé prvky zůstávají podobné. Mezi ně může patřit například hledání nejkratší trasy prostředím, nejrychlejší trasy, trasy s určitou minimální vzdáleností od překážek, algoritmy zajišťující pohyb po těchto trasách, nebo různé způsoby mapování prostředí.

Při navrhování nejen těchto algoritmů se často využívá různých druhů modelů. Tyto modely mohou být virtuální nebo fyzické a jejich cílem je zjednodušit testování. Díky těmto modelům je možné testovat levněji, rychleji, bez nutnosti přístupu k cílovému systému a je možné testovat i situace, které by na cílovém systému nebylo možné vyzkoušet například z bezpečnostních důvodů.

Matematický model nám umožňuje rychlé a levné simulování libovolného stavu, v této práci půjde především o Ackermannův kinematický model. Problémem matematických modelů může být jejich sestavení, které může být velmi komplikované a často se tak používá nějakým způsobem zjednodušený model. Řešením může být fyzický model, který může disponovat podobnými vlastnostmi jako cílový model aniž by bylo nutné tyto vlastnosti matematicky popsat.

První RC model byl patentován Nikolou Teslou v roce 1898. Jednalo se o kovový model lodi se třemi anténami [18]. Od té doby byla elektronika potřebná k ovládání RC modelů výrazně zmenšena a RC ovládání se rozšířilo do mnoha různých odvětví. Tato práce se zabývá návrhem řízení RC modelu auta. Řadu algoritmů používaných při řízení automobilů je možné otestovat v menším měřítku na RC modelu a především algoritmy pro plánování trasy jsou nezávislé na měřítku.

1.1 Cíle

Cílem této práce je navrhnout řízení RC modelu auta. Součástí této práce by mělo být prostředí ve kterém bude možné zadat počáteční a cílový bod trasy a algoritmus, který dokáže navrhnout trasu

mezi těmito body tak, aby bylo možné následně trasu s pomocí zpětnovazebního řídícího algoritmu modelem projet a nedošlo ke kolizi se známými překážkami.

- Popsat a srovnat různé metody lokalizace
- Zvolit vhodný způsob modelování kinematiky RC modelu auta
- Navrhnout řídící algoritmus RC modelu za účelem sledování referenční trasy a rychlosti
- Navrhnout algoritmus pro automatické plánování trasy zohledňující pohybová omezení RC modelu a vnější překážky
- Experimentálně ověřit kvalitu řídících i plánovacích algoritmů

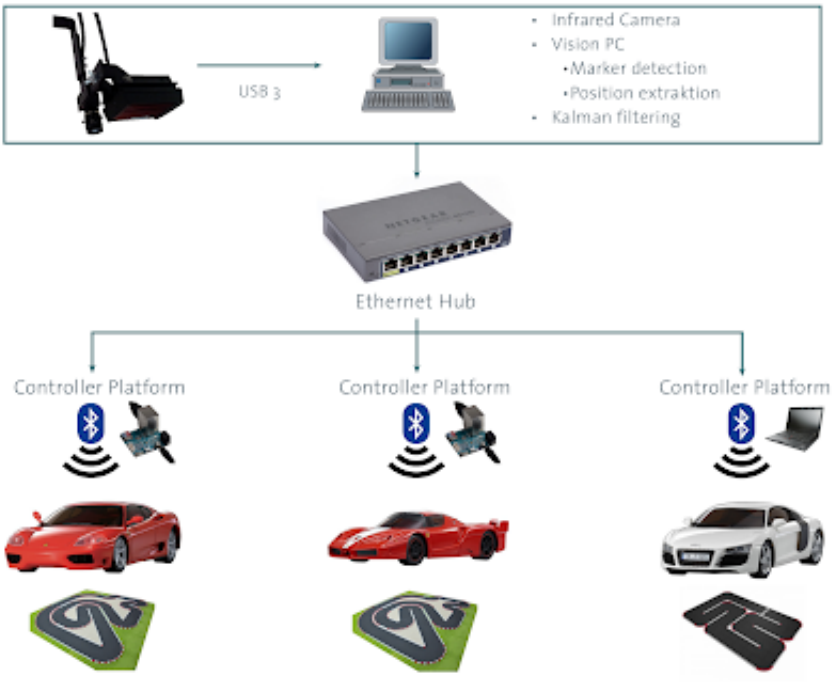
1.2 Stávající stav problematiky

Autonomním řízením RC modelů i reálných automobilů se již zabývalo mnoho prací, které využívaly různé metody lokalizace, plánování i řízení. V této sekci jsou popsány tři z těchto prací.

ORCA Racer, neboli Optimal RC Autonomous Racing je projekt Spolkové vysoké školy v Curychu. Cílem téměř třicetičlenného týmu bylo vyvinout hardware a software, který umožní simultánní autonomní řízení až 10 RC modelů aut v reálném čase.

Základním prvkem je ORCA Vision System, vizuální poziční systém, umožňující 2D sledování objektů pohybujících se v rovině. Tento systém funguje díky kameře PointGrey Flea USB 3 s rozlišením 1280x1024 a snímkovací frekvencí až 150 FPS. Tato kamera byla vybavena širokoúhlým objektivem a infračerveným filtrem. Vedle kamery byl umístěný infračervený zářič. Na RC modely byly umístěny markery odrážející infračervené záření. Z obrazových dat je poté vypočtena pozice modelu.

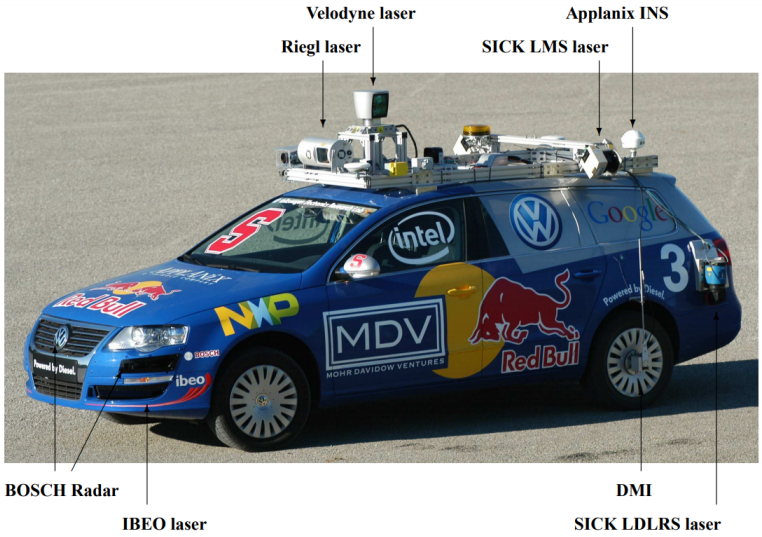
Data o poloze jsou následně posílána přes Ethernet řídícím zařízením. Ty poté pomocí Bluetooth odešlou vhodné řídící instrukce řídící elektronice modelu [13].



Obrázek 1.1: Přehled systému ORCA Racer [13]

Autonomním řízením RC modelů se ve své práci Design and implementation of a time-optimal controller for model race cars zabýval i Robin Verschueren. Podobně jako Orca Racer využíval pro lokalizaci infračervenou kameru. Jeho cílem byla časová optimalizace při průjezdu dráhou. V práci jsou popsány rozdíly mezi průjezdem nejkratší trasy a trasy s nejmenším zakřivením. Na závěr je implementován algoritmus, který hledá časově optimální trasu pro nadcházející úsek trati v reálném čase [20].

Junior je robotické vozidlo schopné autonomního průjezdu městským prostředím. Bylo vytvořeno týmem ze Stanfordovy univerzity a získalo druhé místo na soutěži DARPA Urban Challenge v roce 2007. Vzniklo modifikací vozu Volkswagen Passat, ten byl vybaven elektronickým ovládáním plynu, brzdy, volantu, řazení, ruční brzdy a směrových světel. K zjištění polohy a mapování bylo vozidlo dále vybaveno několika GPS přijímači, odometrickou jednotkou, laserovými senzory vzdálenosti, optickým infračerveným senzorem pro detekci horizontálního značení, na střeše vozu byl připevněn LIDAR a RADAR. V kufru automobilu byla umístěna výpočetní technika [16].



Obrázek 1.2: Junior [16]

Metody řízení, mapování a lokalizace reálného vozu jsou oproti řízení RC modelu poměrně odlišné. Relativně podobné bez závislosti na měřítku jsou metody plánování trasy. Projekt Junior využívá dva základní plánovací algoritmy. Algoritmus pro hledání vhodné globální trasy již namapovanými silnicemi ve formátu RNDF a algoritmus pro plánování předjíždění, projíždění křižovatkou, parkování nebo objíždění překážek, který využívá data ze senzorů. Jedná se o modifikovanou verzi algoritmu A*. Tento reprezentuje stavy vozidla v 4D diskrétní mřížce. Těmito složkami je poloha středu vozidla v dvourozměrném souřadném systému, směr natočení vozidla a směr pohybu. Při hledání trasy jsou počítány dvě heuristiky, jedna neholonomní bez překážek a holonomní s překážkami. Použito je maximum z těchto dvou. Výsledná trasa je na závěr vyhlazena, což umožňuje její rychlejší projetí [16].

Kapitola 2

Systémy určování polohy a rychlosti

K fungování zpětnovazebního řízení je třeba znát aktuální informace o poloze a rychlosti modelu. K tomu je možné použít různé druhy senzorů. V této kapitole jsou porovnány některé z nich.

2.1 Enkodéry

Enkodéry jsou zařízení, která převádí rotační pohyb na signál. Nejčastěji se používají magnetické a optické enkodéry. V optických enkodérech se nachází děrovaný disk, skrz který svítí světlo na optický snímač, při pootočení dojde k přerušení světla, což se na optickém snímači projeví sestupnou hranou výstupního signálu. Enkodéry jsou schopné určit poměrně přesně natočení i rychlost kol mobilního robota a často se k tomuto účelu používají. Nevýhodou je potřeba připojení na hřídel, se kterou je třeba počítat již při mechanickém návrhu robota. Enkodér může získat přesnou informaci o poloze robota v prostoru pouze za předpokladu dokonalé trakce. Při vyšších rychlostech nebo povrchu s nižším třením kvůli tomu může chyba ve výpočtu pozice narůstat tak rychle, že tento senzor nebude použitelný.

2.2 IMU

IMU jednotka je zařízení sloužící k měření akcelerace, úhlové rychlosti a orientace v prostoru. K fungování využívá kombinace dat z akcelerometru, gyroskopu a v některých případech i magnetometru.

Gyroskop slouží k měření úhlové rychlosti. V poslední době se velmi často používají vibrační gyroskopy, které určují úhlovou rychlost měřením Coriolsovy síly vibrujícího objektu [9].

Akcelerometr je senzor sloužící k měření zrychlení. Funguje na principu závaží na pružině. Při zrychlení působí na závaží síla popsaná Newtonovou rovnicí $F = ma$, kde m je hmotnost závaží a a je zrychlení tělesa. Tato síla způsobuje stlačení pružiny. Toto stlačení je možné popsat Hookovým zákonem $F = kx$, kde k je tuhost pružiny a x je výchylka závaží. Spojením s Newtonovým zákonem můžeme získat vztah $a = kx/m$ [11].

Spojením několika akcelerometrů můžeme měřit zrychlení ve více směrech. Ze znalosti časového průběhu zrychlení je možné dopočítat časový průběh rychlosti a dráhy. Takové trasování může být poměrně přesné po krátkou dobu, ale chyba měření se s časem zvyšuje.

V roce 2013 provedl tým na Nanyang Technological University pokus s trasováním osoby pomocí tří IMU senzorů. Při pohybu s maximální rychlostí 1.5m/s dosahovala chyba 1-3% délky měřené trajektorie [22]. Výhodou těchto senzorů je jejich velmi malá velikost, nízká cena, vysoká snímkovací frekvence.

2.3 Satelitní poziční systém

Satelitní poziční systémy fungují díky satelitním vysílačům, které poskytují informaci o poloze, rychlosti a čase neomezenému množství přijímačů. Každý přijímač musí mít pro určení pozice signál minimálně ze 4 vysílačů. Tyto vysílače posílají periodicky informaci o svojí poloze a časovou značku. Přijímací zařízení z časového rozdílu mezi odesláním a přijetím získá čas, který signálu trvala trasa od vysílače k přijímači, vynásobením s rychlostí světla získá vzdálenost od vysílače. Trilaterací pak získá vlastní pozici v prostoru [10].

V současné době fungují krom nejpoužívanějšího systému GPS také systémy Galileo, GLONASS a BeiDou. Přesnost těchto systémů je různá a liší se i v závislosti na pozici přijímače. Řádově se na veřejně přístupném vysílání jedná o jednotky metrů [21].

Použití tak má jen u robotů, u kterých tato relativně nízká přesnost nebrání praktickému použití. Lokalizace pomocí GPS je běžně používána například u dronů, kde není centimetrová přesnost vyžadována, navíc je možné upřesnit informaci o výšce pomocí barometru.

2.4 Motion capture systém

Motion capture je proces zaznamenávání pohybu fyzického objektu do souřadnic. Podle principu zaznamenávání můžeme motion capture systémy rozdělit na mechanické, elektromagnetické a optické.

U mechanických je na sledovanou osobu umístěn exoskelet, který pomocí senzorů sleduje natočení kloubů a dopočítává pozici osoby. Nevýhodou této technologie je, že není schopná získat absolutní pozici nebo natočení.

Elektromagnetický systém funguje díky magnetickým senzorům, umístěným na sledované osobě nebo objektu. Tyto senzory měří vzdálenost od statických magnetických vysílačů. Výhodou je jednodušší zpracování signálu a možnost získávání absolutní pozice.

Při použití optických systémů jsou na osobu nebo objekt umístěny reflexní body. Tyto body jsou sledovány kamerami z několika úhlů a z obrazových dat je dopočtena absolutní pozice. Výhodou je volnost pohybu, poměrně přesné zaměření a možnost sledovat více objektů nebo lidí. Nevýhodou může být citlivost na světelné rušení a oproti jiným metodám určování polohy i poměrně vysoká výpočetní náročnost [6]. Ve svém článku z roku 1995 Dyer, Martin a Zulauf uvádějí, že k zpracování jedné sekundy obrazových dat je potřeba 1-30 minut výpočtu [5]. Z toho je zřejmé, že použití optických motion capture systémů při zpětnovazebním řízení je možné až v posledních letech.

Při řízení uvnitř budovy bez světelného rušení na prostoru několika metrů a zajištění dostatečného výpočetního výkonu umožňuje optický motion capture systém velmi přesnou lokalizaci robota.

Kapitola 3

RC Model

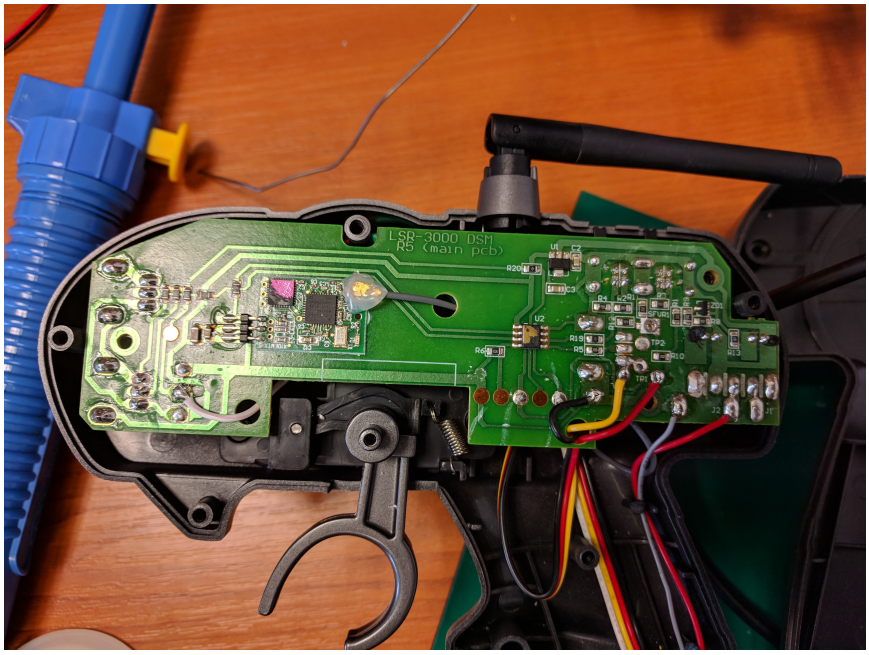
Řízeným systémem je RC model auta Micro-Rally Car od společnosti Losi v měřítku 1:24. Tento RC model je vybaven pohonem všech čtyř kol, diferenciálem a olejovými tlumiči. Výrobce dále uvádí následující parametry:

Délka[mm]	Šířka[mm]:	Rozvor[mm]	Hmotnost[g]
163	94.5	99	153

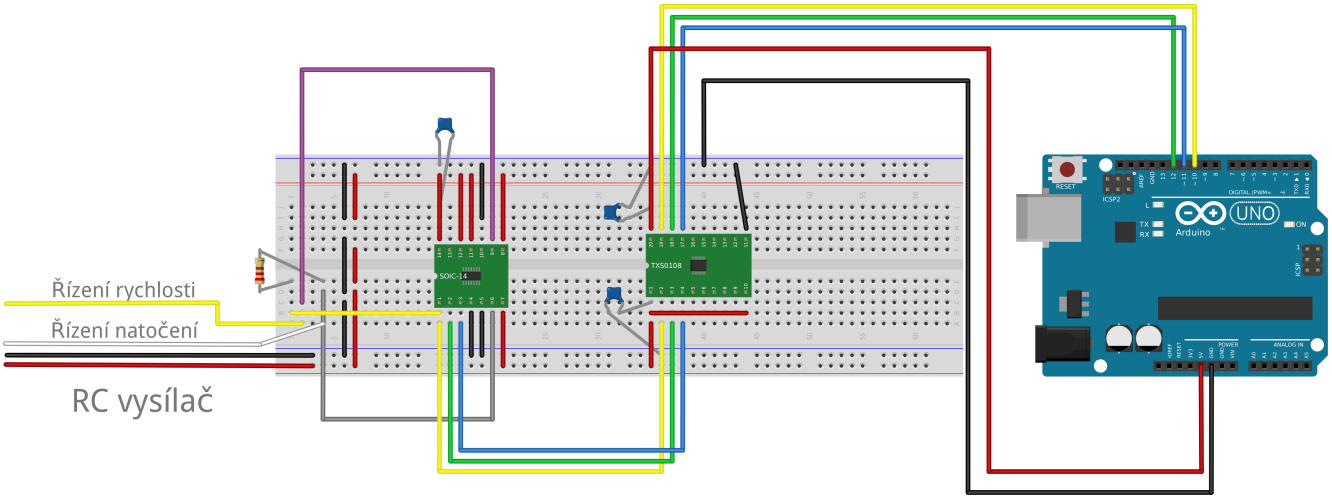
Tabulka 3.1: Parametry udávané výrobcem [14]

Obrázek 3.1: RC model s Vicon markery

Model auta je ovládán dálkovým ovladačem Losi 27 MHz AM, který byl modifikován, aby místo původních mechanických vstupů přijímal signál z mikrokontroleru Arduino Uno. Původní ovládání bylo zajištěné dvěma mechanickými potenciometry pro ovládání rychlosti a natočení kol. Ty byly nahrazeny digitálními potenciometry MCP42010, které pomocí sériového rozhraní komunikují s mikrokontrolerem Arduino Uno, které získává informace o požadovaném natočení ze systému ROS. Dále byly na model připevněny markery pro sledování systémem Vicon.



Obrázek 3.2: Zapojení vysílače



Obrázek 3.3: Zapojení digitálního potenciometru

Kapitola 4

Automatické řízení

U RC modelu je možné řídit natočení předních kol a moment otáčení motoru. Tato kapitola se zabývá jejich nastavením tak, aby RC model procházel požadovanou trasou.

4.1 Trasa

Cílem řídícího algoritmu je projet modelem co nejpřesněji požadovanou trasu požadovanou rychlostí. Tato trasa je určena seznamem trojic celých čísel, které značí souřadnice x a y a požadovanou rychlost, kterou se má model pohybovat od předchozího bodu k aktuálnímu.

4.2 Dopředná kinematika Ackermannova řízení

RC model stejně jako většina automobilů používá Ackermanův princip řízení. Při zatáčení je pak vnitřní kolo natočené více než vnější, protože opisuje křivku s menším poloměrem. V důsledku toho se vnitřní kolo pohybuje pomaleji. Díky diferenciálu se mohou kola otáčet rozdílnou rychlostí a nedochází tedy k prokluzování jednoho z kol. Kombinace Ackermannova řízení a diferenciálu výrazně zvyšuje řiditelnost vozidla a snižuje opotřebení pneumatik.

Úhel natočení kol můžeme spočítat jako

$$\alpha_L = \frac{\pi}{2} - \arctan\frac{R + \frac{l}{2}}{d} \tag{4.1}$$

$$\alpha_R = \frac{\pi}{2} - \arctan\frac{R - \frac{l}{2}}{d} \tag{4.2}$$

α_L a α_R je natočení levého a pravého kola, R je poloměr otáčení RC modelu a d je rozchod kol (viz obrázek 4.1). Pohybový model Ackermannova řízení můžeme zjednodušit tím, že přední kola nahradíme virtuáním prostředním kolem s úhlem natočení

$$\alpha = \frac{\pi}{2} - \arctan\frac{R}{d} \tag{4.3}$$

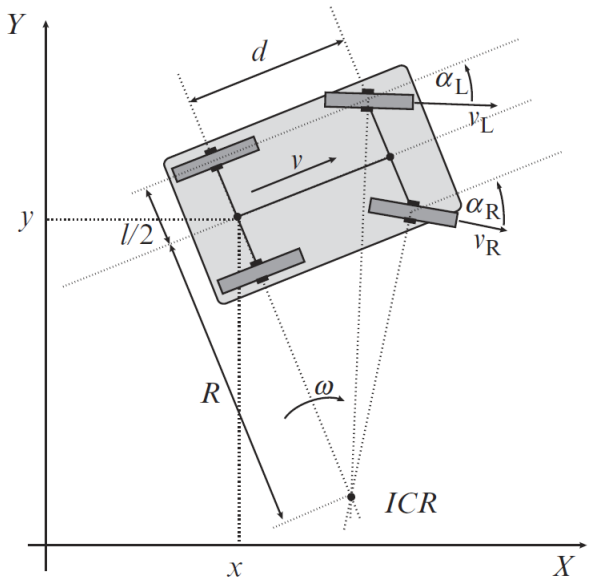
Pohyb modelu poté můžeme popsat rovnicemi

$$\dot{x} = v \cdot \cos(\varphi) \tag{4.4}$$

$$\dot{y} = v \cdot \sin(\varphi) \tag{4.5}$$

$$\dot{\varphi} = v \cdot \frac{1}{d} \cdot \tan(\alpha) \tag{4.6}$$

kde x a y jsou souřadnice v prostoru, φ je natočení v prostoru a α je natočení kol [7].



Obrázek 4.1: Schéma Ackermannova řízení [5]

4.3 Diskretizace Ackermannova modelu

Z kinematiky Ackermannova řízení známe rovnice popisující pohyb RC modelu (4.4)-(4.6). Jejich diskretizací můžeme získat rovnice popisující stav k+1, kde T je perioda vzorkování a $t = kT$. K diskretizaci byla použita Eulerova metoda.

$$x(k+1) = x(k) + v(k)\cos(\varphi(k))T \tag{4.7}$$

$$y(k+1) = y(k) + v(k)\sin(\varphi(k))T \tag{4.8}$$

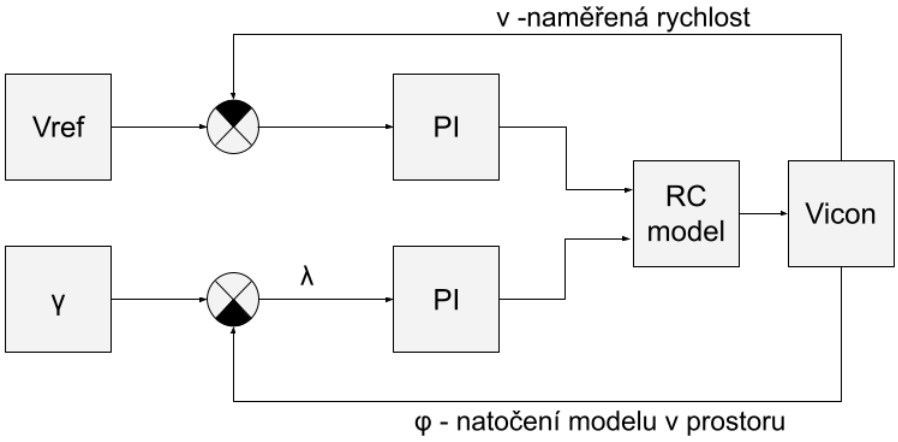
$$\varphi(k+1) = \varphi(k) + v(k)\frac{1}{d}\tan(a(k))T \tag{4.9}$$

4.4 Řízení rychlosti

Cílem je udržovat rychlost RC modelu na úrovni referenční hodnoty. Rychlost RC modelu je závislá na rychlosti otáčení kol a aktuální trakci. Z těchto dvou lze efektivně řídit pouze rychlost otáčení kol a to změnou napětí na motoru RC modelu. To mění řídící elektronika RC modelu podle hodnoty řídící proměnné, kterou získala z ovladače. Vzhledem k vysokému zašumění průběhu zrychlení jsem se rozhodl derivační složku při řízení nepoužít a požadované rychlosti dosáhnout PI regulátorem. Velký vliv na zrychlení má stav baterie, je tedy třeba nastavit poměrně silnou integrační složku, aby vybití baterie kompenzovala.

4.5 Zpětnovazební řízení natočení kol za účelem sledování zadané trasy

Směr pohybu je řízený natočením předních kol. Cílem regulátoru je minimalizovat rozdíl mezi skutečným natočením RC modelu (úhel φ na obrázku 4.3) a natočením požadovaným (úhel λ na ob-



Obrázek 4.2: Schéma PI regulátorů

rázku 4.3). Úhel λ získáme jako rozdíl

$$\lambda = \gamma - \varphi \tag{4.10}$$

kde φ je úhel natočení RC modelu v soustavě souřadnic, tento úhel získáme ze systému Vicon. Úhel γ je úhel úsečky AB spojující cílový bod A se středem RC modelu B vůči ose Y v soustavě souřadnic. Tento úhel získáme ze souřadnic cílového bodu (x1 a y1) a souřadnic RC modelu (x2 a y2) jako

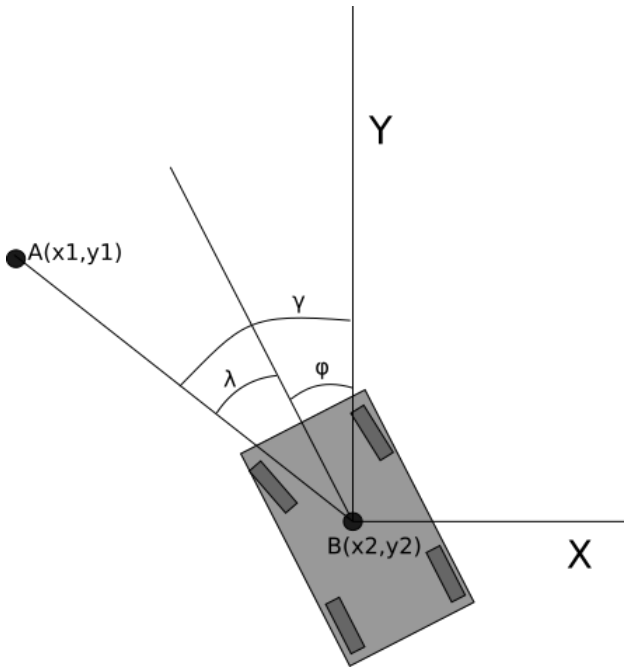
$$\gamma = \texttt{atan2}(x1 - x2, y1 - y2) \tag{4.11}$$

atan2 je funkce, která rozšiřuje funkci arkus tangens o druhý parametr a umožňuje rozlišení jednotlivých kvadrantů. Je definován následujícím předpisem [2]

$$\text{atan2}(y,x) = \begin{cases} \text{atan}\left(\frac{y}{x}\right) & x > 0 \\ \pi + \text{atan}\left(\frac{y}{x}\right) & y \geq 0, x < 0 \\ -\pi + \text{atan}\left(\frac{y}{x}\right) & y < 0, x < 0 \\ \frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ 0 & y = 0, x = 0 \end{cases}$$

Úhel λ by měl být konvexní. V případě, že $\gamma - \varphi > \pi$ rad, pak použijeme vzorec

$$\lambda = -(2 \cdot \pi - (\gamma - \varphi)) \tag{4.12}$$



Obrázek 4.3: Znázornění úhlů

Kapitola 5

Grafické uživatelské rozhraní

Pro jednodušší ladění PI regulátoru, zaznamenávání překážek, návrh tras a jejich vizualizaci jsem vytvořil s pomocí knihovny Tkinter grafické rozhraní. Tato knihovna umožňuje jednoduchou tvorbu grafických prvků v jazyce Python. Výhodou této knihovny je podpora velkého množství operačních systémů.

5.0.1 Canvas

Hlavní částí rozhraní je prvek canvas, který umožňuje vykreslování různých grafických prvků jako bod, čára, mnohoúhelník nebo obrázek. Prvek canvas symbolizuje prostor laboratoře při pohledu shora. Čtvercem je zde ohraničený prostor ve kterém měření funguje spolehlivě, mimo tento prostor je také možné měřit, ale vyskytují se zde slepá místa. Dále canvas obsahuje vizualizaci překážek, různých tras a samotného RC modelu. V místě RC modelu jsou periodicky generovány body, aby bylo možné sledovat historii pohybu modelu.

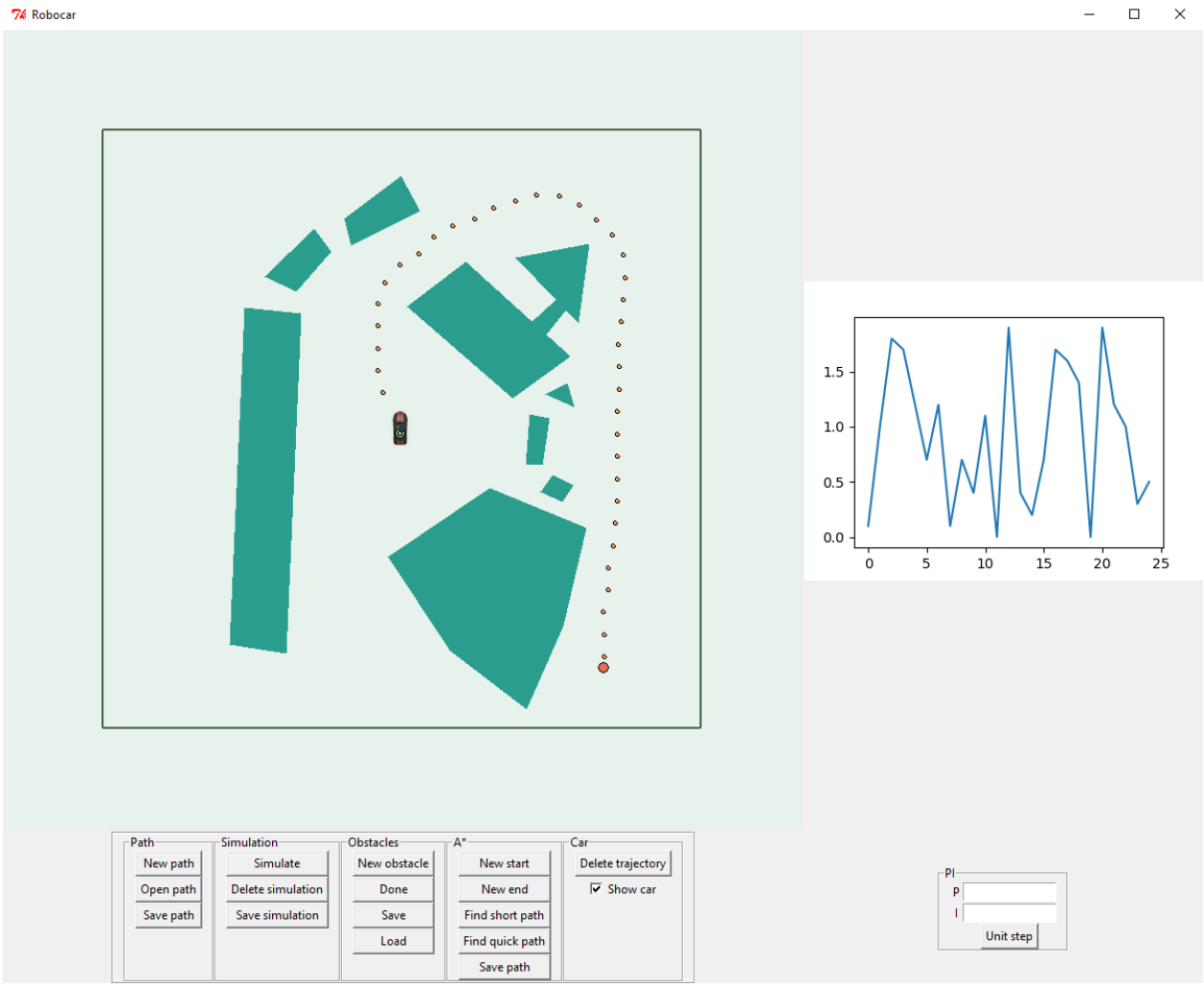
5.0.2 Ovládací panel

Ovládací panel obsahuje sadu tlačítek rozdělených do pěti sekcí, pomocí kterých je možné plánovat trasu a spouštět různé návrhové algoritmy.

První sekcí je sekce Path, ta obsahuje tři tlačítka. Je určená pro manuální návrh trasy. Tlačítko New path spouští návrhový režim, ve kterém při kliknutí na canvas vznikne bod trasy. Postupně je tak možné pomocí jednotlivých bodů navrhnout celou trasu. Tuto trasu je poté možné pomocí tlačítka Save path uložit. Tlačítko Open path otevře dialogové okno, ve kterém uživatel může vybrat některou z již uložených tras, zobrazit ji na canvas a dále s ní pracovat. Načíst tímto tlačítkem je možné nejen manuálně vytvořené trasy, ale i trasy vytvořené algoritmy v ostatních sekcích.

Druhou sekcí je sekce simulation, která umožňuje simulovat průjezd RC modelu aktuální trasou řídícím algoritmem. K tomu je použit Ackermannův kinematický model a řídící algoritmus. Simulační skript používá přímo metody řídícího skriptu a měl by tedy řízení odpovídat i při změně parametrů řízení. Simulaci je možné z canvasu smazat tlačítkem Delete simulation nebo uložit pro další zpracování tlačítkem Save simulation. Simulace se hodí především při manuálním návrhu trasy, pro ověření, zda je vůbec možné tuto trasu modelem projet.

Třetí sekce se zabývá vytvářením překážek. Tlačítko New obstacle zapíná režim vytváření překážky. Kliknutím do požadovaného místa na canvas vznikne bod překážky. Kliknutím na tlačítko Done může uživatel dokončit překážku, algoritmus spojí všechny uživatelem zadané body do konvexního n-úhelníku a vypočte bezpečnostní zónu kolem překážky. Ta se na canvas nezobrazuje, ale je s ní počítáno při návrhu trasy. Vytváření překážek nekonvexních tvarů je možné spojením několika konvexních částí. Stejně jako sekce Path, i sekce Obstacles obsahuje tlačítka Save a Load pro ukládání a načítání množiny překážek.



Obrázek 5.1: Grafické uživatelské rozhraní

Sekce A* obsahuje nástroje potřebné pro automatické hledání trasy. Pomocí tlačítek New start a New end je možné určit počátek a cíl trasy. Tlačítko Find short path poté spustí algoritmus, který mezi těmito body nalezne nejkratší trasu tak, aby bylo možné ji projet RC modelem a aby nedošlo ke kolizi s překážkou. Tlačítko Find quick path spouští podobný skript s jinak nastavenými prioritami, tento skript se snaží nalézt trasu, která bude krátká a zároveň ji bude možné projet co nejvyšší rychlostí s cílem minimalizovat čas potřebný pro cestu mezi počátkem a cílem. Navrženou trasu je možné uložit tlačítkem Save path.

Poslední sekce Car umožňuje přehlednější používání grafického rozhraní. Tlačítkem Delete trajetory je možné odstranit body, které za sebou grafická reprezentace RC modelu zanechává, například před začátkem nového měření. Sekce dále obsahuje zaškrtávací pole pro skrytí grafické reprezentace RC modelu.

5.0.3 Nástroj pro ladění parametrů PI regulátoru

Pravá část grafického rozhraní slouží především k ladění regulátoru. V horní části se nachází graf rychlosti, ten vykresluje průběh rychlosti RC modelu v posledních 25 vteřinách s vzorkovací frekvencí 10Hz.

V pravém dolním rohu se nachází nástroj pro nastavení parametrů PI regulátoru. Jsou zde dvě textová pole, do kterých je možné zadat parametry K_p a K_i. Tlačítkem Unit step je možné spustit experiment, ve kterém je referenční hodnota po dobu 3 vteřin nastavena na 1m/s. Na grafu rychlosti lze poté sledovat průběh pro dané nastavení a zhodnotit kvalitu tohoto nastavení.

Kapitola 6

Experimentální měření s pomocí systému Vicon

V této kapitole je nejprve popsán systém ROS, přes který systém Vicon komunikuje s řídícími algoritmy. Dále jsou zde záznamy trajektorie, rychlosti a zrychlení nejprve manuálního řízení a následně je ověřena funkčnost automatického řízení. Součástí této kapitoly je měření statické charakteristiky.

6.1 Robot operating system

Robot Operating System, zkráceně ROS, je open source framework pro vývoj softwaru pro robotická zařízení. Od počátku vývoje v roce 2007 se ROS stal oblíbenou platformou s velkým množstvím softwarových balíčků a desítkami podporovaných komerčních robotů. ROS obsahuje sadu ovladačů, které umožňují čtení dat ze senzorů a zasílání příkazů aktuátorům v abstrahovaném formátu. Krom sady řídících a mapovacích algoritmů obsahuje ROS také infrastrukturu pro přesun dat. Tato infrastruktura umožňuje spojování komponent robotického systému a jednoduché ukládání přenášených dat [17].

Jedním ze způsobů, jak přesouvat data v systému ROS je využití takzvaných topiců. Ty umožňují přenos jednoduchých zpráv i složitějších datových struktur mezi jednotlivými částmi systému.

Rosbag je sada nástrojů umožňujících nahrávání a opětovné posílání zpráv z ROS topic ve formátu ROS bag.

Pro propojení motion capture systému Vicon s robotic operating system je použit ovladač vicon_bridge spravovaný Markusem Achtelikem. Tento ovladač umožňuje sledovat jednotlivé markery nebo jeden či více objektů [3].

Pro komunikaci mezi grafickým rozhraním a ovládacím skriptem je použit ROS parameter server. Jedná se o slovník přístupný přes síťové aplikační rozhraní. Tento slovník je globálně přístupný, aby bylo možné snadno číst a zapisovat konfiguraci systému [15].

6.2 Záznam experimentu

Model s markery jsem umístil na plochu sledovanou systémem Vicon a manuálně ho pomocí gamepadu řídil. Data jsem v systému ROS ukládal ve formátu ROS bag[1]. Tento ROS bag obsahuje informace o čase měření, tabulku s ukládanými ROS topic[2] a samotné zprávy. Při měření mě zajímaly zprávy z Vicon o pozici RC modelu a zprávy s řídícími proměnnými pro RC model. Řídicí zprávy obsahují lineární a úhlový vektor. K řízení jsou použity dvě proměnné, X z lineárního vektoru a Z

[1] http://wiki.ros.org/rosbag
[2] http://wiki.ros.org/rostopic

Obrázek 6.1: Kamera systému Vicon

z úhlového. Zprávy z Vicon obsahují časovou značku, translační vektor s pozicí RC modelu a kvaternion s natočením RC modelu v prostoru.

6.3 Převod kvaternionu na úhly

Výstupem systému Vicon je struktura, ve které je úhel natočení uložený jako kvaternion.

Kvaternion je čtyřprvkový vektor, do kterého lze uložit libovolné natočení v 3D prostoru. Skládá se z jednoho reálného a tří komplexních prvků[1]. Aby bylo možné tyto úhly použít v řízení, je nutné je nejprve převést na Eulerovy úhly v prostoru kartézských souřadnic v radiánech. K tomu jsem využil následující vzorce [4]

$$\phi = \tan^{-1}(2\frac{q_r q_z + q_x q_y}{1 - 2(q_y^2 + q_z^2)}) \tag{6.1}$$

$$\theta = \sin^{-1}(2(q_r q_y - q_x q_z)) \tag{6.2}$$

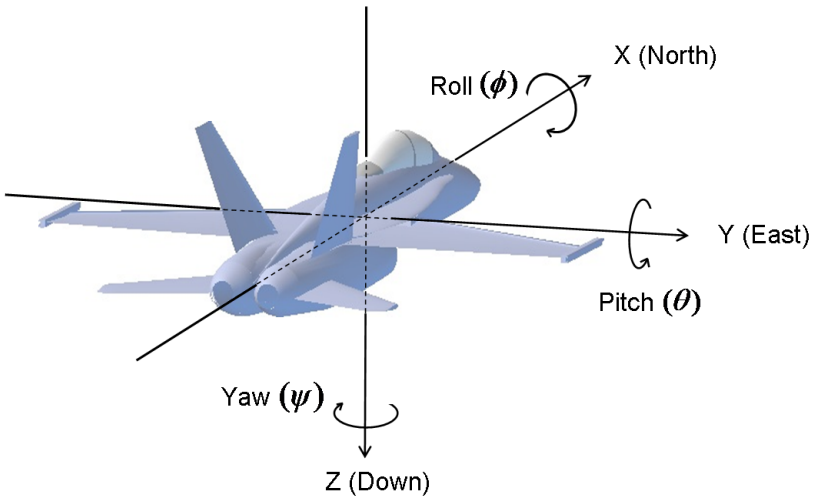
$$\psi = \tan^{-1}(2\frac{q_r q_x + q_y q_z}{1 - 2(q_x^2 + q_y^2)}) \tag{6.3}$$

kde ϕ, θ a ψ jsou Eulerovy úhly v prostoru kartézských souřadnic v radiánech (viz obrázek 6.2).

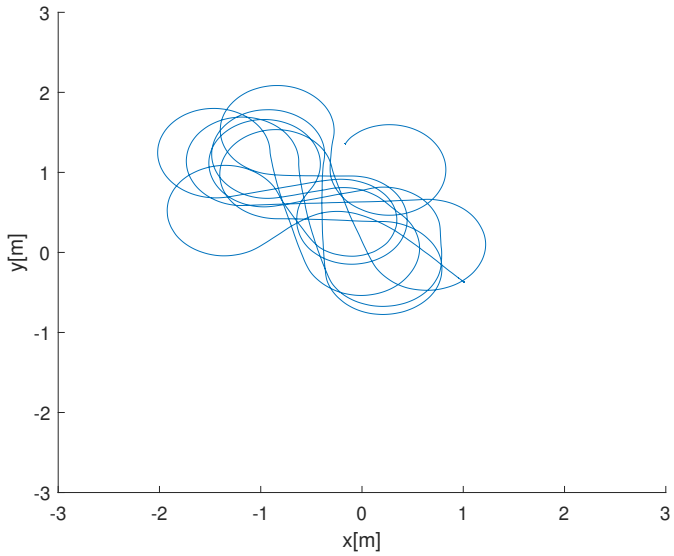
6.4 Vizualizace trajektorie

Naměřený záznam polohy modelu jsem pomocí Matlab vykreslil do grafu 6.3, čímž jsem získal vizualizaci trajektorie.

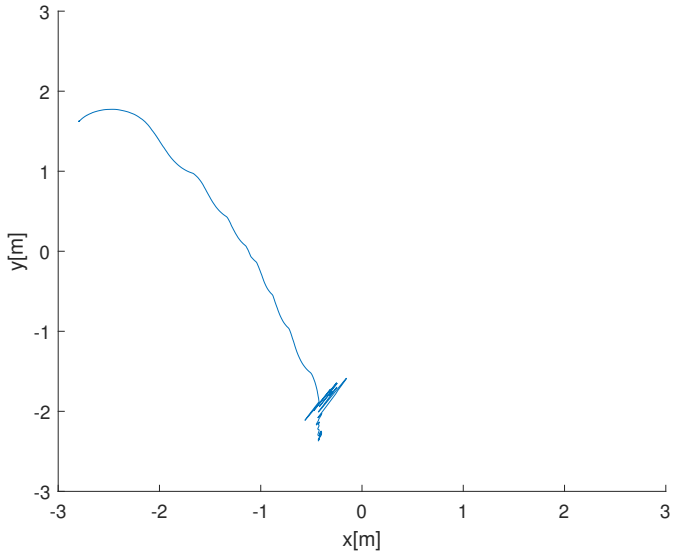
Pracovní plocha systému Vicon má tvar čtverce o straně 6m, při pohybu u krajů plochy se však RC model může dostat do slepého úhlu kamer (obrázek 6.4). Je tedy bezpečnější držet se při plánování trasy dostatečně daleko od krajů měřené plochy.



Obrázek 6.2: Souřadný systém [1]



Obrázek 6.3: Příklad vizualizace trajektorie RC modelu



Obrázek 6.4: Příklad vjezdu do slepého úhlu

6.5 Rychlost RC modelu

Každý záznam souřadnic má svojí časovou značku. Pro každý časový úsek můžeme získat rychlost jako

$$v = \frac{\Delta S}{\Delta T} \tag{6.4}$$

kde

$$\Delta S = \sqrt{\Delta x^2 + \Delta y^2} \tag{6.5}$$

kde x a y jsou souřadnice v kartézské soustavě souřadnic a T je čas.

Rychlost otáčení kol modelu je ovládána pomocí celočíselné proměnné v rozsahu od 0 do 255. Průběhu rychlosti z obrázku 6.5 odpovídá průběh řídící proměnné z obrázku 6.6.

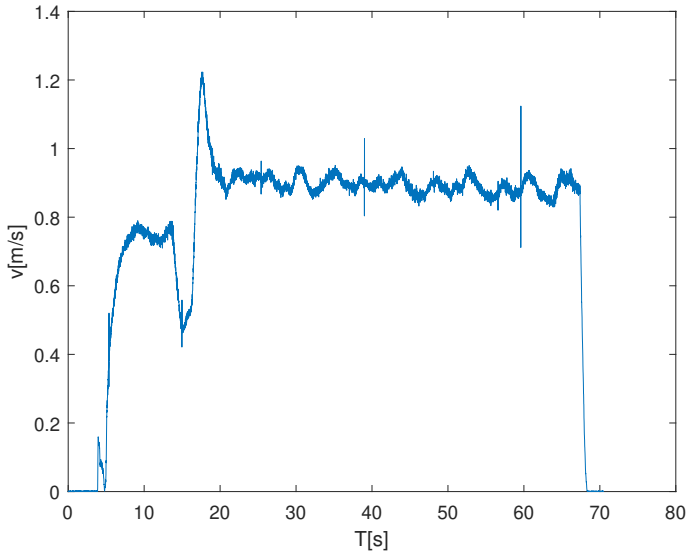
6.6 Zrychlení

Diferencováním průběhu rychlosti získáme průběh zrychlení (obrázek 6.7). V průběhu zrychlení se velmi promítá šum měření. Pro případné použití by bylo nutné alespoň část šumu nějakým způsobem odfiltrovat.

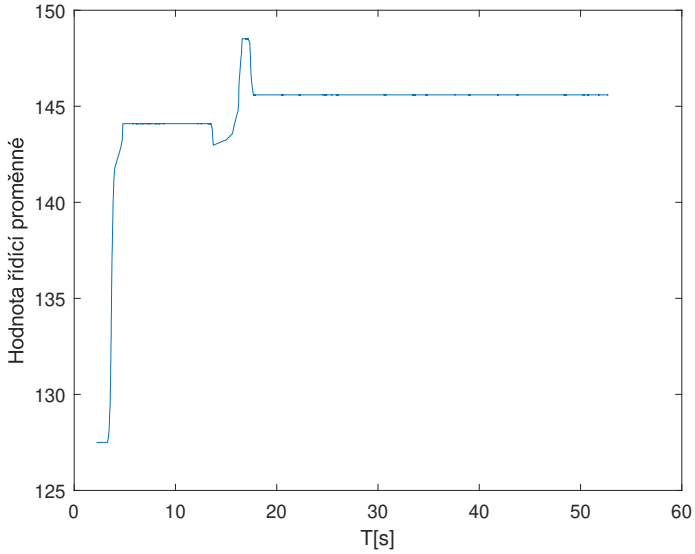
6.7 Statická charakteristika

Z obrázků 6.5 a 6.6 je zřejmé, že rychlost je závislá na hodnotě řídící proměnné. Tato část se zabývá tím, jaký je konkrétní vztah mezi řídící proměnnou a rychlostí RC modelu.

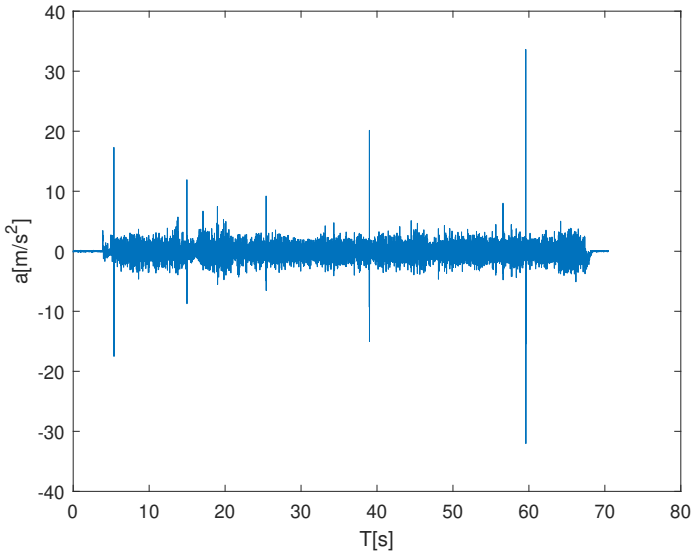
Při měření statické charakteristiky jsem narazil na problém s nedostatečně velkým měřícím prostorem pro měření po rovné dráze. RC model má poměrně vysoké maximální rychlosti, kterých však nestihne dosáhnout a ustálit se v omezeném prostoru laboratoře. K tomu je třeba počítat i s



Obrázek 6.5: Příklad rychlosti RC modelu v čase



Obrázek 6.6: Příklad průběhu řídící proměnné



Obrázek 6.7: Příklad průběhu zrychlení

dostatečným prostorem pro bezpečnou brzdnou dráhu. Z tohoto důvodu jsem se rozhodl měřit statickou charakteristiku na kružnici. Měřící trajektorie by neměla mít příliš malý poloměr, aby nedocházelo k prokluzování kol.

Nastavení řídící proměnné můžeme získat z modelu Ackermannova řízení. Využijeme již zmíněnou rovnici

$$\alpha = \frac{\pi}{2} - \arctan \frac{R}{d} \tag{6.6}$$

Do té dosadíme naměřený rozchod kol $d = 11cm$ a požadovaný poloměr kružnice R, po které se má model pohybovat. Pro $R = 2m$ získáme

$$\alpha = \frac{\pi}{2} - \arctan \frac{2}{0.11} \tag{6.7}$$

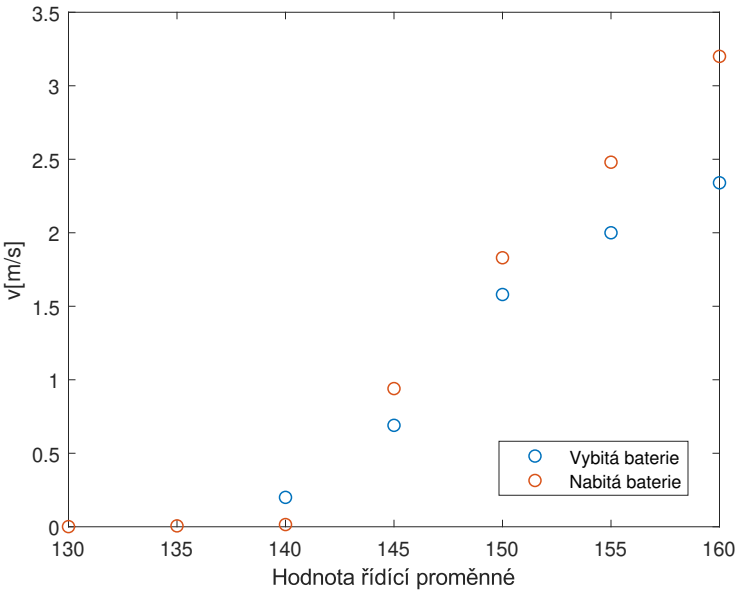
$$\alpha \approx 0.055rad \tag{6.8}$$

Při rychlosti kolem 3m/s se model dostává do nekontrolovaného smyku i při poměrně nízkém natočení kol. Této rychlosti model dosahuje při vstupu kolem hodnoty 160. Z tohoto důvodu nebyla odměřena statická charakteristika pro celý rozsah vstupu.

Statická charakteristika byla naměřena pro dva stavy baterie, modré body označují rychlost, na které se model ustálil, když jeho baterie byla těsně před úplným vybitím. Oranžové body v grafu 6.8 ukazují ustálenou rychlost ihned po výměně baterie.

6.8 Ladění PI regulátoru pro kontrolu rychlosti

K ladění rychlostního PI regulátoru byl využit speciální skript. Ten komunikuje pomocí ROS parameter server s grafickým rozhraním. Zde je možné nastavit parametry regulátoru a pomocí tlačítka skript spustit, ten se následně po dobu tří vteřin pokouší zrychlit a udržovat rychlost 1m/s.



Obrázek 6.8: Statická charakteristika

Tímto způsobem jsem testoval různá nastavení PI regulátoru. Regulátor s hodnotami $K_p = 38$ a $K_i = 42$ je poměrně rychlý a zároveň u něj nedochází k překmitu, který by při řízení rychlosti mohl být nebezpečný.

6.9 Experimentální ověření sledování trasy

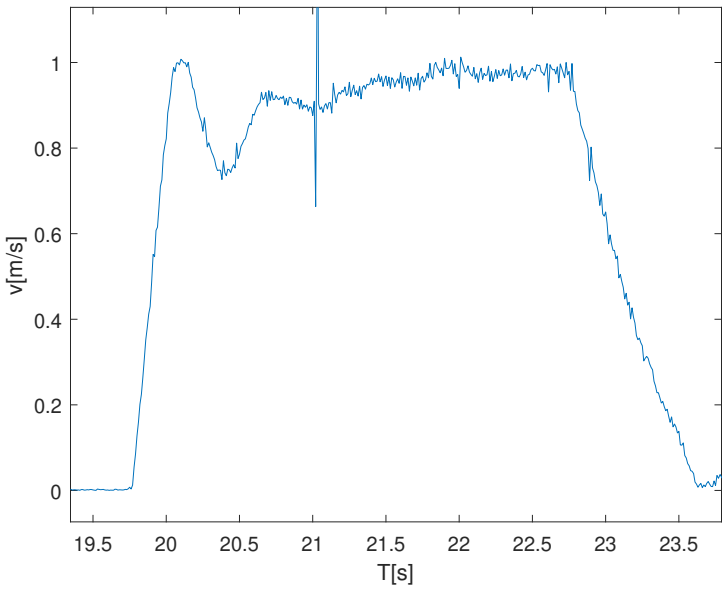
Pro ověření funkčnosti algoritmu jsem manuálně navrhl jednoduchou trasu a nechal ji modelem projet. Z grafu 6.10 je patrné, že s projížděním ostrých zatáček má RC model problém, ale mimo ně se drží zadané trasy poměrně přesně.

Následně jsem testoval opakovatelnost sledování trasy pro různé počáteční podmínky. Z grafu 6.11 je vidět, že velmi rychle došlo ke srovnání průjezdu a zbytek trasy projel model v obou případech téměř totožně.

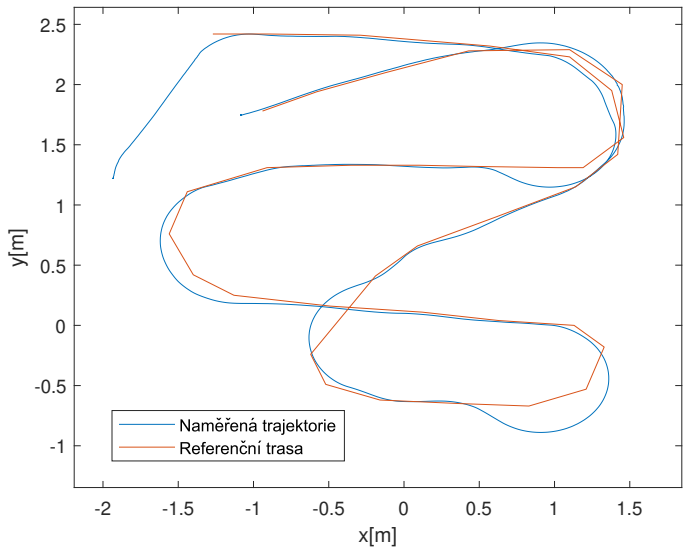
6.10 Experimentální ověření simulace trajektorie

Sledovanou trasu z předchozí kapitoly jsem simuloval Ackermannovým modelem. Natočení kol bylo získáno přímo z řídícího skriptu, do kterého byly na vstup místo naměřené polohy z Viconu přivedeny souřadnice z Ackermannových rovnic. Simulovanou trajektorii jsem vykreslil do grafu 6.12 s experimentálně naměřenou trajektorií.

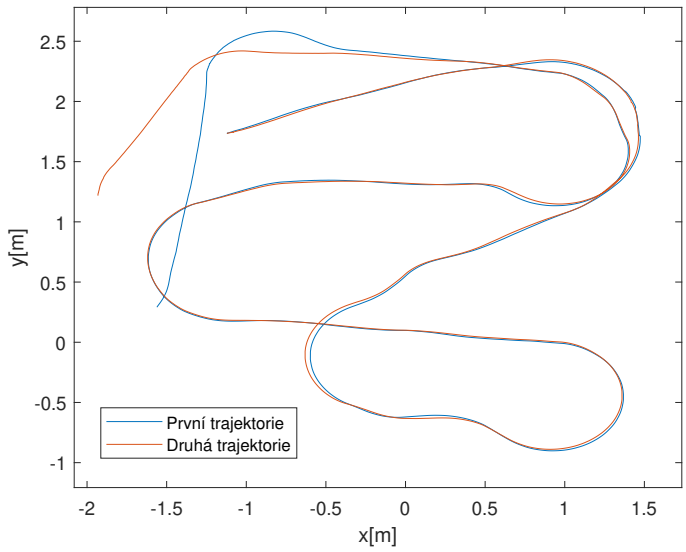
Simulovaná trajektorie poměrně přesně odpovídá skutečné naměřené trajektorii RC modelu.



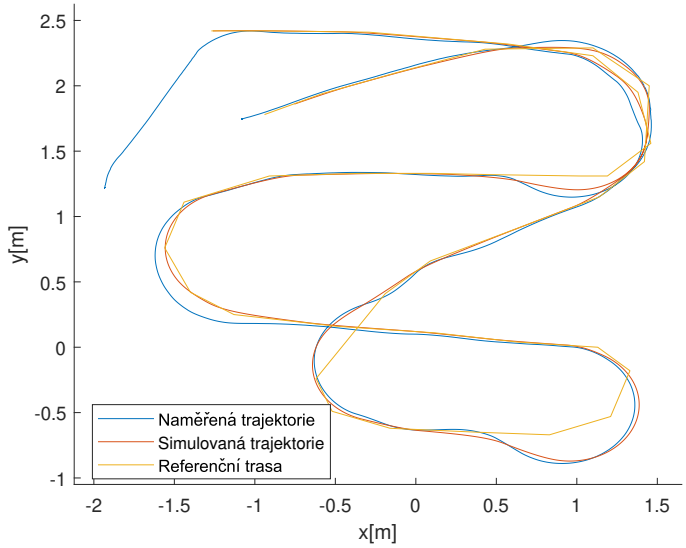
Obrázek 6.9: Přechodová charakteristika regulátoru s hodnotami $K_p = 38$ a $K_i = 42$



Obrázek 6.10: Srovnání naměřené trajektorie a navrhované trasy



Obrázek 6.11: Srovnání trajektorií ze dvou měření



Obrázek 6.12: Srovnání naměřené a simulované trajektorie

Kapitola 7

Plánování trasy a rychlosti

Naplánování trasy z bodu A do bodu B je samo o sobě vědním oborem. Plánování trasy se zabývá tím, jak naplánovat pohyby a manévry robota v pracovním prostoru za účelem dosažení zadaných cílů. Hlavním problémem je výpočet trasy bez rizika kolize od počáteční do cílové pozice. Velmi často krom nalezení trasy musí robot splnit nějaké další požadavky nebo optimalizovat určitá kritéria. Plánování trasy se rozděluje podle znalosti prostředí na zcela známé, částečně známé a zcela neznámé. Ve většině případů se jedná o částečně známé prostředí, kde robot má před plánováním nějakou informaci o prostředí [19].

Aktuální konfiguraci robota, jeho polohu a natočení v pracovním prostředí můžeme reprezentovat jako bod ve stavovém prostoru, který obsahuje všechny možné stavy robota. Robot se může pohybovat mezi jednotlivými stavy implementováním různých akcí. Trasa je tedy popsána jako sekvence akcí, které dostanou robota z počátečního stavu do stavu cílového. Algoritmus vybírá v každém bodě z množiny všech možných stavů, do kterých se dá z aktuálního stavu dostat. Mezi počátečním stavem a cílovým stavem může být větší množství tras. Z těchto tras algoritmus vybírá podle požadovaného kritéria. Mezi ta mohou patřit

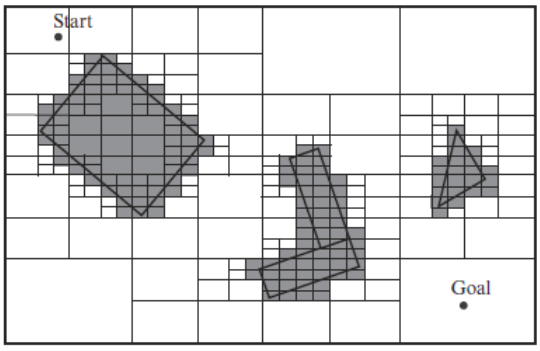
- Délka trasy
- Časová náročnost trasy
- Vzdálenost od překážek
- Výskyt ostrých zatáček
- Splnitelnost robotem - robot nemá možnost pohybovat se všemi směry [7]

7.1 Metody reprezentace prostředí

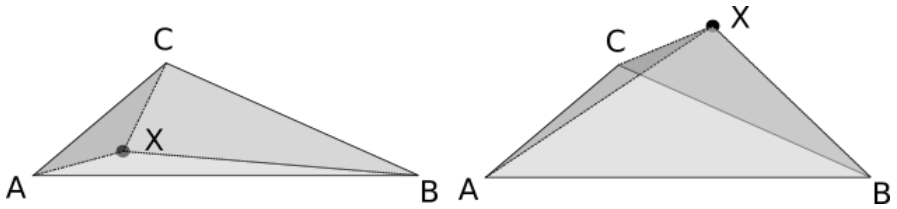
Před samotným plánováním trasy je nutné vytvořit matematickou reprezentaci prostředí vhodnou pro plánovací algoritmy.

7.1.1 Přibližné rozdělení do buněk

Prostředí je rozděleno do buněk, buňky, které alespoň částečně obsahují překážku jsou označeny jako obsazené, zbylé buňky jsou volné. Tato metoda je implementačně jednoduchá, ale část informace o prostředí se ztrácí. Pro efektivnější využití paměti je možné rozdělit prostředí do buněk různých velikostí [7].



Obrázek 7.1: Přibližné rozdělení do buněk různých velikostí [7]



Obrázek 7.2: Příklad ověření, zda se bod X nachází uvnitř trojúhelníku ABC

7.1.2 Přesné rozdělení do buněk

Rozdělení do buněk je přesné, pokud všechny buňky zcela leží ve volném prostoru nebo překážce. Přesné rozdělení je bezztrátové, jelikož spojení všech volných buněk je rovno volné části prostředí. Nevýhodou může být komplikovanější implementace [7].

7.1.3 Mapa cest

Mapa cest se skládá z úseček, křivek a jejich bodů křížení, které popisují cesty, kterými se robot může vydat. Při hledání trasy je cílem spojit počáteční bod s cílovým bodem za pomocí existujících cest. Problémem je nalezení minimálního počtu cest, které by umožnily přístup do libovolné volné části prostředí [7].

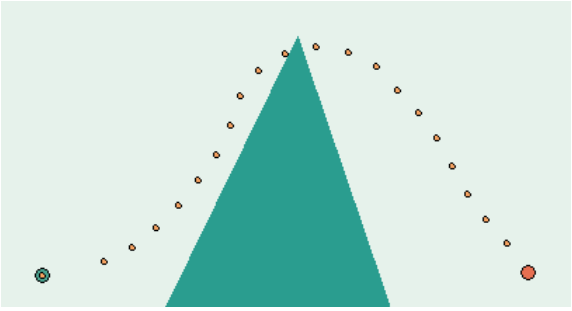
7.1.4 Použitá implementace reprezentace překážek

Rozhodl jsem se použít bezztrátovou metodu přesného rozdělení do buněk. Uživatelem jsou definovány konvexní mnohoúhelníky, které jsou uloženy jako seznam bodů. Ihned po vytvoření překážky je spočten obsah a těžiště mnohoúhelníku.

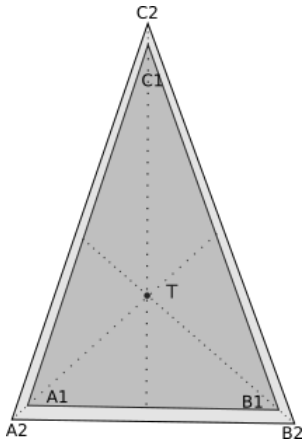
Při ověřování, zda se nějaký bod nachází uvnitř této překážky, vytvoříme z každé strany mnohoúhelníku a testovaného bodu trojúhelník. Následně porovnáme obsah sumy všech těchto trojúhelníků s obsahem původního mnohoúhelníku. Je-li obsah mnohoúhelníku shodný se sumou obsahů trojúhelníků, bod se nachází uvnitř překážky.

Na obrázku 7.2 jsou vidět dva příklady ověření, zda se bod X nachází uvnitř trojúhelníku ABC. V prvním případě

$$S_{ABC} = S_{ABX} + S_{BCX} + S_{CAX} \tag{7.1}$$



Obrázek 7.3: Chybná kolizní dráha



Obrázek 7.4: Bezpečnostní zóna kolem překážky

Bod X je tedy uvnitř trojúhelníku. Ve druhém případě

$$S_{ABC} < S_{ABX} + S_{BCX} + S_{CAX} \tag{7.2}$$

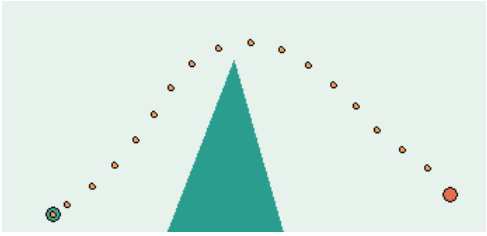
což ukazuje, že se bod X nachází mimo trojúhelník ABC.

Při návrhu trasy je RC model reprezentován bodem, algoritmus tedy nezohledňuje jeho skutečné rozměry. Kvůli tomu může algoritmus navrhnout trasu, která se příliš přiblíží k překážce a může tak dojít ke kolizi. Dalším problémem je diskrétní návrh trasy. Pokud je diskrétní krok větší, než překážka nebo její část, může docházet k ignorování této překážky, viz obrázek 7.3.

Tento problém jsem se rozhodl vyřešit zvětšením reprezentace překážek, čímž vznikne kolem každé překážky bezpečnostní zóna, do které se model nesmí přiblížit. Při zvětšení je nejprve vypočten normovaný směrový vektor mezi těžištěm překážky a každým vrcholem mnohoúhelníku. Následně je tento vektor vynásoben konstantou určující velikost bezpečnostní zóny a každý vrchol překážky je posunut tímto vektorem.

7.2 Generování konfiguračního prostoru

RC model umožňuje řízení momentu motoru pomocí celočíselné proměnné v rozsahu od 0 do 255. Natočení kol je řízené proměnnou ve stejném rozsahu. V každé periodě se tak může RC model přesunout



Obrázek 7.5: Trasa s bezpečnostní zónou

do některého z celkem 65 536 stavů. Při řízení s frekvencí 20Hz tak může po jedné vteřině existovat až $65536^{20} \approx 2 \cdot 10^{96}$ možných stavů.

Pokud počítáme s dokonalou trakcí a dostatečně vysokou periodou vzorkování, tvar finální trajektorie není závislý na rychlosti. Při hledání nejkratší trasy můžeme tedy zvolit rychlost konstantní. Tím výrazně snížíme počet generovaných stavů. Pro zrychlení výpočtu je možné snížit i citlivost natáčení kol, v tu chvíli však hledáme nejkratší trasu pro zjednodušeného robota, která může být delší než optimální trasa v původní úloze.

7.3 A* algoritmus

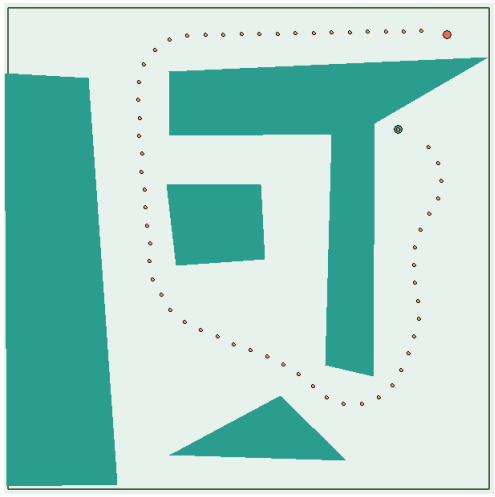
I při zanedbání rychlosti a radikálním snížení citlivosti natáčení kol je počet generovatelných stavů stále příliš velký na to, aby byly generovány a testovány všechny. Pro prohledání jsem se tedy rozhodl použít algoritmus A*, který využívá dodatečnou znalost pro efektivní prohledávání stavového prostoru. Algoritmus A* počítá pro každý stav heuristiku, tedy odhadovanou vzdálenost od současného stavu do cíle. To umožňuje algoritmu rozlišit mezi více a méně slibnými stavy. Touto heuristikou může být například eukleidovská vzdálenost. Při běhu algoritmu je pro každý stav spočítána vzdálenost k dosažení současného stavu a odhad vzdálenosti do cíle. Stav, pro který mají být vygenerováni jeho nástupci, je vybrán jako stav s minimálním součtem ceny do tohoto stavu a odhadu ceny od tohoto stavu do cíle [7].

7.4 Použitý algoritmus hledání nejkratší trasy

Pro další zrychlení prohledávání jsem se rozhodl při generování stavů kontrolovat, zda se již podobný stav v algoritmu vyskytl, a v takovém případě ho dále nevyužívat. Pracovní plocha je tak rozdělena sítí s určitou přesností. Při vygenerování nového stavu dojde k uložení čtverce, do kterého tento stav spadá a v případě vygenerování dalšího stavu spadajícího do tohoto čtverce je tento stav zapomenut, jelikož nemůže vést k výraznému zlepšení. Tato část algoritmu může vést k nalezení suboptimálního řešení. U složitějších návrhů tras však hledání velmi výrazně zrychluje.

Vstupem algoritmu je počáteční a cílový stav. Počátečnímu stavu je přidělená nulová cena a je nastaven jako aktuální stav. Následně je spuštěn prohledávací cyklus s následujícími kroky.

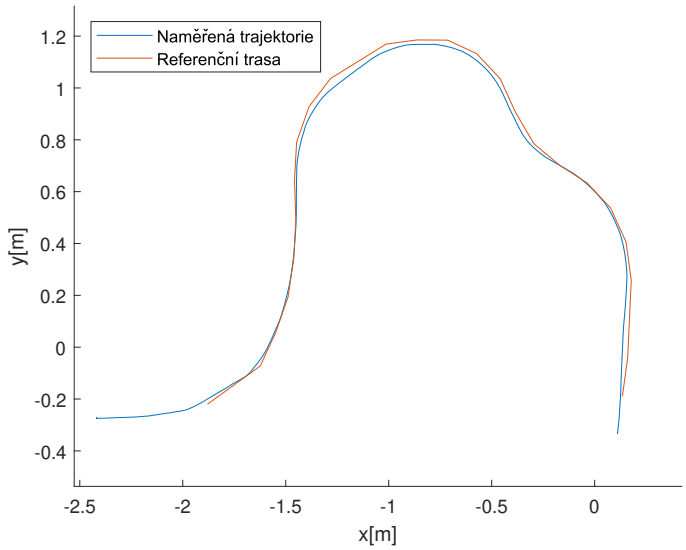
- Rozšíření aktuálního stavu - podle nastavené citlivosti natočení kol jsou vygenerovány všechny možné stavy pro další krok.
- Ověření překážky - pokud se stav nachází uvnitř nebo příliš blízko překážce, je zapomenut.
- Ověření podobnosti s předchozími stavy - pokud již podobný stav byl vygenerován, je tento stav zapomenut.



Obrázek 7.6: Příklad trasy navržené algoritmem

- Spočtení ceny stavu jako $f(x) = g(x) + h(x)$, kde g(x) je cena od počátku do toho stavu, je spočtena jako $g(x_p) + vdt$, kde x_p je stav, ze kterého byl aktuální stav vygenerován. $h(x)$ je eukleidovská vzdálenost mezi aktuálním stavem a cílem.
- Vložení stavu do binární haldy - binární halda byla vybrána pro nízkou náročnost vložení prvku O(log N) a velmi nízkou náročnost nalezení minima O(1).
- Získání minima z binární haldy a nastavení tohoto stavu jako aktuální stav.
- Kontrola zda aktuální stav je dostatečně blízko cíli. Pokud je, je cyklus ukončen a je rekonstruována celá trasa.
- Návrat na počátek cyklu.

```
while(True)
  for uhel in pocetUhlu
    nasledujiciStav.Z = stavajiciStav.Z+dt*v/d*tan(natoceniKol)
    nasledujiciStav.X = stavajiciStav.X+dt*v*cos(nasledujiciStav.Z)
    nasledujiciStav.Y = stavajiciStav.Y+dt*v*sin(nasledujiciStav.Z)
    nasledujiciStav.predchazejici = stavajiciStav
    if nekolidujeSPrekazkou(nasledujiciStav) AND overeniPodobnosti(
       nasledujiciStav)
      nasledujiciStav.cena = cenaKTomutoStavu+vzdalenostOdCile
      binarniHalda.INSERT(nasledujiciStav)
  stavajiciStav=binarniHalda.POP()
  if neexistuje(stavajiciStav)
    #Mezi zadanymi body nelze nalezt trasu
  if stavajiciStav.vzdalenost(cil)<pozadovanaPresnost
    while stavajiciStav != pocatecniStav
      trasa.append(stavajiciStav)
      stavajiciStav = stavajiciStav.predchazejici
      trasa = reverse(trasa)
```



Obrázek 7.7: Srovnání navržené trasy a naměřené trajektorie

7.5 Rozšíření návrhu trasy o možnost couvání

Fyzický RC model umožňuje couvání nastavením řídící proměnné na hodnotu mezi 0 a 127. Možnost couvání může vést k nalezení rychlejší trasy, v některých případech může dokonce umožnit propojení bodů, které bez couvání nebylo možné. Je s ním tedy vhodné počítat při návrhu trasy.

Při generování okolních stavů v okolí rozšiřovaného stavu jsou tedy vygenerovány i stavy pro zápornou rychlost a u každého stavu je uložena informace o směru pohybu.

Algoritmus nejprve mezi couváním a dopředným pohybem nerozlišoval a v řadě případů měl tendenci couvat celou cestu. Při reálném řízení však není dlouhé couvání výhodné, rozhodl jsem se tedy algoritmus za couvání penalizovat snížením rychlosti při generování stavů pohybu vzad. Taková penalizace se projeví pouze při časové optimalizaci, při hledání nejkratší trasy je tak couvání po celou trasu poměrně běžné.

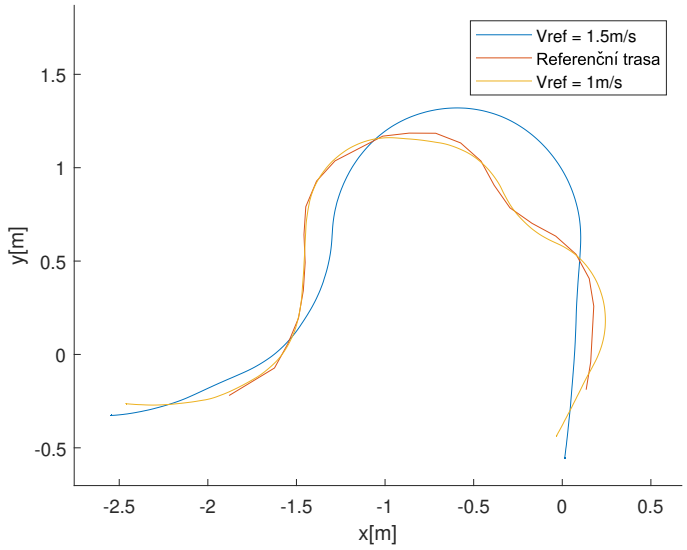
7.6 Experimentální průjezd generovanou trasou

V této kapitole je ověřena funkčnost algoritmu z předchozí kapitoly. Nejprve byly pomocí grafického rozhraní zakresleny překážky a následně byla algoritmem navržena trasa. Tato trasa byla poté použita jako referenční při průjezdu RC modelu. Při prvním průjezdu byla nastavena referenční rychlost na 1m/s. Při druhém byla zvýšena na 1.5m/s. V obrázku 7.7 je vykreslen průjezd při referenční rychlosti 1m/s a navržená referenční trasa.

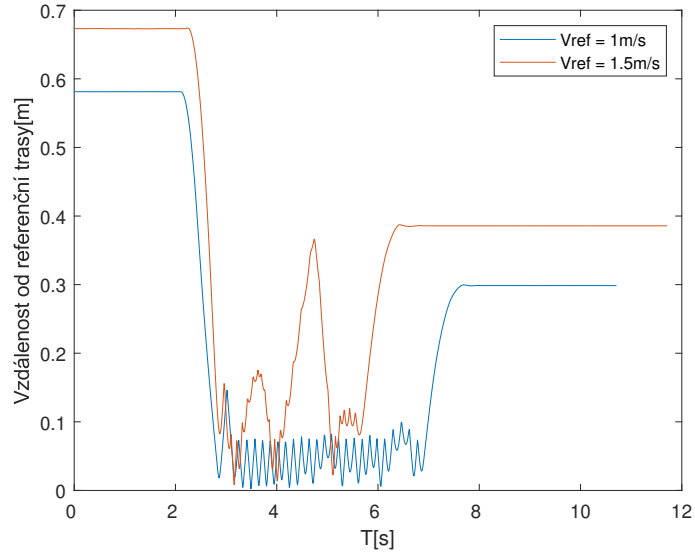
Při hledání nejkratší trasy je výsledná trasa generována s ohledem na parametry RC modelu. Při experimentálním průjezdu navrženou trasou nižší rychlostí je tedy model schopný ji sledovat s vysokou přesností (obrázek 7.7) .

Z trajektorií průjezdu referenční trasy při různých referenčních rychlostech (obrázek 7.8) je patrné, že pro rychlosti kolem 1m/s není sledování referenční trasy problém, ale již při zvýšení na 1.5m/s dochází ke znatelné odchylce.

Na obrázku 7.9 jsou zobrazené minimální vzdálenosti mezi referenční trasou a měřenou trajektorií. Pro každý bod trajektorie je nalezen nejbližší bod z referenční trasy a je vykreslena jejich eukleidovská



Obrázek 7.8: Srovnání průjezdu při různých referenčních rychlostech



Obrázek 7.9: Nejkratší vzdálenost od referenční trasy

vzdálenost v metrech. Zde je patrné nejprve přiblížení k prvnímu bodu referenční trasy a následně její sledování. Při referenční rychlosti 1m/s nepřekročí tato vzdálenost 10cm. Při referenční rychlosti 1.5m/s tato vzdálenost při sledování dosahuje téměř 37cm. Cílem je dosáhnout cílového bodu co nejrychleji, proto se RC model zastaví až po jeho projetí, což způsobuje nárůst minimální vzdálenosti po projetí trasy.

Kapitola 8

Plánování trasy a rychlosti s ohledem na prokluzování pneumatik

Nejkratší trasu je možné sledovat za nízkých rychlostí, ale při zvýšení referenční rychlosti dochází k prokluzování, které původní plánovací algoritmus nebral v potaz. Tato kapitola se zabývá tím, jak započítat prokluz pneumatik při návrhu trasy a jak zvolit vhodnou referenční rychlost pro časově optimální jízdu.

8.1 Model pneumatiky

Vzhledem k tomu, že pneumatiky jsou během jízdy v přímém kontaktu s vozovkou, mají zásadní vliv na dynamické vlastnosti vozidla. Při reálné jízdě vozidla nebo RC modelu není trakce dokonalá a dochází k prokluzování kol vlivem sil působících na vozidlo. To způsobuje odchylku mezi rychlostí otáčení kol a rychlostí pohybu vozidla a také odchylku mezi natočením kol a skutečným směrem zatáčení.

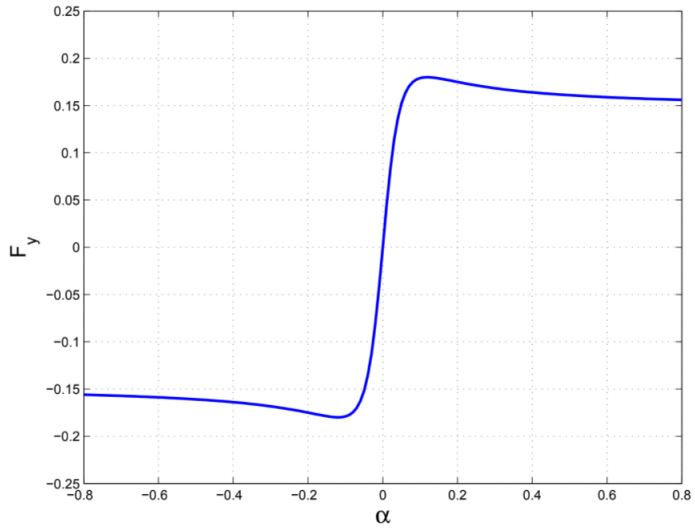
Existuje mnoho různých způsobů, jak toto chování pneumatiky modelovat. Jedním z nich je Pacejkův model pneumatiky publikovaný v roce 1987. Ten popisuje vztah mezi laterálními silami působícími na vozidlo a úhlem skluzu [12].

V grafu 8.1 je vidět běžný tvar závislosti sil na skluzu. Konkrétní parametry jsou odlišné pro různé kombinace pneumatik a povrchů, po kterých se vozidlo pohybuje, tvar však zůstává podobný. V určitém silovém intervalu, v případě tohoto grafu je to přibližně mezi -0.15 a 0.15, dochází ke smyku jen velmi málo. Po překročení kritické hranice pak dochází k rychlému nárůstu skluzového úhlu a vozidlo se dostává do smyku.

Rozhodl jsem se omezit maximální rychlost tak, aby laterální síly nepřekročily kritickou hranici a smykový úhel tak bylo možné zanedbat. Případné malé odchylky způsobené nedokonalou trakcí by měly být kompenzovány zpětnovazebním řízením.

8.2 Optimální nastavení referenční rychlosti při průjezdu zatáčkou

Z modelu pneumatiky plyne existence kritické laterální síly F_{max}, při jejímž překročení dojde ke smyku. Při průjezdu zatáčkou se jedná především o odstředivou sílu, kterou vozidlo působí na vozovku. Při optimalizaci se tedy snažíme nalézt nejvyšší možnou rychlost, při které odstředivá síla nepřekoná



Obrázek 8.1: Příklad Pacejkova modelu pneumatiky [12]

maximum vyplývající z modelu pneumatiky.

$$F_{max} \geq \frac{mv^2}{r} \tag{8.1}$$

kde v je rychlost vozidla a r je poloměr křivosti. $\frac{v}{r}$ nahradíme úhlovou rychlostí a získáme vztah.

$$F_{max} \geq m\omega v \tag{8.2}$$

Za ω můžeme dosadit vztah pro úhlovou rychlost z Ackermannova řízení.

$$F_{max} \geq m(v \cdot \frac{1}{d} \cdot \tan(\alpha))v \tag{8.3}$$

Z této nerovnice můžeme vyjádřit v^2 jako

$$v^2 < \frac{F_{max}d}{m\tan(\alpha)} \tag{8.4}$$

$$v < \pm\frac{\sqrt{d}\sqrt{F_{max}}}{\sqrt{m}\sqrt{\tan(\alpha)}} \tag{8.5}$$

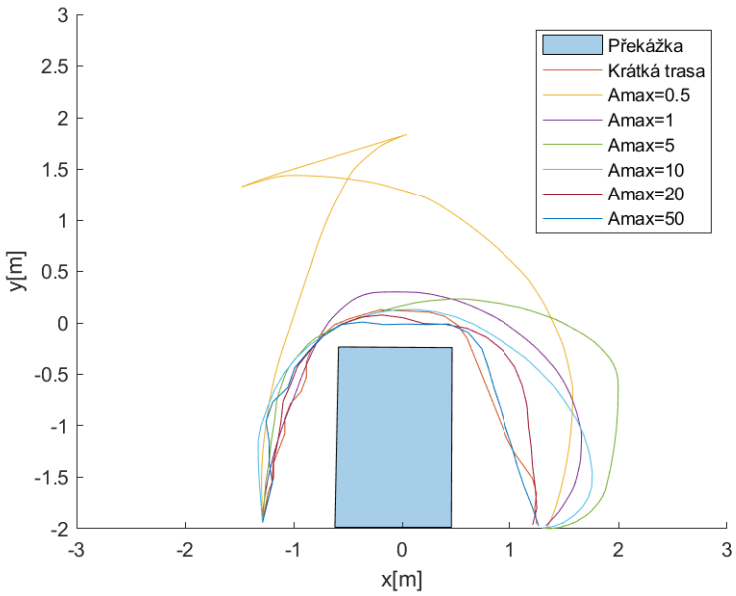
Za d je možné dosadit naměřený rozchod kol $d = 11cm$. F_{max} je potřeba odměřit experimentálně. Pro zjednodušení experimentu je možné nahradit maximální odstředivou sílu maximálním odstředivým zrychlením.

$$v^2 < \frac{a_{max}d}{\tan(\alpha)} \tag{8.6}$$

Pro α blížící se 0 nepůsobí odstředivá síla a maximální možná rychlost se tak limitně blíží nekonečnu. Při plánování trasy je referenční rychlost omezena maximální rychlostí RC modelu.

8.3 Hledání časově optimální trasy

Pro hledání časově optimální trasy byl použit stejný algoritmus jako při hledání nejkratší trasy, pouze s odlišnou cenovou funkcí a generováním nových stavů.



Obrázek 8.2: Srovnání tras s různými a_{max}

Před samotným během algoritmu je třeba nastavit tři konstanty. Rozvor modelu v metrech, konstantu maximálního zrychlení, ta určuje maximální odstředivé zrychlení přípustné pro pneumatiku a je možné ji odměřit experimentálně. Třetí konstantou je maximální rychlost, tedy referenční rychlost, kterou by se měl model pohybovat při jízdě dostatečně rovné na to, aby nebyla omezena odstředivým zrychlením.

Při generování nových stavů je třeba zajistit, aby při jejich průjezdu nedocházelo ke smyku, na počátku algoritmu je tedy pro každé přípustné natočení kol α vypočtena maximální rychlost s využitím vzorců z předchozí kapitoly.

$$v_{ref} = \min\left(\frac{\sqrt{a_{max}}\sqrt{d}}{\sqrt{\tan(|\alpha|)}}, v_{max}\right) \tag{8.7}$$

Při hledání časově optimální trasy byl změněn výpočet ceny stavu $f(x) = g(x) + h(x)$, kde $g(x) = g(x_p)+dt$, kde x_p je stav, ze kterého byl aktuální stav vygenerován. $h(x)$ je stále eukleidovská vzdálenost mezi aktuálním stavem a cílem.

V grafu 8.2 je vidět návrh sedmi různých tras. Červeně je zde trasa navržená původním algoritmem minimalizujícím ujetou vzdálenost. Dále je v grafu zobrazeno šest různých tras pro různé hodnoty maximálního povoleného odstředivého zrychlení. Maximální povolená rychlost byla 4m/s. Pro velmi malá povolená odstředivá zrychlení algoritmus navrhuje tak nízké natočení kol, že je nutné i v takto jednoduché trase zacouvat, aby se RC model dostal do cíle. Algoritmus pro a_{max} 1, 5 a 10 různým způsobem prodlužuje oblouk trasy, aby se snížilo natočení kol a model se mohl pohybovat vyšší rychlostí. Pro agresivnější a_{max} začíná převládat vliv omezení maximální rychlostí a navržená časově optimální trasa se s rostoucím a_{max} blíží nejkratší možné trase.

Jelikož algoritmus jako cenovou funkci používá čas a vzdálenost do cíle, je cena cílového bodu rovna časovému odhadu pro průjezd dané trasy.

$a_{max}[m/s^2]$	$T[s]$
0.5	11.13
1	4.49
5	2.52
10	1.87
20	1.42
50	1.40

Tabulka 8.1: Odhadovaný čas průjezdu pro různé hodnoty a_{max}

8.4 Detekce kolize a vybití baterie

Při řízení může dojít k tomu, že rychlost RC modelu ani vzdáleně neodpovídá vstupní řídící proměnné. Nejčastěji to bývá při vybití baterie, kdy napětí není dostatečné k rozjetí RC modelu a při kolizi, kdy je model blokován překážkou, proti které se pokouší akcelerovat. Především druhý případ je vhodné automaticky detekovat, aby nedocházelo zbytečně k protáčení kol na místě.

Prvním znakem problému může být vysoká hodnota řídící proměnné. Pokud je zároveň s ní rychlost blízká nule, došlo pravděpodobně k vybití nebo kolizi. V takovém případě nemá cenu pokračovat v původní trase. Před ukončením skriptu je však možné uživateli sdělit k jakému problému došlo. Skript tedy nejprve vyzkouší postupně snižovat řídící proměnnou, pokud RC model začne couvat, skript detekuje zvýšení rychlosti a určí, že šlo o kolizi s překážkou. V opačném případě šlo pravděpodobně o vybití baterie.

Kapitola 9

Závěr

V této práci bylo nejprve krátce popsáno několik projektů zabývajících se problematikou autonomního řízení RC modelů i plnohodnotných automobilů. Dále bylo shrnuto a srovnáno několik metod určování polohy. Poté bylo chování RC modelu otestováno při manuálním ovládání. Následně byl vytvořen jednoduchý skript, který přes Robot Operating System posílal předem definované příkazy RC modelu. Tento skript byl dále rozšířen o grafické rozhraní a řadu dalších nástrojů a funkcí. Z kinematiky Ackermannova řízení bylo odvozeno zpětnovazební řízení za účelem sledování trasy. Následně byl s pomocí nástroje pro ladění regulátoru navržen PI regulátor pro udržování referenční rychlosti. Další kapitoly se zabývaly metodami reprezentace prostředí a návrhem nejprve nejkratší a následně časově optimální trasy tak, aby nedošlo ke kolizi s překážkou nebo smyku RC modelu. Na konec byl pro bezpečnější a jednodušší ovládání navržen detektor vybití a kolize.

Prvním cílem bylo popsat a srovnat různé metody lokalizace, tímto se zabývá kapitola Systémy určování polohy a rychlosti, ve které jsou popsané enkodéry, IMU, satelitní poziční systémy a motion capture systémy.

Druhým cílem bylo zvolit vhodný způsob modelování kinematiky RC modelu auta. K modelování kinematiky byl použit Ackermannův model, ten byl následně použit při simulování průjezdu referenční trasou, kde poměrně přesně odpovídal průjezdu RC modelu stejnou trasou. Dále byl využit při plánování trasy, aby byla pro RC model dobře průjezdná.

Třetím cílem bylo navrhnout řídící algoritmus RC modelu za účelem sledování referenční trasy a rychlosti. K tomu byla použita dvojice PI regulátorů. Navržený systém byl schopný sledovat manuálně navržené trasy za předpokladu, že byly navrženy vhodně a neobsahovaly příliš ostré zatáčky. I při rozdílných počátečních podmínkách byl průjezd trasou při opakování experimentu téměř totožný.

Čtvrtým cílem bylo navrhnout algoritmus pro automatické plánování trasy zohledňující pohybová omezení RC modelu a vnější překážky. Nejprve byl navržen algoritmus pro hledání nejkratší trasy, ten je schopný nalézt nejkratší trasu prostředím tak, aby byla dobře průjezdná RC modelem při nižších rychlostech a nedošlo při tom ke kolizi s překážkou. Dále byl vytvořen algoritmus pro plánování trasy, kterou je možné sledovat i za vyšších rychlostí a která by měla být časově optimální při zadaných podmínkách. Navržené trasy byly průjezdné RC modelem a jejich průjezd byl poměrně rychlý, ale pravděpodobně je možné dosáhnout i rychlejšího průjezdu například algoritmem, který by se nevyhýbal smyku a byl schopný ho řídit.

Posledním cílem bylo experimentálně ověřit kvalitu řídících i plánovacích algoritmů. Nejprve bylo otestováno manuální řízení RC modelu, dále sledování tras navržených manuálně i plánovacími algoritmy. K efektivnímu provádění experimentů bylo vytvořeno grafické rozhraní.

Podařilo se navrhnout řízení RC modelu auta, které umožňuje rychlý a bezpečný průjezd prostředím. Navazující práce by mohla ještě více optimalizovat plánovací algoritmy, vylepšit časovou optimalizaci přesnějším modelováním nebo využít jiného způsobu lokalizace RC modelu.

Bibliografie

[1] URL: http://www.chrobotics.com/library/understanding-quaternions.

[2] *atan2*. URL: https://www.mathworks.com/help/fixedpoint/ref/atan2.html.

[3] Ilia Baranov. *vicon_ bridge*. URL: http://wiki.ros.org/vicon_bridge#Nodes.

[4] Jose-Luis Blanco. "A tutorial on se (3) transformation parameterizations and on-manifold optimization". In: *University of Malaga, Tech. Rep* 3 (2010), s. 6.

[5] Scott Dyer, Jeff Martin a John Zulauf. "Motion capture white paper". In: (1995).

[6] Maureen Furniss. "Motion capture: An overview". In: *Animation Journal* 8.2 (2000), s. 68–82.

[7] Klančar Gregor et al. *Wheeled mobile robotics: from fundamentals towards autonomous systems*. Butterworth-Heinemann, an imprint of Elsevier, 2017.

[8] Bonnie Gringer. *History of the autonomous car*. Dub. 2020. URL: https://www.titlemax.com/resources/history-of-the-autonomous-car/.

[9] *Gyro sensors - how they work and what's ahead: about gyro sensor: technical information: other information*. URL: https://www5.epsondevice.com/en/information/technical_info/gyro/.

[10] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger a Elmar Wasle. *GNSS–global navigation satellite systems: GPS, GLONASS, Galileo, and more*. Springer Science & Business Media, 2007.

[11] Yunyi Jia, Longxiang Guo a Xin Wang. "Real-time control systems". In: *Transportation Cyber-Physical Systems*. Elsevier, 2018, s. 81–113.

[12] Alexander Liniger. "Autonomous Drift Control". Dis. 2012.

[13] Alexander Liniger. *ORCA Racer*. 2016. URL: https://sites.google.com/site/orcaracer/documentation.

[14] Losi. *1/24 Micro Rally X 4WD RTR Red*. URL: http://www.losi.com/Products/Features.aspx?ProdID=LOS00002T1 (cit. 13. 11. 2020).

[15] Brad Miller. *Parameter Server*. URL: http://wiki.ros.org/Parameter%20Server.

[16] Michael Montemerlo et al. "Junior: The stanford entry in the urban challenge". In: *Journal of field Robotics* 25.9 (2008), s. 569–597.

[17] Morgan Quigley, Brian Gerkey a William D Smart. *Programming Robots with ROS: a practical introduction to the Robot Operating System*. "O'Reilly Media, Inc.", 2015. ISBN: 978-1-4493-2389-9.

[18] Nikola Tesla. *Method of and apparatus for controlling mechanism of moving vessels or vehicles*. Lis. 1898.

[19] Spyros G Tzafestas. *Introduction to mobile robot control*. Elsevier, 2014.

[20] Robin Verschueren. "Design and implementation of a time-optimal controller for model race cars". Dis. Master's thesis, KU Leuven, 2014.

[21] Megan Wallace. *What is GPS?* Červ. 2019. URL: https://www.nasa.gov/directorates/heo/scan/communications/policy/what_is_gps.

[22] Qilong Yuan a I-Ming Chen. “Localization and velocity tracking of human via 3 IMU sensors”. In: *Sensors and Actuators A: Physical* 212 (2014), s. 25–33.