VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

PLÁNOVÁNÍ TRAJEKTORIE VOZIDLA SE ČTYŘMI NEZÁVISLE ŘÍZENÝMI KOLY
TRAJECTORY PLANNING OF VEHICLE WITH FOUR INDEPENDENTLY STEERED WHEELS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Roman Śniežek**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Martin Brablc**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Roman Šniežek**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**Ing. Martin Brablc**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Plánování trajektorie vozidla se čtyřmi nezávisle řízenými koly

Stručná charakteristika problematiky úkolu:

Cílem této práce je zpracovat kinematický model vozidla se čtyřmi nezávisle řízenými koly a použít jej pro plánování trajektorie takového vozidla v různých situacích.
První část práce bude zaměřená na návrh a simulační testování navržených metod v prostředí Matlab/Simulink. Následně budou navržené metody testovány na reálném experimentálním vozidle (výukový model dostupný v Mechatronické laboratoři).

Cíle bakalářské práce:

1) Proveďte rešerši mapující metody plánování trajektorie bodů i kolových vozidel a popište různé druhy jejich kinematických konstrukcí.
2) Vytvořte kinematický model vozidla se čtyřmi nezávisle řízenými koly a simulačně demonstrujte jeho správnost.
3) Implementujte alespoň 3 vybrané metody plánování cesty nebo trajektorie a simulačně otestujte jejich vhodnost pro použití v prostředí se statickými překážkami různého tvaru. Plánování trajektorie může být realizováno metodou path-following pomocí kinematického modelu. Alespoň jedna metoda musí obsahovat korekci nepřesností polohy a natočení vozidla.
4) S využitím výsledků předchozích závěrečných prací realizujte vhodnou metodou nahrávání trajektorie do řídícího systému Car4, případně proveďte nezbytné úpravy tak, aby bylo možné využít natáčení všech čtyř kol.
5) Jednu vybranou metodu plánování trajektorie implementujte pro experimentální vozidlo Car4 a realizujte demonstrační experiment - průjezd známým prostředím s překážkami.

Seznam doporučené literatury:

NELLES, Oliver. Nonlinear system identification: from classical approaches to neural networks and fuzzy models. Berlin: Springer, 2011. ISBN 978-364-2086-748.

LJUNG, Lennart. System identification: theory for the user. 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR, 1999. ISBN 978-0136566953.

VALÁŠEK, Michael. Mechatronika. Dot. 1. vyd. Praha: České vysoké učení technické, 1996. ISBN 80-010-1276-X.

NOSKIEVIČ, Petr. Modelování a identifikace systémů. Ostrava: Montanex, 1999. ISBN 80-722-50-0-2.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2019/20.

V Brně, dne 22. 10. 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ
DĚKANÁT
L. S.

prof. Ing. Jindřich Petruška, CSc.
ředitel ústavu

doc. Ing. Jaroslav Katolický, Ph.D.
děkan fakulty

Abstrakt

Práce se zabývá kinematickým modelem vozidla se čtyřmi nezávisle řízenými koly a jeho simulací. Dále se zabývá algoritmy pro plánování cesty. Cílem práce je využití kinematického modelu a algoritmu pro sledování cesty k vytvoření trajektorie jako vektoru natočení kol v závislosti na čase a ujeté vzdálenosti. Tato trajektorie umožní vozidlu přesun ze startovní pozice k cíli na mapě se statickými překážkami, aniž by došlo ke kolizi. To pak může být využito při jízdě autonomního vozidla ve známém prostředí

Summary

This thesis focuses on a kinematic model of a four-wheel steering vehicle and its simulation. The aim of the thesis is to use the kinematic model and path following algorithms to create a trajectory as a vector of the wheel steering angles dependent on a time and the distance travelled. This trajectory provides a collision-free transfer from starting position to the goal on a map with static obstacles. This could be used in an autonomous vehicle while navigating in a known enviroment

Klíčová slova

plánování cesty, plánování trajektorie, 4WS, sledování cesty, kinematický model, neholonomní kinematika, regulace

Keywords

path planning, trajectory planning, 4WS, path following, kinematic model, nonholonomic kinematics, regulation

Bibliografická Citace

ŚNIEŻEK, R. *Plánování trajektorie vozidla se čtyřmi nezávisle řízenými koly.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2020. 39 s., Vedoucí bakalárské práce: Ing. Martin Brablc.

Prohlašuji, že jsem bakalářskou práci na téma Plánování trajektorie vozidla se čtyřmi nezávisle řízenými koly vypracoval samostatně s použitím materiálů a zdrojů uvedených v seznamu literatury.

Roman Śniežek

Brno . . . .

Chtěl bych poděkovat vedoucímu práce, Ing. Martinu Brablcovi, za rady i trpělivost při vysvětlování a taky Bc. Tomáši Kirchnerovi za pomoc při seznamování se s projektem Car4 a při práci s ním.

Roman Śniežek

Obsah

1 Úvod

S postupnou automatizací a rychle se vyvíjející technologií roste i požadavek na algoritmy schopné samostatného řízení a rozhodování. Nejpopulárnějším tématem z této oblasti je samozřejmě umělá inteligence a neuronové sítě. Neméně podstatnou část moderní technologie tvoří ale také autonomní bezpilotní vozidla a drony, které ke své činnosti potřebují spolehlivé a efektivní náhrady řízení člověkem.

Společnost Amazon se například snaží využít drony k roznášení zásilek [1]. Tato služba nazývaná „Air Prime“ je momentálně ve fázi testování ve vybraných lokalitách. Ve vývoji jsou i vozidla schopné samostatné jízdy podle předpisů a dopravního značení. Toho je dosaženo opět v kombinaci s počítačovým viděním a plánováním trajektorie.

Plánování cest a trajektorií má tedy potenciál v širokém spektru uplatnění. Zároveň nejde o triviální problém, ale o komplexní úlohu zabývající se dynamikou, kinematikou a závěrečným sestavením algoritmů.

V této práci bude kladem důraz pouze na kinematiku vozidla. Dynamické chování je při výpočtech a tvorbě modelu zanedbáváno. Jediným faktorem, zohledňujícím dynamiku vozidla, bude omezení rychlosti při průjezdu zatáčkou. Tím budou omezeny setrvačné účinky reálného vozidla a nedojde tak ke smýkání kol.

Cesta bude vytvořená nejčastěji se vyskytujícími typy algoritmů pro plánování. Kinematický model bude využitý při sledování této cesty. Tato forma tvorby trajektorie byla vybrána vzhledem k její jednoduchosti a zajímavým možnostem, které přináší. Ty plynou z využití samotného modelu a jeho omezení.

Mezi jiné řešení tohoto problému by bylo proložení cesty křivkou. Vzhledem k požadovaným podmínkám pro polohy, rychlosti, poloměr křivosti a případně zrychlení je ale jednodušší volbou sledování cesty. Ta už z principu využití kinematického modelu splňuje veškeré jeho omezení. Křivka, kterou by se tyto podmínky daly splnit a cestu proložit, je obvykle polynom vyššího stupně[1], spline, nebo - pro vozidla vhodnější - úseky klotoidy. Ta je významná tím, že její poloměr křivosti je po celé délce křivky spojitý a díky této vlastnosti je často používaná při plánování cest nebo stavbě vozovek.

Celý algoritmus měl být v závěru práce experimentálně ověřen na hotovém projektu Car4. Ten je výsledkem několika bakalářských a diplomových prací v mechatronické laboratoři. Vzhledem k uzavření školy a laboratoří byly ale cíle bakalářské práce po dohodě s vedoucím změněny a vozidlo, společně s přenosem dat do mikroprocesoru, bude simulováno v prostředí MATLAB.

Závěrečný experiment tedy proběhne pouze jako simulace vozidla s nahranými hodnotami trajektorie, na známé mapě se statickými překážkami různého tvaru, opět za využití kinematického modelu.

[1] Většinou polynomy 5. a 6. stupně, podle počtu podmínek.

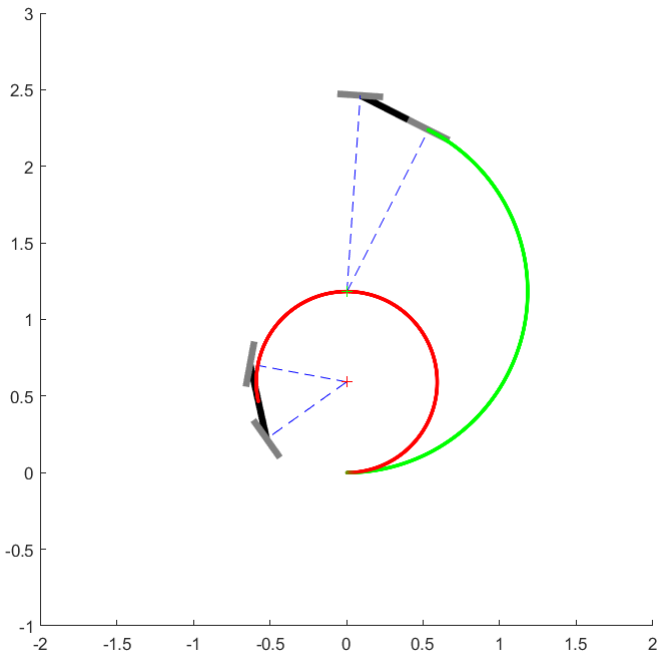
2 Rešeršní část

2.1 Kinematické konstrukce

V úvodu budou charakterizovány obecné možnosti, které s sebou přináší vozidla se čtyřmi nezávisle řízenými koly, dále jen 4WS vozidla[1]. Poté budou popsány způsoby natáčení kol u sériově vyráběných vozidel s nezávislým otáčením přední a zadní nápravy.

Nezávislého řízení kol se využívá především tam, kde je potřeba lepší manévrovatelnost vozidla ve složitém terénu. Nejčastěji můžeme toto řízení nalézt u terénních vozidel, bugin, traktorů nebo nákladních vozů. Nezávislé řízení jednotlivých kol bylo také použito u lunárního roveru, který byl součástí misí Apolla 15, 16 a 17. Nechybí však ani sériově vyráběné vozy s tímto typem řízení. To se ale promítá i do jejich ceny a ve většině případů se osobní automobily s tímto typem řízení neobjevují.

První vozy se 4WS řízením začínaly vznikat ve 30. letech minulého století, kdy celé ovládání fungovalo na mechanickém principu. Postupně se s vývojem elektroniky začaly využívat první mechatronické systémy a ovládání zadních kol se tak postupně vyvinulo z mechanického řízení excentrickým hřídelem, přes elektronicky ovládanou hydrauliku, až po ovládání elektromotory v dnešní době.



.

Obrázek 2.1: Rozdílný poloměr otáčení pro 4WS vozidla - červěně, pro 2WS vozidla - zeleně

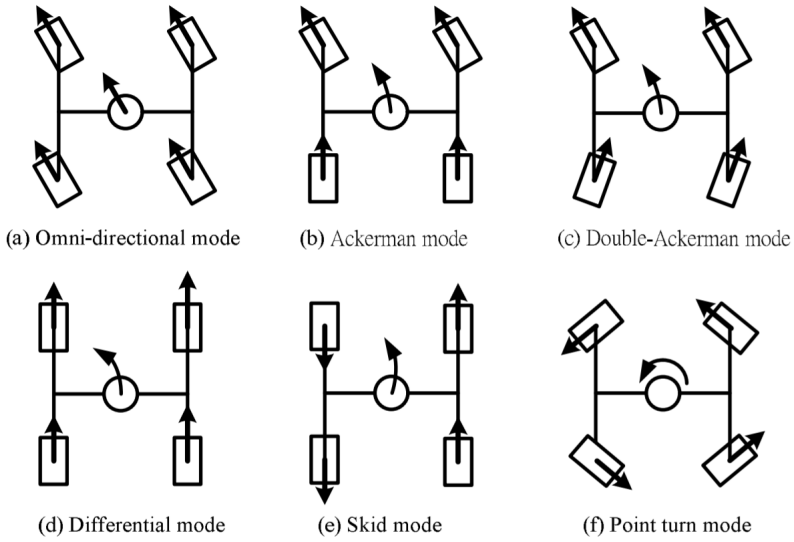
[1] 4 wheel steering

2.1.1 Jízdní vlastnosti vozidel s nezávislým řízením náprav

Nespornou výhodou takového typu řízení je zmenšení poloměru otáčení vozidla. To je patrné z obrázku 2.1. Obrázek byl získán z kinematických modelů 2WS a 4WS vozidel vytvořených v MATLABU[2].

Vozidla byla stejné délky a simulace proběhla při stejném natočení kol a rychlosti. Za pomoci trigonometrie se dá lehce odvodit, že poloměr otáčení 4WS auta bude v takové situaci poloviční, což bylo v simulaci ověřeno.

Zlepšuje se zároveň ovladatelnost při jízdě ve stísněném prostoru. Nezávislé řízení přináší několik „módů“ jízdy. To je způsobeno větší pohyblivostí středu křivosti, který má o jeden stupeň volnosti více, než u klasických aut. Jednotlivé módy jsou znázorněny na obrázku 2.2. Během simulací bude ale vozidlo řízené pouze způsoby (a), (b) a (c).



.

Obrázek 2.2: Různé módy řízení 4WS vozidla. Převzato z [4]

2.1.2 Sériově vyráběné vozy

Výroba 4WS vozidel se rozmohla v 80.letech, kdy se na trh dostává Nissan Skyline R31, Honda Prelude, Mazda Capella/626/MX-6, Mitsubishi Galant VR-4 a další. Přesto, že se tento typ řízení zpočátku netěšil velké oblibě[3], postupně se rozšířilo jeho využití a modely s 4WS řízením můžeme dnes nalézt téměř u každé značky.

Na rozdíl od experimentálních vozidel je u sériově vyráběných vozidel natáčení zadní nápravy omezeno pouze na několik stupnů, obvykle okolo 3° na každou stranu.

Pro nižší rychlosti se kola zadní nápravy natáčí nesouhlasně s koly přední nápravy, tedy mód řízení (c) na obrázku 2.2. Tím se vozidlo začně chovat, jako by byl snížen jeho rozvor[4] a dojde ke změnšení poloměru otáčení [3].

[2]skript je součástí přílohy 01_Kinematicky_model
[3]Hlavně díky své poruchovosti
[4]Vzdálenost přední a zadní nápravy

Při vyšších rychlostech se kola zadní nápravy natáčí souhlasně s předními - aktivní je režim řízení (a). Tím se naopak vůz chová, jako by byl jeho rozvor větší [3] a je zlepšena stabilita například při předjíždění.

2.2 Neholonomní kinematika

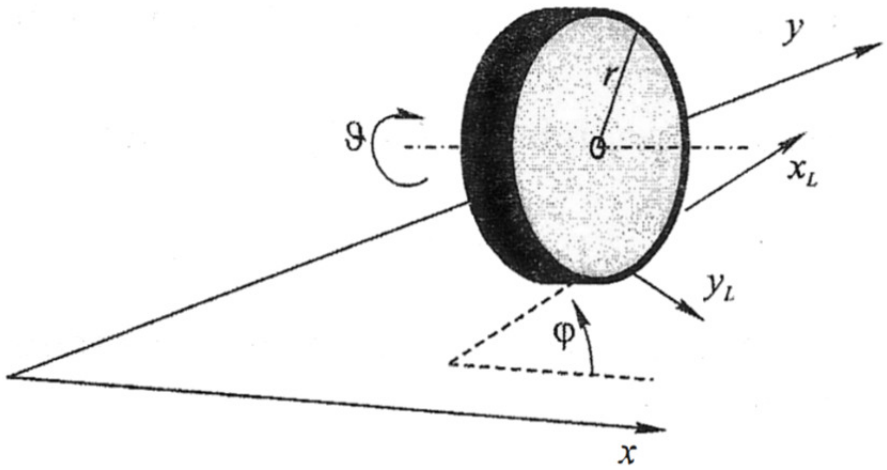
2.2.1 Definice

Dříve, než bude vytvořen kinematický model, je potřeba nejprve zmínit neholonomní kinematiku. Jedním z typů dělení kinematických vazeb je dělení na holonomní a neholonomní. Podle [6] můžeme uvést tyto definice:

- **Holonomní vazby** - pro dané funkce vazeb rychlosti lze najít odpovídající vazbové rovnice pro polohu
- **Neholonomní vazby** - pro dané funkce vazeb rychlosti nelze najít odpovídající vazbové rovnice pro polohu

Typickým případem neholonomních soustav jsou tedy vozidla a případy valení těles, kdy jsou v prostoru pohybově neomezené (mohou se dostat na jakoukoli pozici), ale platí pro ně podmínka nulové rychlosti ve směru jejich lokální osy. Boční pohyb je tedy omezený, ale pohyb dopředu ve směru natočení objektu není.

2.2.2 Příklad valícího se disku



Obrázek 2.3: Schéma valícího se kotouče. Převzato z [6]

Jako příklad uvedu sestavení kinematického modelu pro valící se disk z obrázku 2.3. Kotouč je určen stavovým vektorem $q = [x, y, \varphi, \vartheta]^T$ a je vázaný dvěmi podmínkami rychlosti. A to nulovou rychlostí ve směru lokální osy y_L a valením bez prokluzu v lokální ose x_L, vyjádřenými jako 2.1 a 2.2.

$$\dot{y_L} = 0 \tag{2.1}$$

$$\dot{x_L} = r\dot{\vartheta} \tag{2.2}$$

Transformaci obou podmínek rychlosti do globálního systému provedeme rotační maticí. Základní operace s transformačními maticemi jsou uvedené v jakýchkoli učebních materiálech a skriptech zabývajících se kinematikou (např. [6] nebo [7]). Tím získáme rovnici v maticovém tvaru 2.3, dále rozepsanou na jednotlivé rovnice 2.4 a 2.5, popisující omezení pohybu.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \cdot \begin{bmatrix} \dot{x_L} \\ \dot{y_L} \end{bmatrix} = \begin{bmatrix} r\dot{\vartheta} \cdot \cos\varphi \\ r\dot{\vartheta} \cdot \sin\varphi \end{bmatrix} \tag{2.3}$$

$$\dot{x} - r\dot{\vartheta} \cdot \cos\varphi = 0 \tag{2.4}$$

$$\dot{y} - r\dot{\vartheta} \cdot \sin\varphi = 0 \tag{2.5}$$

Po úpravě rovnic podmínek na tvar 2.6 s neholonomními vazbami w_i získáme z 2.4 a 2.5 maticové tvary jednotlivých vazeb rychlosti 2.7 a 2.8:

$$w_i \cdot q = 0 \tag{2.6}$$

$$w_1 = \begin{bmatrix} 1 & 0 & 0 & -r\cos\varphi \end{bmatrix} \tag{2.7}$$

$$w_2 = \begin{bmatrix} 0 & 1 & 0 & -r\sin\varphi \end{bmatrix} \tag{2.8}$$

Ty můžeme dále napsat jako jedinou matici vazeb: $A(q) = [w_1, w_2]^T$ a upravit 2.6 na:

$$A(q)\dot{q} = 0 \tag{2.9}$$

Vektor $\dot{q}$ se dá následně rozepsat do tvaru:

$$\dot{q} = S(q)u = s_1(q)u_1 + s_2(q)u_2 \tag{2.10}$$

V případě 4WS vozidla byl ve zdroji [5] použit na jeho rozklad matematický software, ale v případě valícího se kotouče lze z rovnic 2.4 a 2.5 odvodit, že hledané vektory $S(q)$ a u budou mít složky podle 2.11. Při rozkladu bylo naším cílem vytvořit vektor u jako vektor vstupních veličin systému.

$$\dot{q} = S(q)u = \begin{bmatrix} r\cos\varphi \\ r\sin\varphi \\ 0 \\ 1 \end{bmatrix} \cdot u_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \cdot u_2 \tag{2.11}$$

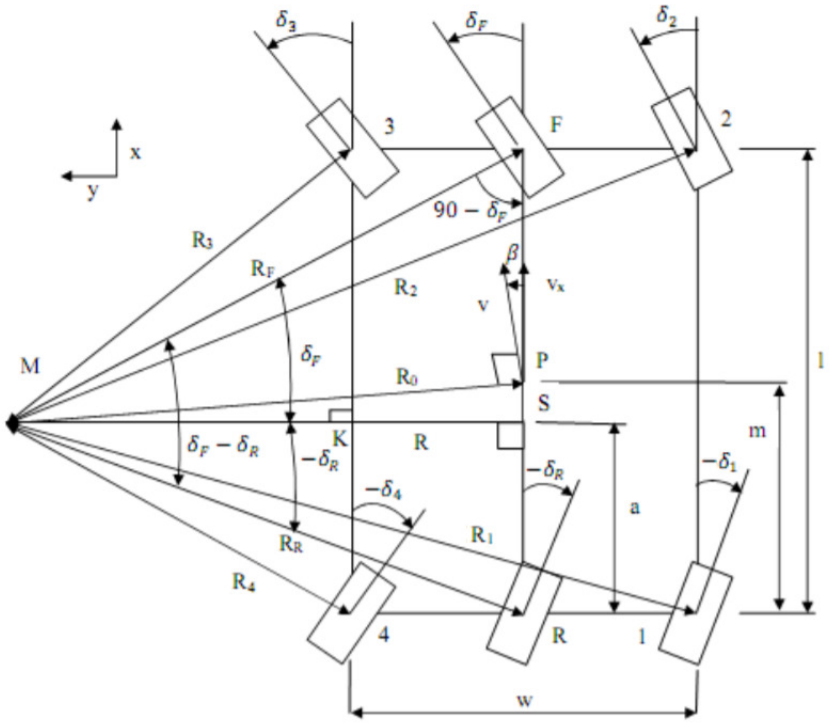
Rovnice 2.11 je tedy už hotový kinematický model valícího se kotouče se vstupy $u_1 = \dot{\vartheta}$ (rychlost otáčení kotouče kolem lokální osy y_L) a $u_2 = \dot{\varphi}$ (rychlost natáčení kotouče kolem lokální osy z_L).

Jelikož je cílem této kapitoly seznámit se s neholonomní kinematikou, je na příkladu valícího se kotouče uveden pouze stručný postup při tvorbě kinematického modelu. Obecný postup při vytváření modelů je popsaný v použité literatuře.

2.2.3 Ackermannova podmínka

Obecně se při projíždění zatáčkou každý bod vozidla pohybuje okolo jediného středu otáčení. Je tak nutné, aby byl úhel natočení u vnějších i vnitřních kol odlišný, jinak by docházelo k jejich smýkání po vozovce.

Ackermannova podmínka tedy říká, že úhel vnitřních kol musí být větší než u vnějších. Toho si můžeme všimnout níže na obrázku 2.4, kde jsou vyznačeny osy kol a jejich jediný průsečík, tedy střed otáčení.



.

Obrázek 2.4: Schéma veličin k výpočtu natočení jednotlivých kol. Převzato z [5]

Toto by byla pravda v případě zanedbávání dynamických účinků. Při zohlednění sil působících na pneumatiky se tento model hodí pouze při nižších rychlostech. Během jízdy vyšší rychlostí bude i přes splnění akcermannovy geometrie docházet ke smýkání pneumatik po vozovce vlivem „smykového úhlu[5]“. Proto se u závodních a jiných rychle jedoucích vozidel využívá anti-ackermannova geometrie, kde jsou úhly natočení vnitřních kol menší a vnějších kol větší. Tím je dosaženo kompenzace dynamických vlivů při vyšších rychlostech.

U 2WS vozidel je výpočet jednotlivých natočení jednodušší, vzhledem k tomu, že se střed otáčení pohybuje pouze po ose zadních kol. Při výpočtu natočení jednotlivých kol 4WS vozidla je zapotřebí více pomocných proměnných[6], přesněji 2.12-2.15. Jednotlivé veličiny použité

[5]Anglicky „slip angle“

[6]Při odvození rovnic byly využity goniometrické vzorce a podrobnější odvození je uvedený opět v [5]

při jejich výpočtech jsou vyznačeny na obrázku 2.4.

$$R_R = l\frac{\sin\frac{\pi}{2} - \delta_F}{\sin\delta_F - \delta_R} \tag{2.12}$$

$$R = R_R \cdot \cos\delta_R \tag{2.13}$$

$$a = R_R \cdot \sin\delta_R \tag{2.14}$$

$$R_0 = \sqrt{R_R^2 + (a+m)^2 - R_R(a+m)\cos\left(\frac{\pi}{2} + \delta_R\right)} \tag{2.15}$$

Samotné natočení jednotlivých kol se vypočítá podle rovnic 2.16-2.19.

$$\delta_1 = arctan\left(\frac{a}{R + \frac{w}{2}}\right) \tag{2.16}$$

$$\delta_2 = arctan\left(\frac{a+l}{R + \frac{w}{2}}\right) \tag{2.17}$$

$$\delta_3 = arctan\left(\frac{a+l}{R - \frac{w}{2}}\right) \tag{2.18}$$

$$\delta_4 = arctan\left(\frac{a}{R - \frac{w}{2}}\right) \tag{2.19}$$

2.3 Plánovače cesty

V této kapitole budou uvedeny čtyři příklady algoritmů, využívané při plánování cesty a v upravené verzi i plánování trajektorie. Podobných algoritmů existuje mnoho, navzájem se můžou lišit použitím rozdílných metrik, způsobem ohodnocení cesty, možnostmi pohybu, nebo jsou kombinací více metod a postupů dohromady. Vzhledem k této rozmanitosti zde nebudou uvedeny pseudokódy, protože jednotlivé verze programů se mohou v některých aspektech odlišovat. Dále uvedené algoritmy jsou nejčastěji používané a v literatuře nejčastěji se vyskytující.

2.3.1 Dijkstrův algoritmus

Mezi nejzákladnější plánovače cesty patří Dijkstrův algoritmus. „Algoritmus byl popsán nizozemským informatikem Edsgerem Dijkstrou v roce 1956 a publikován o tři roky později“[7].

Výstupem algoritmu je nejkratší cesta z vrcholu A k vrcholu B v grafu, případně ohodnocení cest k vrcholům. Výhodou algoritmu je jeho konečnost, kdy je pro konečný vstup do programu vždy dosaženo řešení a nehrozí vznik smyčky, ve které by se zasekl. Popis běhu algoritmu je například ve zdroji [9], nebo v práci [10], kde jsou také uvedeny další informace ohledně teorie grafů a jejich prohledávání. Dále bude uveden pouze stručný popis.

Dijkstrův algoritmus každé hraně grafu - cestě - přiřadí hodnotu ceny. Ta znázorňuje například její délku, případně její schůdnost. Pro algoritmus plánující na pravidelné mřížce je tato hodnota konstantní. Dále se prohledávají sousední body aktuální pozice a jim se přiřazuje hodnota ceny použité cesty tak, že se přičte k její aktuální hodnotě. Cena proto s narůstající délkou cesty roste. Algoritmus hledá vždy bod s nejmenší hodnotu ceny, ten je poté určen jako příští pozice a prohledává se jeho okolí. V případě nalezení vhodnější (s nižší cenou) cesty k již navštívenému bodu je jeho aktuální cena přepsána. Takto algoritmus prohledá graf a ke každému bodu přiřadí hodnotu ceny nejkratší cesty k němu.

[7] Přeloženo z [8]

Existuje více různých úprav a verzí. V případě hledání specifické cesty je zahrnuto značení rodičů navštívených buněk, případně je přidána prioritní množina navštívených vrcholů, ve které se nachází vrcholy seřazené podle jejich ceny a podle tohoto pořadí jsou dále vybírány.

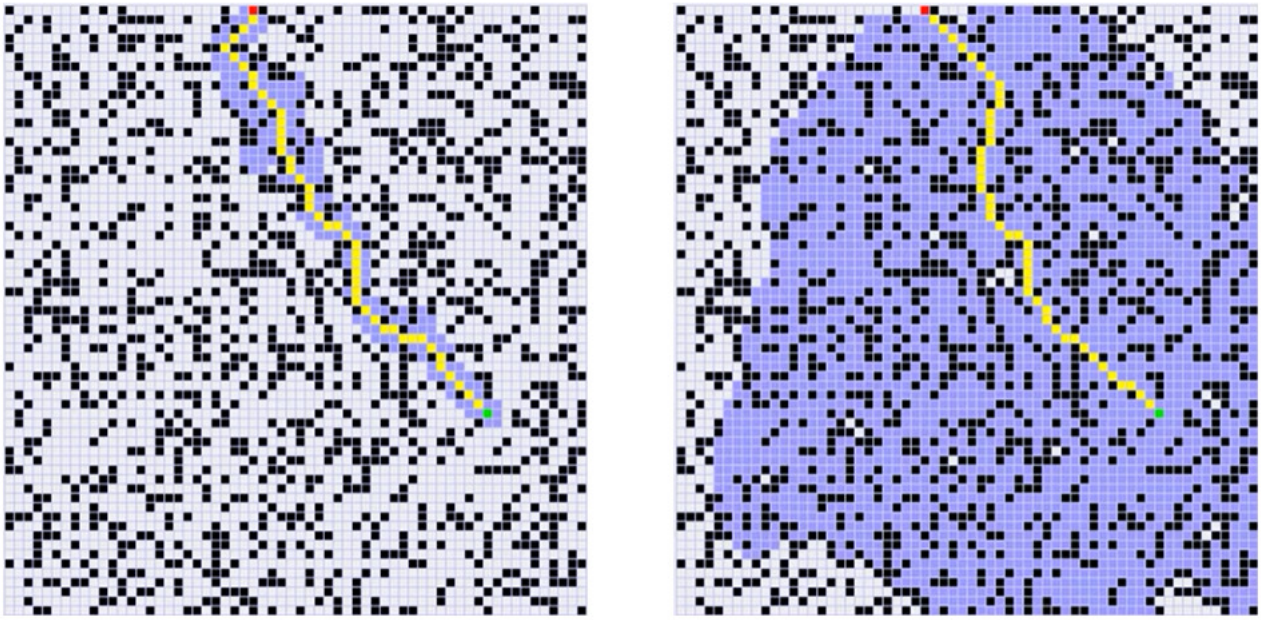
Hlavní problém algoritmu je, že není hodnocen směr prohledávání a navštívené buňky jsou otevírány pouze podle vzálenosti od startovní pozice. Je tedy dosahováno prohlédávání mapy souměrně kolem počátku. Toho lze využít v případech, kdy není směr cesty k cíli a směr k cíli shodný (například bludiště), protože dochází k rovnoměrnému postupu všemi směry. Přesto se algoritmus dodnes používá například u směrovacího protokolu OSPF (Open Shortest Path First).

2.3.2 A* algoritmus

Základ A* tvoří Dijkstrův algoritmus. A* je ale heuristickým algoritmem. To znamená, že je použita heuristika, neboli „odborný odhad“. Touto heuristikou je nově přidaná hodnota ceny, která představuje vzdálenosti vrcholů od cíle, proto dává přibližný odhad směru k cíli. K měření této vzdálenosti se používá euklidovská metrika. Celková cena pro pohyb na grafu je tedy součtem ceny cesty a této vzdálenosti od cíle.

Stručný popis A* algoritmu je ve zdroji [11]. Na rozíl od prohledávání veškerých vrcholů s nejmenší hodnotou ceny - ve všech směrech - je A* schopný systematického prohledávání vrcholů ve směru k cíli díky odhadu vzdálenosti. To dělá algoritmus mnohem efektivnější při hledání cesty v otevřené mapě [8], kdy přibližné sledování směru k cíli zaručuje nalezení cesty.

Pro porovnání je na obrázku 2.5 zobrazena prohledaná část grafu v iteraci, kdy došlo k nalezení cesty. A* algoritmus na levé a Dijsktrův algoritmus na pravé straně. Jak lze vidět, je efektivita A* algoritmu rozhodně lepší a k prohledávání opravdu dochází pouze směrem k cíli.



Obrázek 2.5: Porovnání Dijkstrova a A* algoritmu. Převzato z [12]

[8] Otevřenou mapou je chápána mapa s konvexními útvary, které se dají s určitou aproximací označit za bodové a nehrozí proto uvíznutí v „koutu“

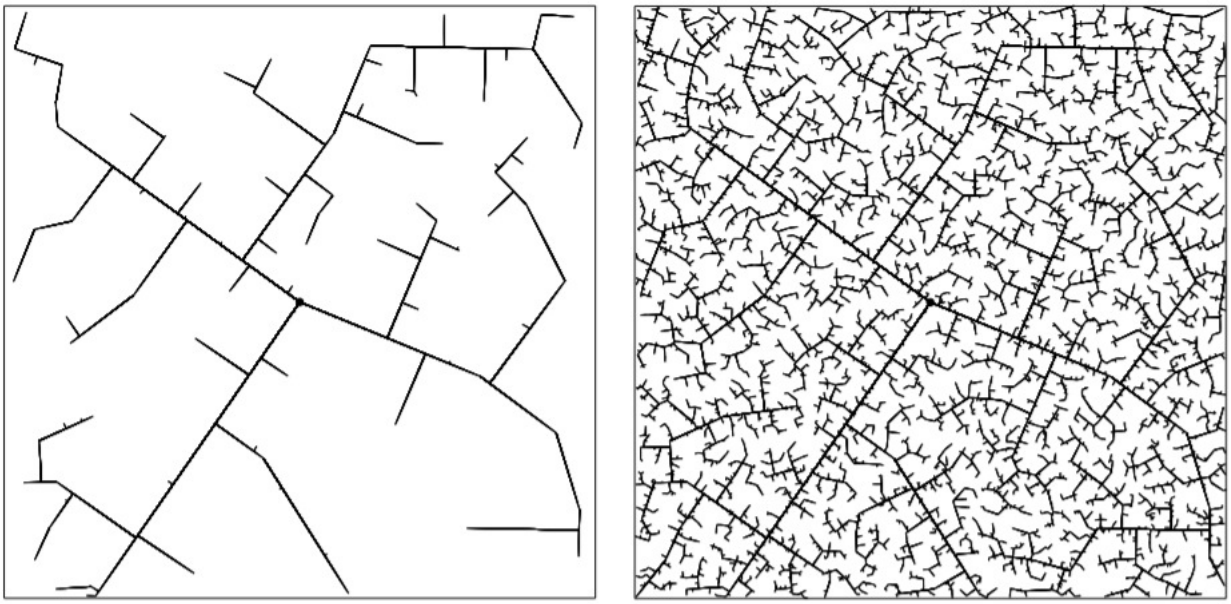
2.3.3 Náhodně rostoucí stromy

RRT (Rapidly exploring Random Trees) je algoritmus vyvinutý dvojicí Steven M. Lavalle a James J. Kuffner Jr. Algoritmus vybere náhodnou konfiguraci v prostoru a najde nejbližší už existující vrchol grafu (při spuštění programu je to pouze startovní pozice). Pokud spojnice obou vrcholů neprotíná překážku, je dvojice propojena novou hranou. Její délka může být omezena na určitou maximální vzdálenost a k propojení proto nemusí při velké vzdálenosti dojít. V opačném případě má nová hrana délku v závisloti na volném protoru před překážkou tak, aby nedošlo ke kolizi.

Po n iteracích dochází k prohledání prostoru a vytvoření grafu. Počet iterací lze nastavit nebo přidat podmínky, jako například nalezení cíle. Nejedná se však o algoritmus hledající nejkratší cestu, pouze náhodně prohledávající prostor. V případě ukončení podmínkou - nalezení cíle - dochází ke zpětnému prohledání rodičů a nelezení cesty. Výsledná cesta není optimální, ale pouze dostačující.

Obdobně jako u předchozích plánovačů existuje několik úprav základního algoritmu. Jedním příkladem je RDT (Rapidly exploring Dense Trees), kde spojení nové náhodné konfigurace probíhá s nejbližším bodem, ale v případě, že se nalezne hrana, blíže než existující vrchol, dojde k propojení přímo s hranou a vzniknou tak dva nové vrcholy ležící na normále k dané hraně grafu. Na obrázku 2.6 je porovnání mapy po určitém počtu iterací pro RDT.

Algoritmus použitý v této práci obsahuje krok navíc, ve kterém je prozkoumáno, jestli se k novému vrcholu grafu nelze dostat kratší cestou z jiného vrcholu předchozích generací. Tím vznikne lehká korekce „odbíhání“ ze žádaného směru a náhodého šíření větví stromu. Tato úprava algoritmu je známa jako RRT*.



Obrázek 2.6: Hustota grafu RDT. Vlevo pro 45 a vpravo pro 2345 iterací. Převzato z [13]

2.3.4 Plánování pomocí potenciálových polí

Jak název napovídá, funguje algoritmus na základě vytvořeného pole. Toto pole silově působí na objekty, které se v něm nachzázejí silou rovnající se záporně vzaté derivaci potenciálové funkce U. Vektor síly tedy můžeme vyjádřit za pomocí gradientu funkce jako 2.20.

$$\vec{F} = -grad(U) \tag{2.20}$$

Tuto sílu pak lze nastavit jako vstup do systému. Tím dosáhneme přímého vytvoření trajektorie. Další možností je využití gradientního sestupu a na diskrétní mapě hledat sousední buňky s nejnižší hodnotou potenciálu, takto postupovat do minima funkce a získat žádanou cestu.

Hlavní rozdělení potenciálové funkce je na přitažlivé a odpudivé pole. Obecně je výsledné potenciálové pole tvořeno dvěma typy polí a existuje mnoho funkcí, které se na jejich výpočet dají využít:

- **Přitažlivé pole** $U_{přit}$ V případě přitažlivého pole se jedná nejčastěji o funkce tvaru eliptického paraboloidu 2.21 [14] a funkce kónického tvaru 2.22 [15] s počátkem v cílovém bodě cesty.

$$U_{přit} = C \cdot \sqrt{\frac{(x-x_g)^2}{a^2} + \frac{(y-y_g)^2}{b^2}} \tag{2.21}$$

$$U_{přit} = C \cdot \left[\frac{(x-x_g)^2}{a^2} + \frac{(y-y_g)^2}{b^2}\right] \tag{2.22}$$

$$\tag{2.23}$$

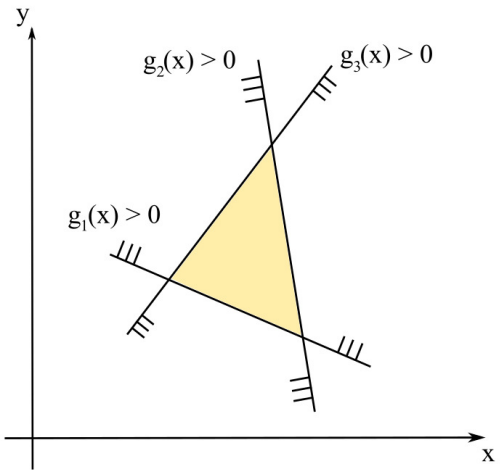
- **Odpudivé pole** $U_{odp,i}$ Odpudivé potenciálové pole je tvořeno nejčastěji funkcí ve tvaru „kopečku“ 2.24 použitém ve zdroji [14]. Ta je díky svému tvaru využívaná převážně pro bodové překážky a předměty. Zdroj [16] uvádí rovnici 2.25 pro obecné překážky. Okraje překážky jsou popsány lineárními rovnicemi a překážka je tak definována soustavou nerovnic ve tvaru $g_i(x) > 0$. Aby byla funkce odpudivého potenciálu definována na celém oboru reálných čísel $\mathbb{R}$, je přidána malá konstanta $\delta > 0$. Soustava nerovnic ohraničující překážku je znázorněna na obrázku 2.7. Této vlastnosti se dá využít a přidat odpudivé pole i pro okraje mapy.

$$U_{odp,i} = C \cdot e^{-\left[(x-x_o)^2+(y-y_o)^2\right]} \tag{2.24}$$

$$U_{odp,i} = \frac{1}{\delta + \sum_i g_i(x) + |g_i(x)|} \tag{2.25}$$

Celkové odpudivé potenciálové pole je získáno jako maximum z jednotlivých potenciálových polí překážek v daném bodě

$$U_{odp}(x,y) = max\{U_{odp,i}(x,y)\} \tag{2.26}$$



Obrázek 2.7: Vytvoření překážky v potenciálovém poli soustavou nerovnic

Jak je uvedeno ve zdroji [16], vznikne tímto způsobem méně lokálních minim potenciálové funkce v prostoru mezi překážkami a lokální minima vzniknou pouze v oblastech hustějšího rozmístění překážek.

- **Celkové pole** U je pak pouhým součtem svých složek

$$U = U_{odp} + U_{přit} \tag{2.27}$$

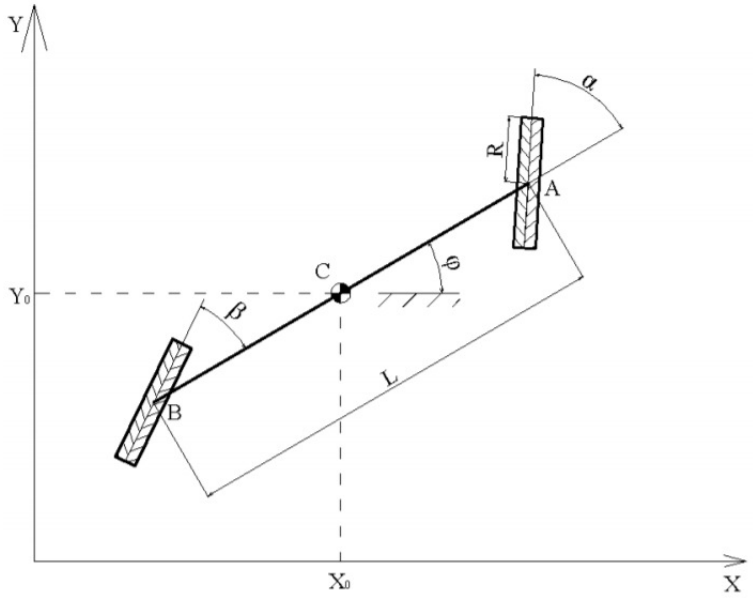
Hlavním problémem tohoto algoritmu je už zmíněný výstkyt lokálních minim potenciálové funkce. V těch dojde k uvíznutí při sestupu ke globálnímu minimu. Řešením mohou být vhodně vytvořené virtuální překážky [15], které upraví potenciálové pole v okolí minima a nedovolí tak uvíznutí v něm.

3 Model vozidla se čtyřmi řízenými koly

Model vozidla byl vytvořen v prostředí MATLAB, kde se zároveň spouštěly veškeré ostatní podprogramy a simulace. Vhodným nástrojem k modelování a simulacím je také nádstavba Simulink. Vzhledem ke složitosti modelu a nutnosti tvořit komplikovanější rovnice bylo vhodnější použít pouze Matlab, který je na tvorbu rovnic jednodušším nástrojem.

3.1 Kinematický model

Ve zdroji [5], ze kterého bylo čerpáno, je uvedeno odvození neholomního kinematického modelu 4WS vozidla [1]. Na obrázku 3.1 můžeme vidět schéma pseudobicyklu se značením a rozměry. Souřadné systémy použité při odvození byly globální systém (x, y) a lokální systém s počátkem v bodě C a pootočením φ.



Obrázek 3.1: Model jednostopého vozidla (převzato z [5])

„Stav pseudobicyklu je jasně určen polohou a orientací bodu C x, y, φ, natočením předního kola α, natočením zadního kola β a pootočením předního hnaného kola ϑ. Soustava má tedy celkem 6 stupňů volnosti.“ [5]

[1] Zde musím upozornit na chybu vyskytující se v odvození modelu, kde byla do vzorců dosazena celá délka vozidla, namísto poloviční. V kinematickém modelu je tedy nutné upravit délku na $\frac{L}{2}$. Dále budu uvádět už opravené vzorce

Stavový vektor můžeme definovat takto: $\mathbf{q} = [x, y, \varphi, \vartheta, \alpha, \beta]^T$ a opravený kinematický model[2] jako:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \\ \dot{\vartheta} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_2 + \begin{bmatrix} s_{31} \\ s_{32} \\ s_{33} \\ 1 \\ 0 \\ 0 \end{bmatrix} u_3 \tag{3.1}$$

$$s_{31} = -R\frac{\sin\varphi\cos\beta\sin\alpha + \sin\varphi\sin\beta\cos\alpha - 2\cos\varphi\cos\beta\cos\alpha}{2\cos\beta} \tag{3.2}$$

$$s_{32} = R\frac{2\sin\varphi\cos\beta\cos\alpha + \cos\varphi\cos\beta\sin\alpha + \cos\varphi\sin\beta\cos\alpha}{2\cos\beta} \tag{3.3}$$

$$s_{33} = R\frac{\cos\beta\sin\alpha - \sin\beta\cos\alpha}{L\cos\beta} \tag{3.4}$$

Kde jednotlivé vstupy u_i symbolizují:

$$u_1 = \dot{\alpha}$$
$$u_2 = \dot{\beta}$$
$$u_3 = \dot{\vartheta}$$

V případě kinematického modelu jsou na vstupu modelu rychlosti pro otáčení a natáčení kol. Ty budou v dalších simulacích výstupem z regulátorů. Jako vstupy do systému, schopné skokově měnit své hodnoty, je tak nemusíme považovat za stavy systému. Stavový vektor se proto může zjednodušit na: $\mathbf{q} = [x, y, \varphi]^T$ a kinematický model 3.1 na:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} s_{31} \\ s_{32} \\ s_{33} \end{bmatrix} u_1 \tag{3.5}$$

Kde vstup u_1 symbolizuje rychlost otáčení kol

$$u_1 = \dot{\vartheta}$$

3.2 Simulace

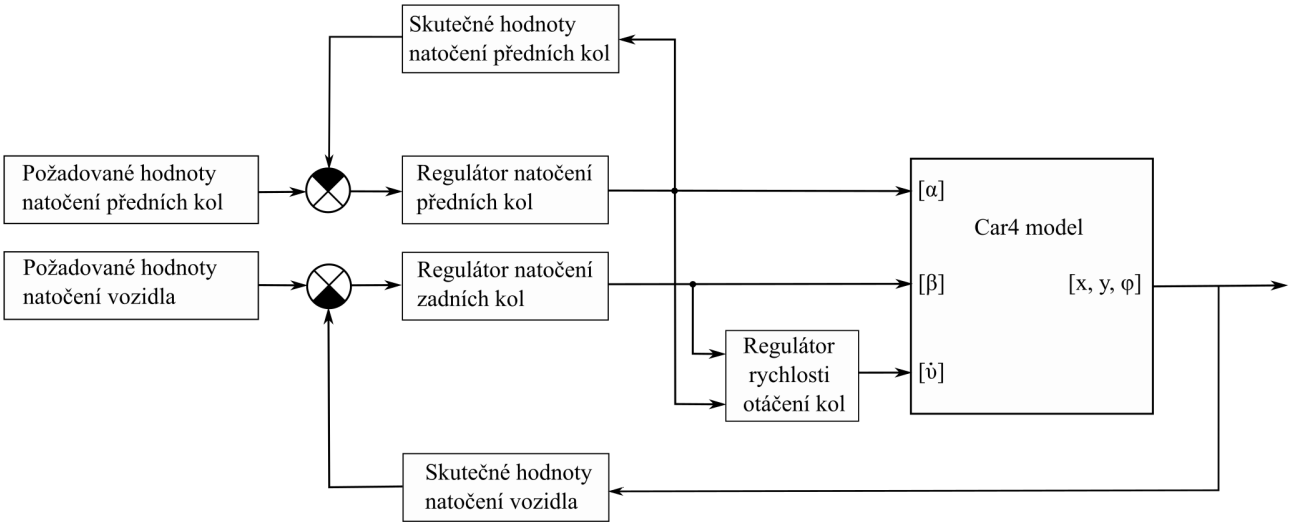
Celý model 4WS vozidla byl v Matlabu uzavřen do MIL (model in the loop) smyčky se zpětnovazebním řízením. Regulátorem byla ovládána rychlost a požadované natočení kol a výstupní hodnoty byly následně vykreslovány.

Blokové schéma zpětnovazební smyčky je znázorněno na obrázku 3.2. Nejprve je získána odchylka skutečné hodnoty od žádané a ta je přivedena do regulátoru, který na základě této odchylky vloží na vstup do systému akční hodnotu, ta dále ovlivní natočení kol.

[2]Samotné odvození modelu je uvedeno podrobně v [5]

Přímým výstupem z modelu vozidla je derivace stavového vektoru $[\dot{x}, \dot{y}, \dot{\varphi}]$, ten je proto nutné nejprve zintegrovat. Pro jednodušší integraci byla v mém případě použita Eulerova metoda s dostatečně malým časovým krokem dT. Pokud by mělo být dosaženo vyšší přesnosti, bylo by vhodné použít například metodu Runge-Kutta [3]. Samotná integrace byla vložena přímo do modelu a výstupem tedy je pouze stavový vektor. Po integraci se skutečné hodnoty porovnávají s žádanými, zároveň je uložen (případně vykreslen) stavový vektor a smyčka se opakuje.

Regulátor rychlosti funguje pouze jako omezený P-regulátor, který řídí hodnotu rychlosti buď podle funkce natočení kol $f(max\{\alpha, \beta\})$ tak, aby byla rychlost při zatáčení nižší, nebo ji nechává konstantní.



Obrázek 3.2: Blokové schéma MIL

3.3 Vykreslení vozidla

K animaci a vykreslení pohybu jsou potřeba základní transformace matic. Vzhledem k dvojrozměrné úloze postačí rotace kolem osy z za pomoci rotační matice a translace přičtením vektoru. Jako příklad je uvedena rovnice 3.6, kde dochází k transformaci lokálního systému do globálního systému rotací o úhel ξ a posunutím o $[T_x, T_y]^T$. Z rovnice se taky dá odvodit postup při inverzní transformaci.

$$\begin{bmatrix} x_g \\ y_g \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \end{bmatrix} + \begin{bmatrix} \cos\xi & \sin\xi \\ -\sin\xi & \cos\xi \end{bmatrix} \cdot \begin{bmatrix} x_l \\ y_l \end{bmatrix} \tag{3.6}$$

Při známém natočení vozidla a všech kol, se provede nejprve rotace jednotlivých kol, jejich translace na pozici přední/zadní nápravy a následná rotace a translace celého vozidla.

Dále jsou vykresleny oskulační kružnice předního a zadního kola pseudobicyklu, které získám určením středu otáčení a výpočtem poloměru. Při odvození vzorců se vycházelo ze směrnicových tvarů přímek os kol 3.7 a 3.8.

[3] Metoda Runge-Kutta čtvrtého řádu v matlabu označována jako řešič ode45

$$y_1 = \tan\left(\alpha + \frac{\pi}{2}\right) x_1 + q_1 \tag{3.7}$$

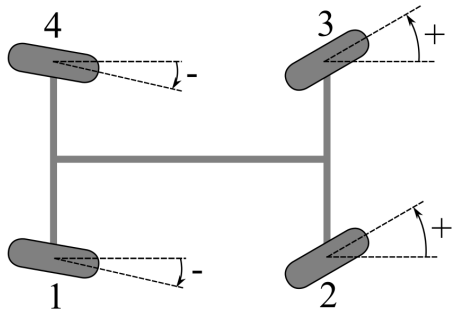
$$y_2 = \tan\left(\beta + \frac{\pi}{2}\right) x_2 + q_2 \tag{3.8}$$

Jde o soustavu n rovnic o n neznámých, po upravách dostaneme rovnice do tvaru 3.9 a získáme tak průsečík 3.10. Případy, kdy je matice **A** singulární, jsou vyřešeny zvlášť pomocí funkcí *if*.

$$\mathbf{Ax} = \mathbf{b} \tag{3.9}$$

$$\begin{bmatrix} P_x \\ P_y \end{bmatrix} = \begin{bmatrix} \tan\left(\alpha + \frac{\pi}{2}\right)\frac{L}{2} \\ -\tan\left(\beta + \frac{\pi}{2}\right)\frac{L}{2} \end{bmatrix} \cdot inv\left(\begin{bmatrix} \tan\left(\alpha + \frac{\pi}{2}\right) & -1 \\ \tan\left(\beta + \frac{\pi}{2}\right) & -1 \end{bmatrix}\right) \tag{3.10}$$

Posledním krokem před samotným vykreslením je převedení jednostopého vozidla na dvoustopé. V kapitole 2.2.3 jsou uvedeny výpočty natočení jednotlivých kol dvoustopého vozidla s uvažováním ackermanovy podmínky. Výchozí natočení vozidla, označení úhlů a jejich znaménka jsou na obrázku 3.3.

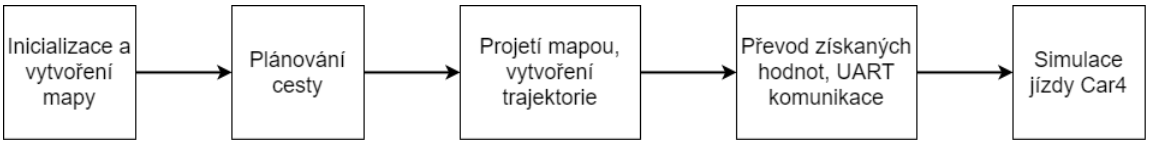


Obrázek 3.3: Číslování kol a indexy úhlů natočení kol

4 Simulační experimenty

Během tvorby bakalářské práce došlo ke změně zadání, a tak se celá práce zabývá pouze vytvořením programu pro naplánování trajektorie. K otestování na skutečném vozidle nedošlo a namísto toho byl vytvořen pouze model přenosu dat a reálného vozidla. Program je tedy rozsáhlý a je proto vhodné jeho části popsat.

Na obrázku 4.1 je celý běh programu stručně zapsán do vývojového diagramu a rozdělen na základní části. V následující kapitole budou jednotlivé celky programu podrobněji vysvětleny a popsány.



Obrázek 4.1: Popis běhu programu

4.1 Vytvoření mapy

Pro různé typy plánovačů cesty se využívají jiné typy map. Zatímco některé se používají na diskrétně rozdělené mapě, jiné využívají mapy spojité. A* a Dijkstrův algoritmus pracují s diskrétní mapou, RRT a metody gradientních sestupů obvykle pracují s mapou spojitou.

Proto se v případě Dijkstrova a A* algoritmu musela mapa převést na mřížku s určitým rozlišením. Mapa je převedena na binární, kde jsou překážky označeny číslem 1 a volný prostor 0. Tato binární mapa se pak posílá do daných funkcí. Zároveň se diskrétní mapa použila i při sestavení potenciálových polí.

4.1.1 Generování překážek

Při generování překážek se podle zadání práce generují překážky různých tvarů. Jedinou podmínkou pro překážky byl jejich konvexní tvar. Nekonvexní překážky je proto nutné rozdělit na více konvexních tvarů.

V nastavení hlavního programu [1] je možnost zapnout generování náhodné mapy. Generování probíhá na základě zadaných parametrů, přesněji jsou to rozměry mapy a počet překážek.

Následně jsou generovány náhodně umístěné pozice překážek, počet stran polygonů společně s natočením a velikostí. Překážky by měly vznikat tak, že před umístěním dojde ke kontrole krytí se startovní a cílovou pozicí. Tím nehrozí vznik chyb při překrytí některé z pozic překážkou.

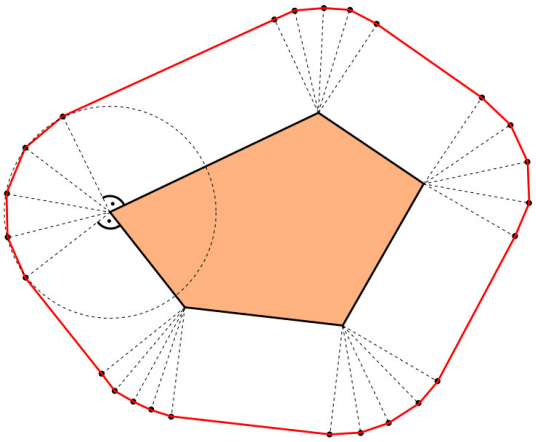
4.1.2 Bezpečná část mapy

Až na metody s potenciálovými poli nedochází u ostatních použitých metod k bezpečnému odstupu od překážek. Algoritmy hledají nejkratší cesty, které už ve své podstatě budou co nejblíže překážkám. Všechny metody považují auto pouze za hmotný bod a jeho velikost nehraje roli.

[1] Ten je součástí příloh práce pod názvem „main.m“

Je tedy potřeba vyřešit odstup od překážek. K tomu se dá využít minkowskeho suma, která slouží ke sčítání dvou množin do výsledné. Pokud se tedy sečte množina vozidla a překážky, dá se okolo překážek vytvořit okraj tvořený množinami vozidla.

Tento způsob by byl složitý, proto byla napsána funkce, která napodobuje transformaci vozidla do hmotného bodu a přičtení jeho velikosti okolo překážek podle zdroje [17]. Tím se zbavíme během výpočtů nutnosti kontrolovat rozměry vozidla a bez ohledu na jeho pozici by nemělo dojít ke kontaktu s překážkou. Velikost auta je tedy aproximována na úhlopříčku obdélníku autu opsaném, tato velikost je poté přičtena jako normála k hranám překážek a z těchto bodů vytvořena nová posunutá hranice. Pro jednotlivé rohy překážky byla hrana zaoblena třemi body ležícími na kružnici se středem v daném rohu. Ukázka nově vytvořené hranice je na obrázku 4.2. Tato funkce funguje pouze pro konvexní tvary překážek. V případě nekonvexních je nutné tyto objekty rozložit na více konvexních.



Obrázek 4.2: Vytvoření okraje překážky s určitou vzdáleností od překážky

4.2 Plánovače cesty

Na rozdíl od předchozí kapitoly 2.3, věnující se rešerší, zde uvedu úpravy už existujících algoritmů a postupy při jejich aplikaci do zbytku programu případně jejich tvorbě.

4.2.1 Dijkstrův a A* algoritmus

Kód algoritmu byl použitý ze zdroje [18] [2]. Algoritmus funguje jako popsaný v kapitole 2.3.2, s tím rozdílem, že pro Dijkstrův algoritmus byla odstraněna hodnota vzdálenosti od cíle. V případě nenalezení cesty byly přidány chybové hlášení a ukončení programu.

Experimentováno bylo také s kombinací potenciálových polí a algoritmu A*, kdy se místo hodnoty vzdálenosti od cíle využila hodnota potenciálu daného bodu. Ve výsledku bylo ale rozhodnuto, že se použijí pouze základní algoritmy.

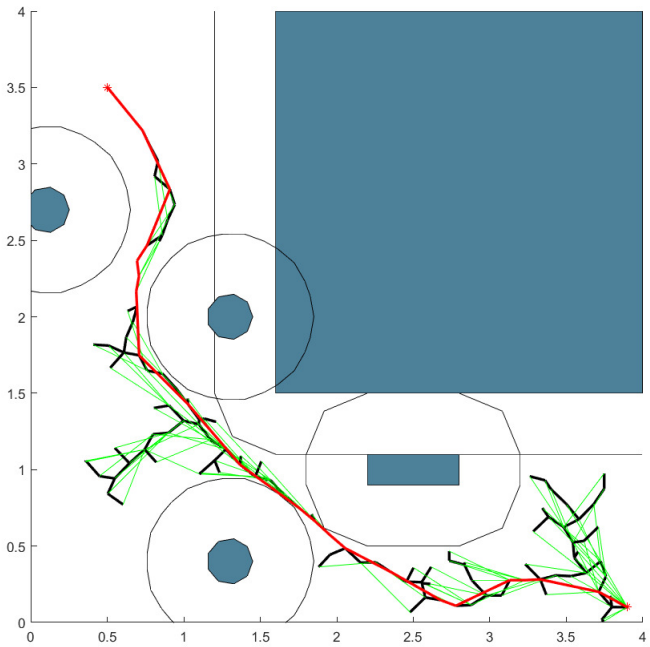
[2] Licence algoritmu je uvedená v přílohách

4.2.2 Náhodně rostoucí stromy

Zdrojový kód algoritmu byl použitý ze zdroje [19] [3], jedná se o RRT*. K propojení náhodné konfigurace v prostoru tedy dochází s nejbližším existujícím uzlem a zpětně dochází k prohledání uzlů a hledání vhodnější cesty. Vytvořená mapa je na obrázku 4.3. Černou barvou jsou znázorněny náhodně generované větve, zelenou pak zpětně vyhledané vhodnější cesty a výsledná cesta je označena červeně.

Vzhledem k povaze náhodně generovaných konfigurací dochází ke stejnému chování jako u Dijkstrova algoritmu a graf prorůstá prostorem rovnoměrně ve všech směrech. Opět zde záleží na způsobu využití algoritmu, kde je v případě otevřené mapy výhodnější postupovat směrem k cíli.

Řešení tohoto nedostatku je zmíněno například v [20] a to rozdělením pravděpodobnossti generování. V mém případě bylo použito normální rozdělení pravděpodobnosti se střední hodnotou v cílovém vrcholu. Směrodatná odchylka byla úměrná vzdálenosti vozidla od cíle, čímž byla v případě otevřené mapy zaručena konvergence k řešení. V případě nekonvexních tvarů překážek, nebo uzkých prostor mezi nimi, však hrozí uvíznutí ve smyčce a je tedy nutné stanovit maximální počet iterací.

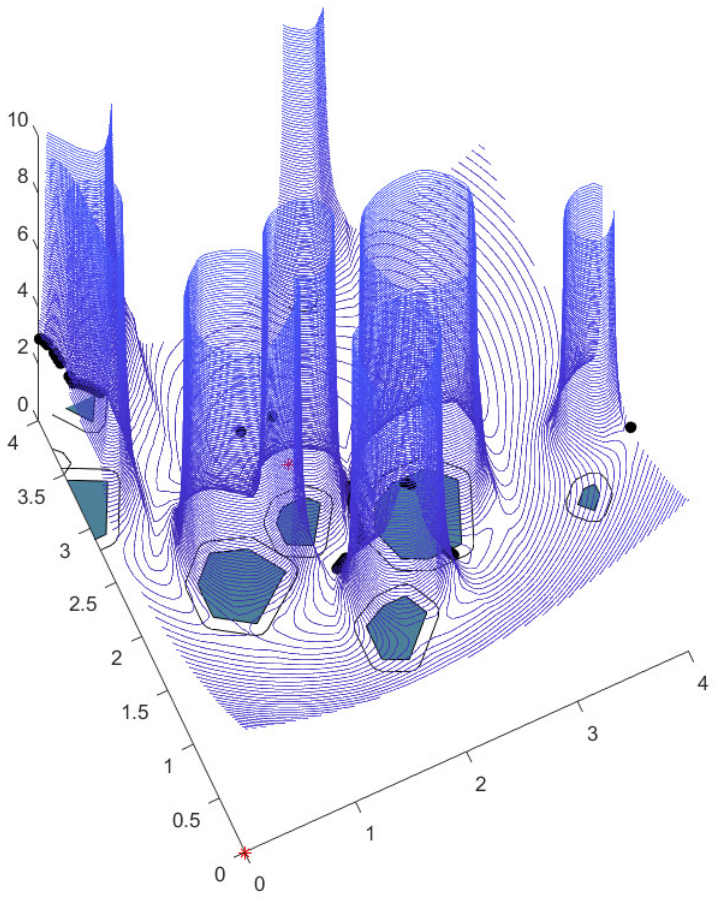


Obrázek 4.3: Větve náhodného stromu vykreslené v MATLABU

4.2.3 Potenciálové pole

Prvním krokem bylo vytvoření potenciálového pole, to bylo vypočítáno z rovnic uvedených v kapitole 2.3.4. Hotové pole je vykreslené na obrázku 4.4, kde jsou vidět také stacionární body, přesněji lokální minima funkce, znázorněna černými body.

[3] Licence algoritmu je uvedená v přílohách



Obrázek 4.4: Potenciálové pole vykreslené v MATLABU

Samotné hledání cesty na mapě pak probíhá metodou gradientního sestupu. Jsou prohledávány hodnoty potenciálu v bodech okolo aktuální pozice a po nalezení nejnižší hodnoty se do této buňky algoritmus posune.

4.3 Sledování cesty

Na rozdíl od ostatních metod je v této práci využito sledování cesty jako zdroj trajektorie. Při sledování cesty se využije kinematického modelu a regulátorů, které budou navádět vozidlo k cíli. Během toho se v každém kroku budou ukládat dosažené hodnoty natočení kol a rychlosti v závislosti na ujeté vzdálenosti. Takto vytvořená trajektorie bude tedy plně respektovat všechny kinematické omezení vozidla.

Pro zjednodušení postupu je výhodnější ukládat hodnoty v závislosti na ujeté vzdálenosti, která byla vypočítána z rychlosti otáčení kol v každém kroku jako:

$$s_{n+1} = s_n + \dot{\vartheta} \cdot R \cdot dT \tag{4.1}$$

Tím je zaručeno, že rychlost skutečného auta bude nezávislá na datech ze simulace a nebude tedy záležet na její hodnotě. Během testování se ale vlivem kroku dT ukázalo vhodnější nepřesahovat rychlost vozidla nastavenou během vytváření trajektorie.

Pokud bude mít vozidlo v simulaci časový krok dT, model bude dostatečně přesný a rychlosti natáčení kol během simulace dostatečně pomalé, budou hodnoty převedené do reálného auta s dosatečně rychlým natáčením kol a nižší rychlostí dosaženy podstatně přesnější a opět nebude nutné výrazně zasahovat do jeho jízdy regulátory.

4.3.1 Rychlost vozidla

Co se regulace rychlosti týče, byl navržen pouze jednoduchý regulátor určující rychlost na základě natočení kol. Pro zlepšení dynamických vlastností vozidla je v zatáčkách rychlost snížena a na rovných úsecích opět zvýšena podle vytvořené funkce $f(max\{\alpha, \beta\})$. Graf funkce je vykreslený na obrázku 4.5.

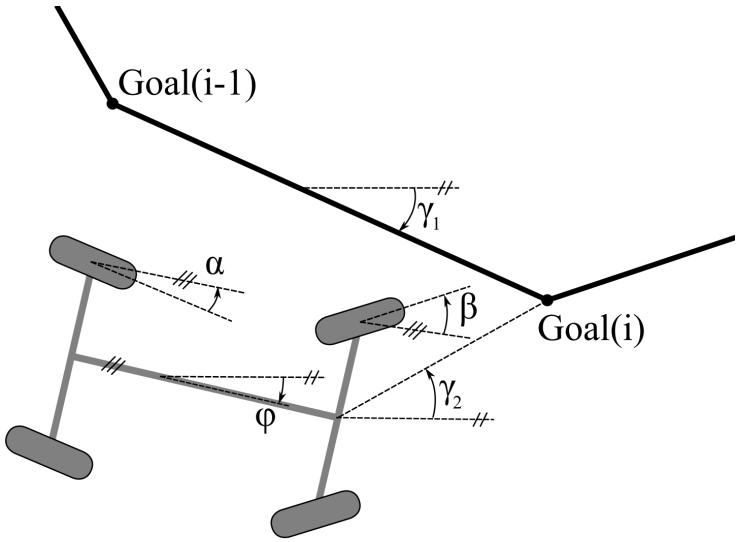
Obrázek 4.5: Závislost rychlosti otáčení kol na jejich natočení

4.3.2 Natočení kol

K demonstraci možností 4WS vozidla bylo rozhodnuto o vytvoření dvou nezávislých P-regulátorů. Rozdělení je na přední kola, navádějící vozidlo k cíli, a zadní kola, starající se o natočení celého vozidla rovnoběžně s daným úsekem cesty. Na obrázku 4.6 jsou znázorněny hodnoty nutné pro regulaci.

Akční veličina pro servopohony přední nápravy je získána z odchylky úhlů γ_1 a $(\beta + \varphi)$ znázorněných na obrázku 4.6. Kde úhel γ_1 vypočítáme ze směrnice úsečky aktuálního cíle a středu přední nápravy.

Jak již bylo zmíněno v předchozím odstavci, regulátor zadních kol se snaží srovnat auto s aktuálním úsekem cesty, tedy úsekem mezi $Goal(i-1)$ a $Goal(i)$. Vstupem do regulátoru je odchylka úhlu natočení auta φ a úseku cesty γ_2.



Obrázek 4.6: Schéma regulovaných a žádaných hodnot natočení

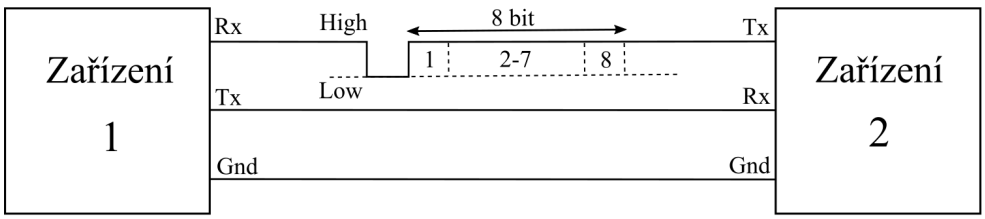
4.4 Simulace převodu dat a jízdy vozidla

Namísto odesílání dat přímo do vozidla Car4 bude vzhledem k nedostupnosti laboratoří přenos dat pouze simulovaný a hodnoty budou opět vloženy do kinematického modelu.

Komunikace s autem probíhá pomocí protokolu UART. Jde o způsob synchronní komunikace. Obě komunikující zařízení musí mít tedy nastavenou stejnou frekvenci vysílání a příjmu dat. Přenos funguje na 3 linkách, Rx - posílání data, Tx - příjem dat, a Gnd - zem.

Před samotným odesláním je převeden vektor žádaných hodnot na rozlišení mikrokontroleru. Pro natočení serv se hodnoty pobyhovaly v rozmezí $< 1300, 2300 >$ a pro rychlost $< 0, 3997 >$[4]. Veličiny jsou dále převedeny na šestnáctibitové hodnoty a ty rozděleny na dvě osmibitové, seřazené od nejvýznamějšího bitu po nejméně významný.

Proces odesílání jednotlivých 8bit hodnot začíná na výchozí hodnotě High. Pro zahájení přenosu klesne hodnota na Low. Následují jednotlivé bity čísla a poté je nastavena opět hodnota High, symbolizující konec vysílání. Poté program čeká na potvrzení o příjmu dat a jakmile dojde k potvrzení příjmu, celý proces se opakuje. Průběh odeslání osmibitového čísla je na obrázku 4.7.



Obrázek 4.7: Jednoduchá komunikace protokolem UART

[4]Hodnoty v simulaci ale nerespektují tyto rozsahy. Pro natočení je zde použit rozsah 0-1000

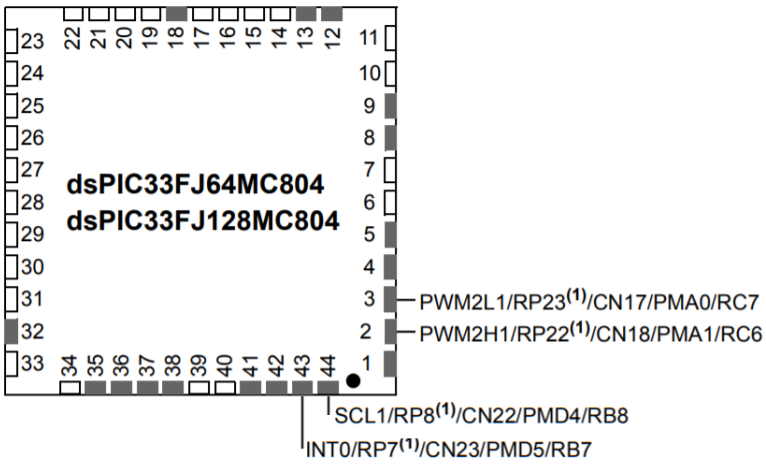
Odesílá se tak nejprve prvních osm nejvýznamějších bitů čísla, následně zbylých osm. Z nich je po přijetí vytvořena celá hodnota a uložena do paměti. Jakmile dojde k odeslání veškerých hodnot, jsou převedy zpět z rozlišení mikrokontroleru na fyzikální veličiny.

Poslední částí simulace je projetí vozidla mapou k cíli. Je k tomu využit upravený kinematický model s rychlejšími regulátory serv a menším simulačním krokem. Během jízdy se počítá ujetá vzdálenost z pootočení kol a na jejím základě se určí vhodné natočení kol z vektoru přijatých dat. Postupně se tedy hodnoty z vektoru posílají na serva vozidla a při posledním prvku je simulace ukončena.

5 Práce na Car4

Na úvod by bylo vhodné napsat základní informace o hardwaru vozidla a uvést aktuální informace. Samotné experimentální vozidlo je vyfoceno na obrázku 5.2. Software funguje na 16bit mikroprocesoru od firmy Microchip [1]. Vozidlo má hnaná všechna čtyři kola [2] a natáčení kol je realizováno za pomocí čtyř servopohonů [3].

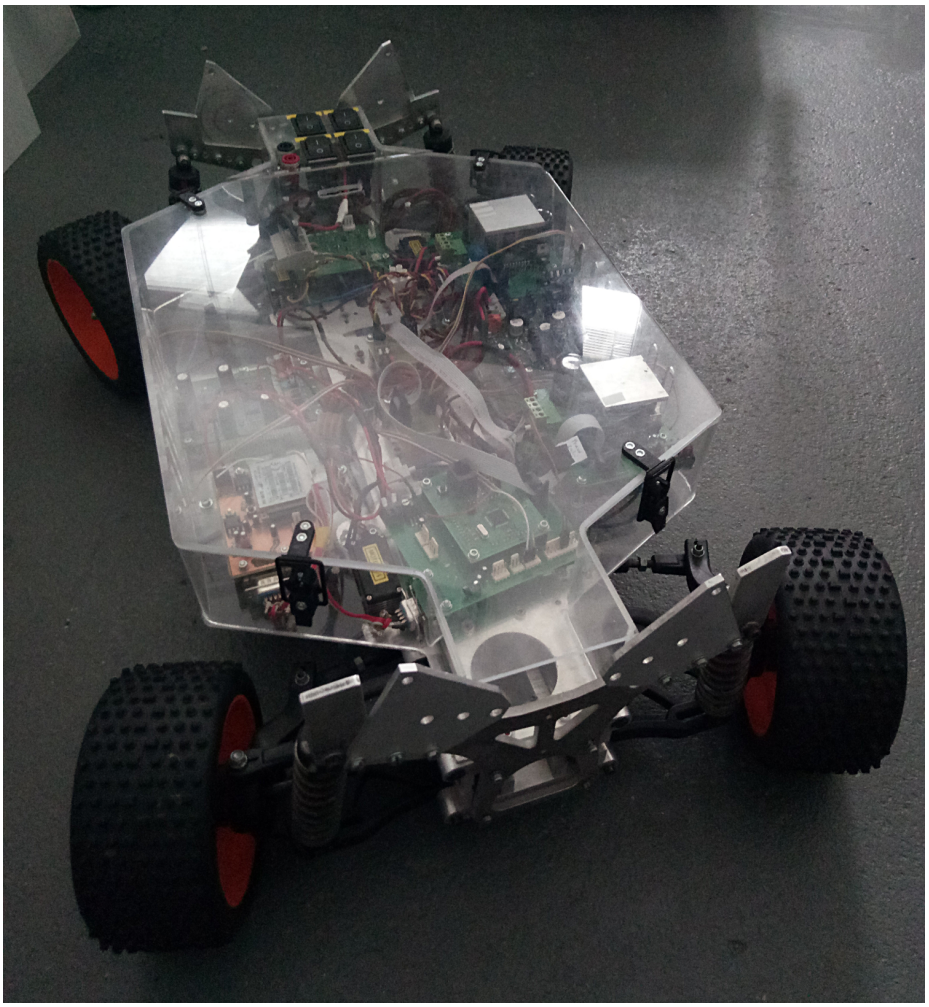
Pokud jde o úpravy hardwaru, projekt doposud pracoval jako obyčejné 2WS vozidlo. Zprovozněn byl tedy pohon a natáčení předních servopohonů. Jediným úkolem nutným k plné funkčnosti tedy bylo zprovoznění natáčení i zadních kol. K tomu bylo potřeba zmapovat využité piny a zjístit, který z nich funguje jako PWM výstup z procesoru. Vzhledem k obsazení většiny vhodných výstupů byly vyměněny piny sloužící ke komunikaci přes UART a na tyto uvolněné piny (2 a 3) zavedeny zadní servopohony. Komunikace se poté přepojila na jiné digitální piny (43 a 44). Poslední verze modelu v SIMULINKU je součástí příloh práce.



Obrázek 5.1: Jednotlivé piny mikrokontroleru. Upraveno z [21]

Vzhledem k uzavření laboratoří nebyly změny v softwaru vozidla dokončeny. Změny se týkaly hlavně způsobu výpočtu natočení kol z enkodérů. Doposud byly z enkodérů počítány rychlosti kol. Ty ale v programu nebyly potřeba a algoritmus navíc příliš zatěžoval řídící jednotku, která nezvládala rychlosti počítat. Důležité bylo tedy pouze odečítání natočení kol.

[1] Přesněji dsPIC33FJ128MC804
[2] jde o čtyři motory PD4266-24-14-BFEC
[3] Hitec HS-M7990TH



Obrázek 5.2: Experimentální vozidlo Car4 s nezávisle řízenými koly

6 Závěr

V první praktické části práce byl úspěšně vytvořen kinematický model vozidla s nezávisle řízenými koly. Jako dostatečné přiblížení byl modelován pouze pseudobicykl, který se dá jednoduše převést na model dvoustopého vozidla pomocí přepočtů úhlů natočení kol. Model byl vyzkoušen v simulaci, kde se na vstupy zadávaly náhodné rychlosti natáčení kol pri konstantní dopředné rychlosti. Správnost modelu byla taktéž ověřena vykreslováním oskulačních kružnic a os kol, které by se měly protínat v jediném středu otáčení. Takto zhotovený model bylo tedy možné použít pro další úlohy.

Dále byla vytvořena mapa s náhodnými překážkami různého tvaru a na ní úspěšně vyzkoušeny čtyři algoritmy pro hledání cesty. Tyto algoritmy byly implementovány buďto jako už vytvořené, upravené pro vhodnější použití, nebo byly vytvořeny vlastní.

Za pomoci kinematického modelu byla metodou sledování cesty získána trajektorie v závislosti na ujeté dráze. Toho se dosáhlo využitím dvou nezávislých regulátorů pro zadní i přední kola. Ty naváděly vozidlo k cíli. Regulátory se navrhly tak, aby bylo využito nezávislého natáčení kol a mohly být vidět jízdní možnosti 4WS vozidla.

V poslední části se provedl náhradní experiment s upraveným kinematickým modelem a byla simulována jízda vozidla Car4 s nahranou trajektorií z předchozího bodu. Pro větší realističnost simulace se do vektoru trajektorie zavedla chyba. Z originálního zadání bylo v praktické části práce splněno pouze zprovoznění nezávislého řízení všech kol. Demostrační experiment a nahrávání trajektorie do vozidla nebylo před uzavřením laboratoří stihnuto.

K celkovému hodnocení výsledků se musí dodat, že použitá metoda sledování trajektorie je sice mnohdy nepředvídatelná, na druhou stranu ale zaručuje vytvoření trajektorie s uvažováním veškerých kinematických omezení modelu. Při vytvoření okraje okolo překážek a použití vhodných regulátorů je dosaženo uspokojivé přesnosti sledování cesty a doražení do cíle.

Uložení trajektorie v závislosti na ujeté dráze způsobí nezávislost na rychlosti vozidla. Pokud se v simulaci navíc zpomalí rychlost servopohonů a nebude docházet ke skokovým změnám natáčení kol, omezí se i nepřesnost natočení kol u reálného vozidla. Nelze tedy o metodě sledování trajektorie hovořit jako o nevhodné, ale pouze jako o zajímavé alternativě k metodám optimalizačním.

Nejvhodnějším algoritmem je podle mého názoru A* algoritmus. Co se týče času řešení i úspěšnosti nalezení cesty k cíli, jeví se jako nejefektivnější. Na rozdíl od RRT se nejedná o náhodné konfigurace v prostoru, také nemůže dojít k uvíznutí algoritmu jako u metody potenciálových polí a šíření optimální cesty je oproti Dijkstrova algoritmu směrováno pouze k cíli. Jeho výhodou je i možnost rozšíření a implementace přímo segmentů trajektorie během plánování cesty.

Návázání na práci vidím v úspěšném implementování algoritmu do vozidla Car4 a také ve vylepšení algoritmů ve fázi dojetí k cíli. Zde je možnost zavedení požadovaných hodnot stavů vozidla v cílovém bodě. Momentálně se algoritmus bez ohledu na natočení zastaví v postačující vzdálenosti od cíle. V kombinaci s parkovacím asistentem, který byl vytvořen a otestován v předchozích pracech, by se dalo přepínat mezi jednotlivými fázemi cesty - průjezd mapou a zaparkování - a vytvořit tak dokonalejší celek. Další prací by mělo být také dokončení úprav softwaru, odečítání hodnot natočení kol a počítání ujeté vzdálenosti.

Seznam zkratek a symbolů

4WS 4 Wheel Steering

2WS 2 Wheel Steering

OSPF Open Shortest Path First

RRT Rapidly exploring random trees

RDT Rapidly exploring dense trees

MIL Model in a loop

List of Abbreviations

4WS 4 Wheel Steering

2WS 2 Wheel Steering

OSPF Open Shortest Path First

RRT Rapidly exploring random trees

RDT Rapidly exploring dense trees

MIL Model in a loop

Seznam obrázků

Literatura

[1] Wikipedia contributors. *Amazon Prime Air* [online]. 2020-5-1 [cit. 2020-5-2]. Dostupné z: https://en.wikipedia.org/wiki/Amazon_Prime_Air

[2] SEVERSON, Aaron. Four-wheel steering demystified. In: *autoweek.com* [online]. 2015-6-12 [cit. 2020-5-2]. Dostupné z: https://www.autoweek.com/car-life/a1871191/four-wheel-steering-demystified/

[3] BARNETT, Josh. Technology explained: rear-axle steering. In: *total911.com* [online]. 2014-10-7 [cit. 2020-5-2]. Dostupné z: https://www.total911.com/technology-explained-rear-axle-steering/

[4] LIN, Chih-Jui, et al.: Design and Implementation of 4WS4WD Mobile Robot and Its Control Applications. In:IEEE[online]. 2013[cit. 2016-05-10]. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6614666

[5] JASANSKÝ, M. *Návrh dynamických modelů pro řízení trakce experimentálního vozidla.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010. 123 s. Vedoucí diplomové práce Ing.Robert Grepl, Ph.D.

[6] GREPL, Robert. Kinematika a dynamika mechatronických systémů. Brno: Akademické nakladatelství CERM, 2007, 158 s. : il. ; 26 cm. ISBN 978-80-214-3530-8.

[7] PŘIKRYL, Karel. Kinematika. Vyd. 5., V Akademickém nakladatelství CERM 3. vyd. Brno: Akademické nakladatelství CERM, 2008, 142 s. : il. ISBN 978-80-214-3679-4.

[8] Wikipedia contributors. *Dijkstra's algorithm* [online]. 2020-6-6 [cit. 2020-6-8]. Dostupné z: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

[9] Dijkstra's Algorithm - Computerphile. In: *Youtube* [online]. 2017-2-15 [vid. 2020-4-19]. Dostupné z: https://www.youtube.com/watch?v=ySN5Wnu88nE&t=10s

[10] HAMERNÍK, M. Grafy a algoritmy pro hledání nejkratších cest. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2009. 47 s. Vedoucí bakalářské práce Mgr. Martina Bobalová, Ph.D.

[11] A* (A Star) Search Algorithm - Computerphile. In: *Youtube* [online]. 2017-1-4 [vid. 2020-4-19]. Dostupné z: https://www.youtube.com/watch?v=GazC3A4OQTE

[12] Compare A* with Dijkstra algorithm. In: *Youtube* [online]. 2015-3-1 [vid. 2020-5-2]. Dostupné z: https://www.youtube.com/watch?v=g024lzsknDo

[13] LAVALLE, Steven M. *Planning algorithms* [online]. Cambridge: Cambridge university press, 2006, 842p, poslední úprava: 2012-4-20 [cit. 2020-4-15]. Dostupné z: http://planning.cs.uiuc.edu/bookbig.pdf

[14] Autonomous Path Planning. In: *Youtube* [online]. 2017-3-4 [vid. 2020-5-2]. Dostupné z: https://www.youtube.com/watch?v=fNBrIngCJp8&t=1517s

[15] SIDDIQUI Rymsha. Path Planning Using Potential Field Algorithm. In: *medium.com* [online]. 2018-7-28 [cit. 2020-5-2]. Dostupné z: https://medium.com/@rymshasiddiqui/path-planning-using-potential-field-algorithm-a30ad12bdb08

[16] HWANG, Yong Koo and AHUJA Narendra. A potential field approach to path planning, In: *IEEE Trans. Robotics and Automation 8* [online]. 1992, P: 23-32. [cit 2020-4-15]. Dostupné z: https://www.semanticscholar.org/paper/A-potential-field-approach-to-path-planning-Hwang-Ahuja/1cf496e7db9712bfd6f9373b161cbc359031f568

[17] Minkowski Sum of Polygons. In: *Youtube* [online]. 2014-3-2 [vid. 2020-5-2]. Dostupné z: https://www.youtube.com/watch?v=3qMNuoTHNHs

[18] UELAND, Einar. Astar-Algorithm. In: *mathworks.com* [online]. 2019-11-3 [cit. 2020-6-8]. Dostupné z: https://ch.mathworks.com/matlabcentral/fileexchange/56877-a-astar-a-star-search-algorithm-easy-to-use

[19] VEMPRALA, Sai. 2D/3D RRT* algorithm. In: *mathworks.com* [online]. 2017-1-5 [cit. 2020-6-8]. Dostupné z: https://www.mathworks.com/matlabcentral/fileexchange/60993-2d-3d-rrt-algorithm

[20] QU Jun. Nonholonomic Mobile Robot Motion Planning. In: *lavalle.pl/index.html* [online]. [cit. 2020-5-2]. Dostupné z: http://msl.cs.uiuc.edu/~lavalle/cs576_1999/projects/junqu/

[21] Microchip Technology Inc. dsPIC33FJ128MC804. In: *microchip.com* [online datasheet]. [cit. 2020-5-2]. Dostupné z: https://www.microchip.com/wwwproducts/en/dsPIC33FJ128MC804

Seznam příloh

- 01_kinematicky_model - Kinematický model 4WS vozidla, obsahuje simulaci k porovnání poloměru otáčení.
- 02_planovani_trajektorie - Hlavní simulace.
- 03_Funkcnost_servopohonu - Ukázka zprovoznění všech 4 servopohonů vozidla Car4.
- 04_Ukazka_jizdy - Videoukázka simulované jízdy.
- 05_Ukazka_pot_pole - Videoukázka sestupu potenciálovým polem.
- 06_Ukazka_Astar - Videoukázka šíření algoritmu A*.
- 07_Ukazka_RRT - Videoukázka šíření algoritmu RRT.