VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

VÝPOČETNÍ MODEL ŘÍZENÍ A CHOVÁNÍ SAMOČINNĚ ŘÍZENÉHO VOZIDLA

COMPUTATIONAL MODEL OF CONTROL AND BEHAVIOUR OF SELF-DRIVEN VEHICLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MICHAL KUŽELA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JOSEF STRNADEL, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Kužela Michal**
Program: Informační technologie
Název: **Výpočetní model řízení a chování samočinně řízeného vozidla**
Computational Model of Control and Behaviour of Self-Driven Vehicle
Kategorie: Modelování a simulace

Zadání:

1. Zdokumentujte požadavky, problémy, pojmy a principy související s činností samočinně řízeného vozidla. Navrhněte vhodnou abstrakci samočinně řízeného vozidla, jeho okolí a aspektů klíčových z hlediska jeho řízení a chování.
2. Proveďte rešerši v oblasti prostředků výpočetního modelování systémů a analýzy jejich vlastností a zvolte prostředky vhodné k řešení zadané práce.
3. Pomocí zvolených prostředků vytvořte výpočetní model řízení a chování samočinně řízeného vozidla a výpočetní model jeho okolí s cílem analyzovat vliv způsobu řízení vozidla na chování vozidla v daných podmínkách.
4. Vlastnosti modelu vhodně ověřte.
5. Diskutujte a zhodnoťte možnosti vytvořeného modelu z hlediska analýzy vlivů zmíněných v bodu 3 a navrhněte možné směry pokračování v projektu.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 zadání, vytvoření základního výpočetního modelu samočinně řízeného vozidla.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Strnadel Josef, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 9. května 2019

Abstrakt

Cílem této práce je zdokumentovat informace týkající se autonomních vozidel a na jejich základě vytvořit a implementovat model autonomního vozidla. Následně provést analýzu chování tohoto modelu za různých podmínek. Zvolený problém jsem řešil pomocí jazyku JavaScript, frameworku Electron a Node.JS. Model se skládá ze dvou částí. První část modelu je samotné vozidlo reprezentované především fyzikálními vzorci. Druhá část modelu jsou senzory detekující překážky za pomoci algoritmu založeného na průsečíku přímek. Okolí modelovaného systému vozidla má stochastické parametry. Vytvořené řešení umožňuje simulovat chování modelu. Probíhající simulaci je možné vizualizovat a také analyzovat výsledky za užití pomocného nástroje vytvořeného k tomuto účelu. Na základě získaných dat lze porovnat chování modelu za různého počasí a stanovených rychlostních omezení.

Abstract

This thesis explains basic principles of autonomous vehicles. Based on these principles, it describes proposed simulation model of autonomous vehicle with its behavior in different outer conditions. The model is composed from sensors and vehicle itself. Sensors use an algorithm based on line intersection to detect collisions. Vehicle model is mostly described by physical formulas and custom decision algorithms. Its surrounding objects have parameters that are randomly selected for each simulation run. This thesis also explains implementation of the simulation model with its own simulation framework and an analysis tool for collected data. All of these features are implemented in JavaScript with use of Node.js and an Electron framework.

Klíčová slova

simulace, modelování, JavaScript, Node.JS, Electron, autonomní vozidlo, samočinně řízené vozidlo, analýza dat

Keywords

simulation, modeling, JavaScript, Node.JS, Electron, autonomous vehicle, self driven vehicle, data analysis

Citace

KUŽELA, Michal. *Výpočetní model řízení a chování samočinně řízeného vozidla.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Josef Strnadel, Ph.D.

Výpočetní model řízení a chování samočinně řízeného vozidla

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Josefa Strnadela, Ph.D.. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Kužela
13. května 2019

Poděkování

Rád bych poděkoval panu Ing. Josefu Strnadelovi, Ph.D za cenné rady, které mi poskytoval v průběhu řešení této bakalářské práce. Taktéž děkuji studentce Simoně Dlouhé za podporu a matematické konzultace.

Obsah

Kapitola 1

Úvod

Tato práce se zabývá tématem simulace chování samočinně řízeného (autonomního) vozidla. Popisuje postup návrhu modelu využitého pro simulaci, který vychází z poznatků získaných studiem aktuálních trendů v této oblasti a analýzou technických aspektů vozidel včetně jejich senzorů, které vozidlu umožňují „vidět“ své okolí. Zabývá se také vytvořením simulačního prostředí a nástroje pro následné zkoumání informací získaných z jednotlivých běhů simulace.

Autonomní vozidla jsou dnes často diskutovaným tématem. Spousta automobilových výrobců se snaží dosáhnout vyšší úrovně automatizace řízení. Již delší dobu je běžnou praxí, že jsou vozidla vybavena částečnou automatizací, pro příklad uvedu adaptivní tempomat, varování při změně jízdních pruhů nebo také nouzové brzdění. Některé automobilky ale cílí na vyšší stupeň automatizace, a to takovou, že vozidla již nebudou vyžadovat přítomnost řidiče. Uvedení takových vozidel do provozu je momentálně nepravděpodobné, zákony to ve většině částí světa nepovolují. To ovšem nebrání výzkumu v této oblasti, a tak společnosti jako Tesla a Waymo pokračují ve zdokonalování svých autopilotů. Společnost Waymo získala roku 2018 dokonce povolení pro testování jejich autonomní taxislužby v provozu v Kalifornii.

Testování vozidel v reálném provozu je důležitou součástí vývoje, takové testování je ale časově i finančně náročné a také může být nebezpečné pro ostatní účastníky provozu. Proto je vhodné hledat jiné způsoby testování autonomních vozidel, před tím, než se pustí do reálného provozu. Toho je možné dosáhnout například počítačovou simulací s využitím simulačního modelu vozidla, jehož chování je možné podstatně rychleji a levněji upravit než u reálného vozidla. Vytvořením právě takového modelu a prostředí, ve kterém ho lze testovat se zabývá tato práce. Práce zahrnuje předdefinované scénáře pro analýzu chování navrženého modelu za situací, které v daných scénářích mohou nastat. Se znalostí JavaScriptu je také možné přidávat do simulačního nástroje další scénáře a upravovat libovolně model autonomního vozidla, podle toho, jaké se očekává chování v dané situaci. Tato práce by tedy mohla být přínosem v oblasti vývoje autonomních vozidel především ve fázi testování chování vozidla.

Toto téma jsem zvolil kvůli zájmu o nové trendy ve vývoji automatizace. Myslím si, že je potřeba zkoumat cesty jak kontrolovat chování vozidel v krizových situacích dříve, než taková situace nastane v provozu na skutečné silnici. Bohužel ani autonomní vozidlo nedokáže bezpečně vyřešit všechny situace, které se mohou na cestách vyskytnout, je tak potřeba také řešit určité morální zásady, které ovšem stroj nemá. Chování autonomních vozidel není nikým pevně definováno, čeká nás tak v budoucnosti otázka, jak se vozidlo má zacho-

vat, pokud ze situace neplyne východisko bez možných ztrát na životech. Věřím tomu, že by i v tomhle ohledu mohla být tato práce přínosem.

Kapitola 2 nazvaná „Autonomní vozidla“ se zabývá shrnutím poznatků o autonomních vozidlech, jejich vybavením a uvádí také potřebné znalosti pro návrh modelu. V kapitole 3 jsou představeny technologie využité pro tvorbu simulačního modelu, prostředí pro běh simulace a nástroje pro analýzu informací. Se znalostmi uvedenými v těchto kapitolách lze vytvořit návrh modelu autonomního vozidla, o tom pojednává kapitola 4. Po vytvoření návrhu modelu je možné začít implementaci simulačního modelu s využitím dříve uvedených technologií. Kapitola 5 uvádí čtenáře do problematiky implementace a použitých postupů při implementaci. Následně je v kapitole 6 vyhodnocen výstup simulací za využití implementovaného nástroje pro analýzu informací posbíraných během experimentů se simulačním modelem.

Kapitola 2

Autonomní vozidla

V této kapitole je popsáno současné technické vybavení autonomních vozidel a obsahuje také porovnání vybraných autonomních vozidel. Následně jsou uvedeny potřebné fyzikální vlastnosti a vzorce relevantní k tématu.

2.1 Rozdělení podle stupně samočinnosti

Vozidla jsou rozdělena podle standardu SAE J3016 do šesti stupňů samočinnosti.

<table>
<tr><th>Úroveň automatizace</th><th colspan="2">Za co zodpovídá řidič?</th><th>S čím pomáhá vozidlo?</th><th>Příklad funkcí vozidla</th></tr>
<tr><td>SAE level 0</td><td colspan="2" rowspan="3">Vozidlo ovládá po celou dobu řidič.
I v případě, kdy jsou zapnuté podpůrné
funkce vozidla, řidič musí nestále dohlížet na
bezpečnost</td><td>varování řidiče a
nouzové akce</td><td>Nouzové brzdění,
hlídání slepého úhlu,
hlídání jízdy v pruzích</td></tr>
<tr><td>SAE level 1</td><td>asistence
při zatáčení nebo
změnách rychlosti</td><td>Udržování v pruhu
nebo
adaptivní tempomat</td></tr>
<tr><td>SAE level 2</td><td>asistence
při zatáčení i
změnách rychlosti</td><td>Udržování v pruhu a
adaptivní tempomat</td></tr>
<tr><td>SAE level 3</td><td rowspan="3">Pokud je
automatizované
řízení aktivní,
řidič vozidlo neovládá</td><td>Pokud vozidlo požádá,
řidič musí převzít kontrolu</td><td rowspan="2">samostatné za
určitých podmínek</td><td>Řízení
v dopravní zácpě</td></tr>
<tr><td>SAE level 4</td><td rowspan="2">Vozidlo nebude žádat
řidiče
o převzetí kontroly</td><td>Lokální taxi,
pedály a volant
nemusí být ve vozidle
nainstalovány</td></tr>
<tr><td>SAE level 5</td><td>samostatné
za všech podmínek</td><td>Viz. level 4,
jen bez omezení</td></tr>
</table>

Tabulka 2.1: Přehled úrovní automatizace vozidel podle infografiky ke standardu SAE J3016. Tabulka zobrazuje jednotlivé úrovně a rozsah automatizace. Nultá až druhá úroveň označuje vozidla vyžadující dohled řidiče, vozidla vyšší úrovně automatizace zvládnou řídit samostatně, většina však s omezeními.[2]

Nultý až druhý stupeň jsou na silnicích běžné, jedná se o vozidla vybavena například adaptivním tempomatem, brzdným systémem ABS nebo automatickým nouzovým brzděním. Třetí stupeň již splňují známá autonomní vozidla, jedná se o vozidla, která dokáží převzít kontrolu nad řízením, ale vyžadují stálý dohled řidiče a případný zásah.

[2]Inspirováno: https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic, dostupná i tištěná verze

Čtvrtý stupeň automatizace je cílem většiny dnešních automobilek zabývající se vývojem autonomních vozidel. Jedná se o vozidla, která dokáží operovat zcela bez zásahu řidiče, ale stále mají nějaká omezení. Vozidla mohou být omezena například zmapovanou oblastí nebo počasím.

Posledním, pátým stupněm je již kompletní odstranění lidského prvku z řízení vozidla. V podstatě se jedná o čtvrtý stupeň automatizace bez dalších omezení. [8]

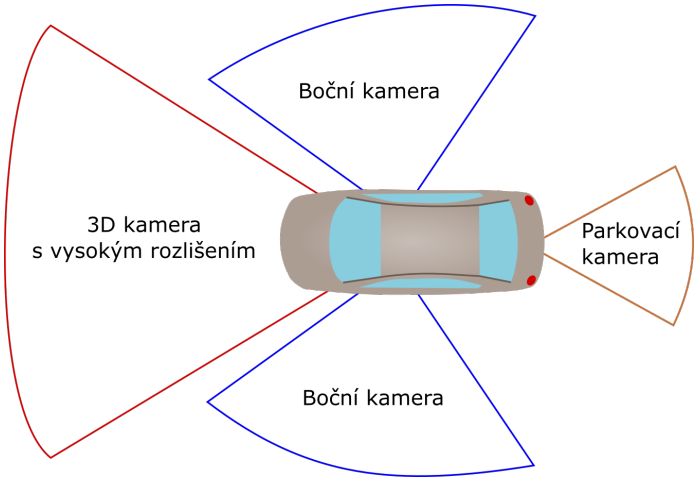
2.2 Technologické součásti autonomních vozidel

V této podkapitole jsou popsány jednotlivé prvky autonomních vozidel. Zaměřuje se pouze na prvky důležité v rámci tématu automatizace řízení, neuvádí brzdné systémy a podobné prvky vozidel. Rovněž se zde neuvádí systémy sdílení informací přes cloud ani GPS, jelikož se nejedná o technologie relevantní k projektu, přestože se reálně používají.

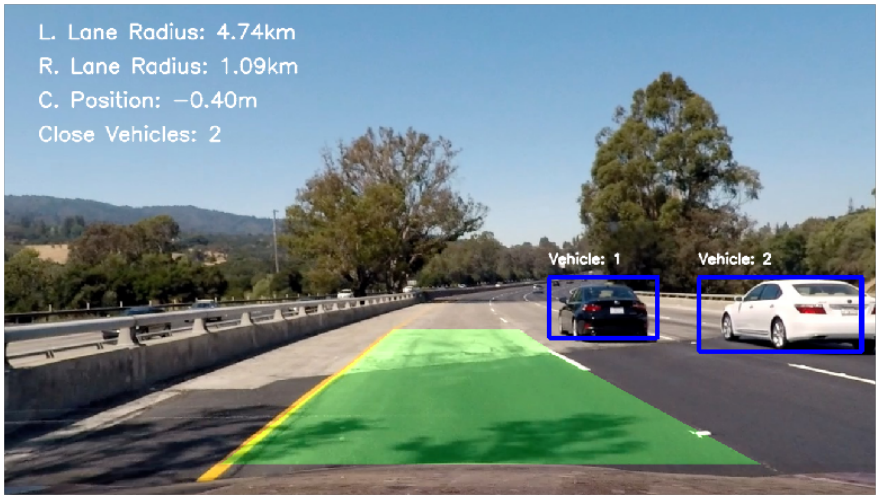
Kamerový systém

Kamerové systémy jsou z finančního hlediska nejefektivnějšími senzory autonomních vozidel. Mezi jejich hlavní využití patří detekce jízdních pruhů a svislého i vodorovného dopravního značení. Dokáží také měřit vzdálenosti objektů s pomocí výpočetních algoritmů.

Oproti Radaru a LiDARu mají vyšší počet selhání při rozpoznávání objektů, například nemusí rozpoznat tištěný obrázek na billboardu oproti reálnému objektu a také jsou lehce ovlivnitelné počasím. [2]



Obrázek 2.1: Příklad rozmístění kamerového systému na autonomním vozidle. Dnešní autonomní vozidla mají 360° pokrytí s využitím až 8 kamerových senzorů. Přední kamery se využívají na rozpoznání objektů (lidé, vozidla, dopravní značení), zadní slouží k parkování a boční pro obecný přehled o okolí vozidla.

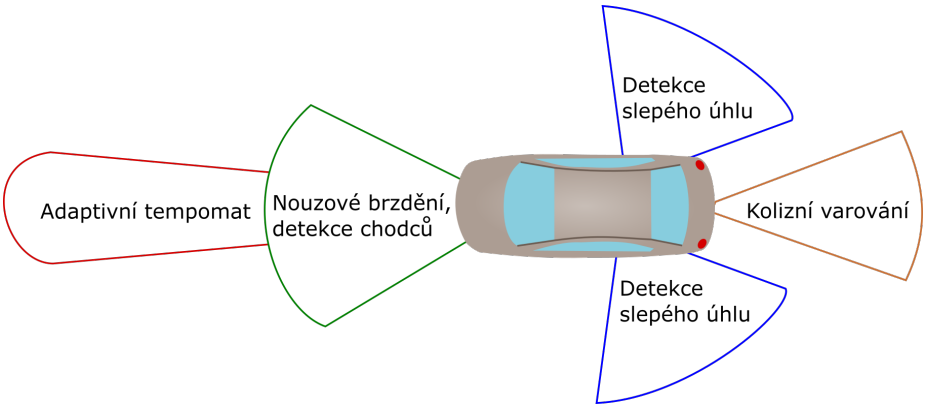


Obrázek 2.2: Ukázka použití kamerového systému na autonomním vozidle. Za pomoci dat získaných z kamery, dokáže počítač rozpoznat vzdálenosti, jízdní pruhy i typy objektů kolem vozidla. Rozpoznání typu objektu provádí za pomoci porovnávacích algoritmů, například s využitím souboru dat klasifikovaných podle typu objektu.[4]

RADAR – radio detection and ranging

Radar využívá k měření vzdálenosti, úhlu a rychlosti okolních objektů radiové vlny. Přesnost a detekční vzdálenost radaru závisí na jeho frekvenčním pásmu. Radar měří vzdálenost na základě rozdílu času od vyslání radiové vlny do jejího návratu.

Jeho využití je výhodné především kvůli nižší ceně než u LiDARu, ale také vyšší spolehlivosti i ve zhoršených podmínkách. Dokáže určit i relativní rychlost detekovaných objektů. [2]



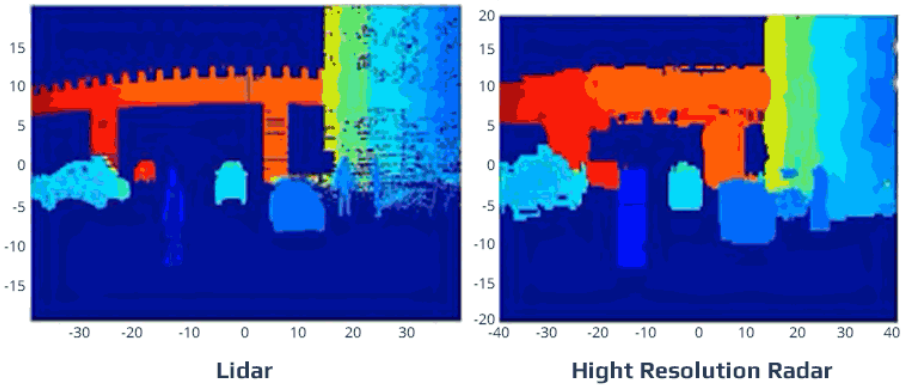
Obrázek 2.3: Příklad sestavy radarového systému autonomního vozidla. Radarů se běžně používá na vozidle větší množství, každý může mít jiné zaměření. Mezi využití radaru patří přizpůsobení rychlosti nejbližšímu objektu, detekce slepých úhlů a varování před kolizí.

[4]Převzato z: https://medium.com/@ricardo.zuccolo/self-driving-cars-opencv-and-svm-machine-learning-with-scikit-learn-for-vehicle-detection-on-the-bf88860e055

Radary se mohou dále dělit podle jejich využití:

Využití	vzdálenost (m) / zorné pole	frekvence (GHz)
Adaptivní tempomat	200 / 10°	77
Detekce slepého úhlu	20 / 30°	>= 24
Varování před kolizí	30	>= 24

Vyšší frekvence může být použita ke zvýšení dosahu, například u adaptivního tempomatu, nebo rozšíření zorného pole u radarů využívaných k včasnému varování před kolizí. [3]



Obrázek 2.4: Porovnání vize LiDARu a radaru s vysokým rozlišením. Z obrázku je patrné, že radar zdaleka nedosahuje rozpoznávacích schopností LiDARu. Tento nedostatek je možné částečně kompenzovat kamerovým systémem.[6]

LIDAR - light detection and ranging

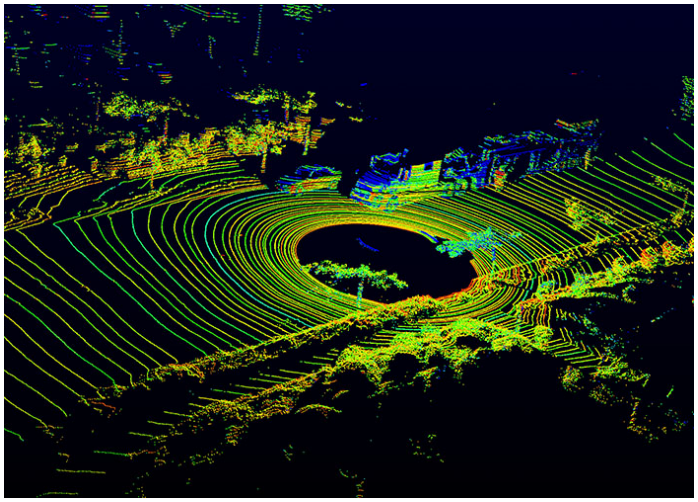
LiDAR měří vzdálenost objektů na základě rozdílu časů vyslání a přijetí světelného impulzu z emitoru, laserové diody, který se po odrazu od objektu vrací do detektoru, umístěného poblíž emitoru. Shromážděná data následně využívá ke generování trojrozměrné reprezentace okolního prostředí.

Dnešní LiDARy dosahují vzdáleností přes 250 metru a nabízí celou řadu výhod pro autonomní vozidla. Jednou z podstatných výhod je vysoké rozlišení a přesnost získaných dat. Oproti radaru může jeden otočný senzor pokrýt celých 360° okolo vozidla, dokáže rozpoznat více objektů ve stejné linii a je rychlejší, neboť používá světelné paprsky, ale také mnohonásobně dražší. Mezi nevýhody patří také snížená spolehlivost za příliš silného okolního světla. [2]

[6] Převzato z: https://www.intellias.com/the-ultimate-sensor-battle-lidar-vs-radar

Senzor	Orientační cena
LiDAR	$7000
Radar	$50 - $150
Kamera	$25 - $200
Ultrazvuk	$15 - $20

Tabulka 2.2: Orientační porovnání cen senzorů[7]



Obrázek 2.5: Ukázka 3D výstupu LiDAR senzoru firmy Velodyne. Jedná se o rotující senzor, mapující celé okolí vozidla, včetně vertikálního rozměru, oproti radaru tak dokáže detekovat více objektů za sebou v jedné linii. Ze získaných dat skládá trojrozměrný obraz okolí, který lze využít jako podklad pro rozhodovací algoritmy vozidla.[9]

Ultrazvukové senzory

Ultrazvukové senzory využívají pro měření vzdálenosti od okolních objektů zvukové vlny. Senzor vyšle zvukovou vlnu určité frekvence a na základě rozdílu času odeslání a přijetí zvukové vlny určí vzdálenost.

Jedná se o nejlevnější druh senzorů a zároveň velmi spolehlivý na krátké vzdálenosti. Ultrazvukové senzory jsou hojně využívány například pro parkovací asistenty. V případě autonomních vozidel se využívají také k detekci blízkých kolizí po stranách vozidla. [2]

Zvukové vlny vydávané senzorem se pohybují ve frekvencích nad 20 kHz a jejich akustický tlak je přes 100 dB. Dosah se běžně pohybuje kolem 2.5 metru, ale může být i vyšší [4].

2.3 Stručný přehled aktuálně vyvíjených vozidel

Ve většině zemí je nyní výskyt **plně** autonomních vozidel na veřejných komunikacích zakázán. Přesto se různé společnosti pokouší přiblížit k vytvoření plně autonomního vozidla. V této kapitole jsou stručně popsána vybraná vozidla, která jsou již testována v reálném provozu.

[7] Zdroj: https://www.automotivelectronics.com/cost-of-components-of-a-self-driving-car
[9] Převzato z: https://velodynelidar.com/hdl-64e.html

Tesla

Vozidla značky Tesla již několik let obsahují autopilota, který využívá následující sestavu senzorů:

- Přední radar – dosah přes 150 metrů
- Kamerový systém – 8 kamer s dosahem přes 250 metrů
- Ultrazvukové senzory – 12 upravených senzorů
- GPS

Autopilot je schopný zatáčení, úpravy rychlosti, detekce překážek i parkování. Tesla také tvrdí, že je s využitím aktuálního vybavení vozidel, což mimo jiné znamená i bez použití LiDARu, schopna vytvořit plně autonomní vozidlo pátého stupně viz. podkapitola 2.1. [7]

Ultrazvukové senzory vozidel Tesla mají dosah přes 8 metrů od vozidla, zároveň jsou upraveny tak, že dokáží detekovat kolize i skrze dveře vozidla. Tesla využívá senzory k parkování i k detekci kolizí v ostatních pruzích a to za jakékoli rychlosti. [5]

Waymo

Waymo zvolilo oproti firmě Tesla opačnou strategii, vsadili na využití LiDARu, který nabízí vyšší přesnost a rozsah. Vyvinuli také tři typy LiDARů s rozdílným dosahem, a začínají je sériově vyrábět, čímž postupně snižují jejich cenu.

Vozidla využívají následující senzory:

- LiDAR – 4x krátký dosah, 1x střední dosah, 1x dlouhý dosah
- Kamerový systém – 8 kamer
- Ultrasonické senzory – 8 senzorů
- GPS

Oproti vozidlům značky Tesla je zde značná nevýhoda v ceně, ale s postupným rozšiřováním výroby LiDARů ceny pravděpodobně klesnou, tento trend lze sledovat již dnes, kdy ceny klesly během pár let o více než 90%.

Aktuálně firma experimentuje s vytvořením autonomní taxi služby. To může mít výhodu v tom, že se vozidla pohybují ve známém prostředí – taxi služby většinou operují na omezené oblasti a není tedy nutná podpora plné automatizace – pokud se jim to podaří, budou mít autonomní vozidlo čtvrtého stupně. [1]

2.4 Fyzikální vzorce pohybu vozidla

Tato sekce popisuje vzorce potřebné pro model autonomního vozidla. Zaměřuje se na pohyb vozidla, vzhledem k zaměření práce se zabývá pouze základními vlastnostmi, nepočítá s využitím systému ABS, skluzy a dalšími podobnými jevy. Tyto rovnice poslouží jako základní kameny k utváření dalších rovnic a funkcionalit popsaných v dalších kapitolách.

Rovnoměrně zrychlený pohyb

Pro simulaci je nezbytné vozidlo uvést do pohybu a také zjistit jeho aktuální rychlost. K tomuto účelu lze použít vzorec rovnoměrně zrychleného pohybu.

Ten je definován:

$$v = v_0 + a * t \tag{2.1}$$

Značka	Popis	Jednotka
v_0	počáteční rychlost	m/s
v	okamžitá rychlosti	m/s
a	zrychlení vozidla	m/s^2
t	uplynulý čas	s

Rovnoměrně zpomalený pohyb

Pokud bylo vozidlo uvedeno do pohybu, je nutné ho dokázat také zpomalit a zastavit. K tomuto účelu slouží vzorec rovnoměrně zpomaleného pohybu, ten je odvozen od vzorce rovnoměrně zrychleného pohybu 2.1. Rozdíl mezi těmito vzorci je ve zrychlení – zatímco u předchozího vzorce je kladné, u zpomaleného pohybu je záporné.

$$v = v_0 - a * t \tag{2.2}$$

Brzdná dráha

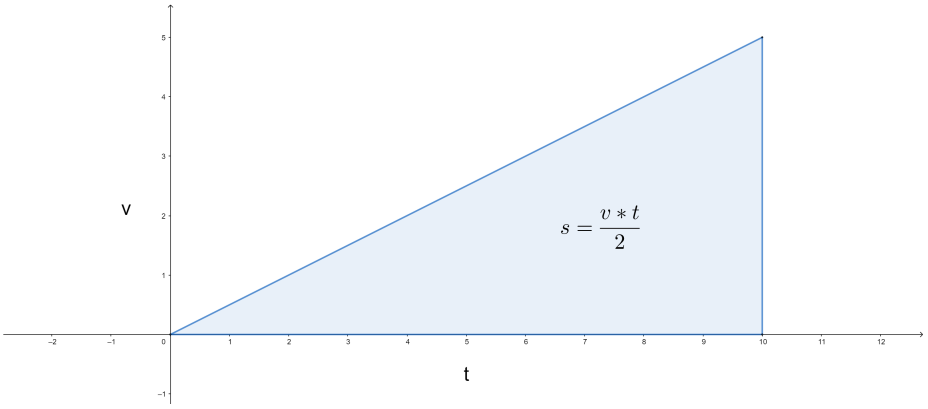
Mimo samotné zpomalení vozidla až případně na nulovou rychlost, je důležité také znát správný okamžik, ve který má vozidlo začít brzdit.

Vzorec brzdné dráhy odvodím z následujícího obecného vzorce dráhy:

$$s = v * t \tag{2.3}$$

Značka	Popis	Jednotka
v	okamžitá rychlosti	m/s
s	dráha	m
t	uplynulý čas	s

Dráha je počet metrů, které vozidlo ujede za stanovený čas při dané okamžité rychlosti. Protože se ale vozidlo nepohybuje stálou okamžitou rychlostí, odvodím vzorec dráhy rovnoměrně zrychleného pohybu. Tu lze spočítat jako plochu pod grafem, viz. obrázek 2.6.



Obrázek 2.6: Graf k výpočtu dráhy rovnoměrně zrychleného pohybu. Celkovou dráhu lze spočítat jako plochu pod grafem

Protože okamžitá rychlost nebude konstantní, nahradím okamžitou rychlost zrychlením z rovnice rovnoměrně zrychleného pohybu 2.1.

$$s = \frac{v * t}{2} = \frac{(a * t) * t}{2} = \frac{1}{2} * a * t^2 \tag{2.4}$$

Získám tak kompletní vzorec dráhy rovnoměrně zrychleného pohybu.

$$s = \frac{1}{2} * a * t^2 \tag{2.5}$$

Zbývá odvodit vzorec brzdné dráhy. Ten poskládáme ze vzorců 2.5 a 2.2. Předem neznáme čas ani brzdnou dráhu, čas je možné vypočítat s využitím aktuální rychlosti, cílové rychlosti a zrychleni. Při zpomalení na nenulovou cílovou rychlost v, vypočítám čas následovně:

$$t = \frac{v_0 - v}{a} \tag{2.6}$$

Pokud se jedná o zastavení vozidla, dosadím rychlost $v = 0m/s$, protože vím, že cílová rychlost bude nulová.

$$t = \frac{v_0}{a} \tag{2.7}$$

K získání brzdné dráhy potřebné ke zpomalení na nulovou rychlost dosadím do rovnice 2.5 za proměnnou t rovnici pro výpočet času 2.7.

$$s = \frac{1}{2} * \frac{v_0^2}{a} \tag{2.8}$$

A pro brzdnou dráhu potřebné ke zpomalení na nenulovou rychlost, nahradím čas za rovnici 2.6

$$s = \frac{1}{2} * \frac{(v_0 - v)^2}{a} \tag{2.9}$$

Úhlová rychlost

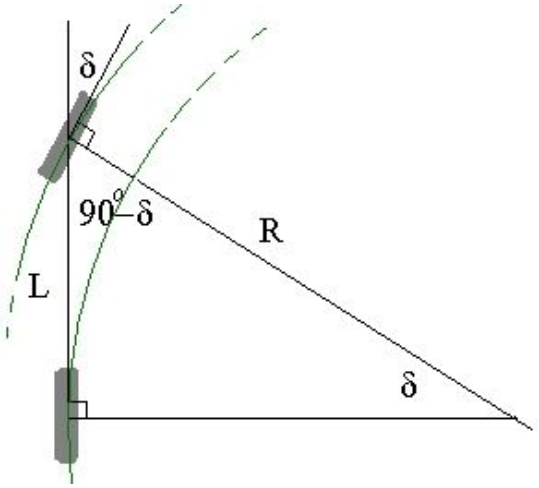
S pomocí předchozích rovnic dokáže vozidlo zrychlit, zastavit a dokonce zjistit potřebnou brzdnou dráhu. Modelované vozidlo ale musí také zvládnout zatáčení.

Vzhledem k zaměření práce na samočinné řízení vozidla, vynechám smyky a tření kol o vozovku. Využiji tedy k výpočtu zatáčení vzorec pro výpočet úhlové rychlosti.

$$\omega = \frac{v}{R} \tag{2.10}$$

$$R = \frac{L}{\sin(\delta)} \tag{2.11}$$

Značka	Popis	Jednotka
v	okamžitá rychlost	m/s
ω	úhlová rychlost	rad/s
R	Poloměr kružnice, po které se pohybuje	cm
L	Vzdálenost mezi přední a zadní nápravou	cm
δ	Úhel přední nápravy	rad



Obrázek 2.7: Ilustrace výpočtu úhlové rychlosti. Vzdálenost mezi nápravou se počítá "do kříže", například vzdálenost mezi předním levým kolem a pravým zadním kolem.[11]

2.5 Shrnutí

V této sekci byly shrnuty důležité poznatky sloužící jako podklad k návrhu a implementaci modelu. Zabývala se aktuálním vybavením autonomních vozidel, stručným popisem jednotlivých technických prvků a také důležitými vzorci pro vytvoření modelu.

[11] Převzato z: https://www.jimb.de/Projects/Car%20Physics.htm

Kapitola 3

Technologie využité k vytvoření modelu a simulaci

Tato kapitola pojednává o vybraných technologiích k vytvoření navrženého abstraktního modelu a následnou simulaci prováděnou nad simulačním modelem. Nejprve však krátce shrnuje užitou terminologii, aby bylo zřejmé, co je cílem práce.

3.1 Terminologie

Tato podkapitola popisuje základní terminologické výrazy užívané v této práci. Podkapitola čerpá z prezentace [6].

Systém

Systém je soubor prvků systému, které mají mezi sebou určité vazby. Rozlišují se dva typy systémů:

- Reálný systém
- Nereálný systém

Tato práce vychází z reálného systému – tímto systémem je vozidlo, které se pohybuje v určité oblasti. K provedení správné akce na základě aktuální situace v jeho blízkém okolí využívá senzory, pomocí kterých získává informace o okolních objektech a jízdních pruzích viz. kapitola 2.

Model

Napodobenina systému jiným systémem. Může se jednat například o matematický nebo fyzikální model.

Modelování

Modelování je metoda vytváření modelů systémů. Můžeme modelovat pouze to co známe a dokážeme popsat.

Simulace

Simulace zahrnuje experimentování s modelem. Pomocí simulace nad modelem získáváme nové znalosti o systému. Je vhodná především v případech, kdy nemáme dostatečné prostředky pro experimentování s reálným modelem. Pro experimentování můžeme využít simulační model, který má v takových případech nižší náklady než experimentování s reálným modelem.

Abstraktní model

Abstraktní model je zjednodušený model reálného systému. Platí mezi nimi tedy homomorfní vztah – vynecháváme prvky které pro modelovaný systém nejsou podstatné. Ke zjednodušení dochází proto, že popsat komplexní chování reálného systému je příliš obtížné, ne-li nemožné. V abstraktním modelu se tedy zaměřujeme pouze na prvky, které jsou podstatné pro náš cíl. Vztahy mezi jednotlivými prvky zjednodušit nelze, je potřeba aby byly stejné jako v reálném systému.

Abstraktní model může být reprezentován například konečným automatem, pomocí rovnic nebo Petriho sítí.

Simulační model

Mezi simulačním a abstraktním modelem platí izomorfní vztah – vyžaduje shodnost prvků systému i jejich vztahů 1:1. Simulační model je ve své podstatě abstraktní model, nad kterým se dají provádět experimenty, může být například zapsaný formou počítačového programu. K zápisu takového modelu může být využit libovolný programovací jazyk. Existují taktéž knihovny a programy, které zjednodušují tvorbu modelu a celkovou práci se simulací a simulačním modelem, například SIMLIB, Modelica a další.

Verifikace a validace

Validace je ověření platnosti simulačního modelu. V tomto procesu se snažíme dokázat, že pracujeme s modelem adekvátním modelovanému systému. Validaci je potřeba provádět neustále, pokud se výsledky simulace odlišují od očekávaného chování původního systému, je potřeba provést korekci modelu.

Verifikace je ověření platnosti izomorfního vztahu mezi simulačním a abstraktním modelem. Jako verifikaci si lze představit ladění chování modelu zapsaného v programovacím jazyce oproti navrženému abstraktnímu modelu.

3.2 JavaScript

JavaScript je interpretovaný, dynamicky typovaný programovací jazyk s možností využití objektově orientovaného přístupu. Původně byl určen k obohacení webových stránek v prohlížeči. Syntakticky se podobá například jazyku Java, C nebo C#. Na klientské straně umožňuje například měnit strukturu HTML dokumentů nebo přidání validace formulářů v reálném čase pomocí tzv. klientských skriptů. Skripty je možné vepsat přímo do HTML kódu, nebo rozdělit do samostatných souborů a následně odkazovat ve více HTML dokumentech.

V dnešní době JavaScript neslouží pouze k vytváření skriptů běžících na klientské straně, ale využívá se také na straně serveru, kde běží na tzv. “JavaScript engine”. Mezi nejznámější patří například V8 engine, který se využívá v jádru prohlížeče Google Chrome.

JavaScript běžící v prohlížeči na klientské straně nemá přímý přístup k souborům na souborovém systému ani k paměti počítače a dalším informacím, které by mohly znamenat potencionální ohrožení uživatele.

Tyto omezení platí pro prohlížeče, ale s využitím jiných prostředí, například Node.js, je možné například zapisovat i číst ze souborového systému, měnit oprávnění souborů nebo odesílat síťové pakety. Se správným výběrem a nastavením prostředí je možné JavaScript využít téměř pro jakýkoli účel – od jednoduchých klientských skriptů, přes TCP/UDP servery až po desktopové aplikace s GUI.

V této práci se programovací jazyk JavaScript využívá k vytvoření simulačního modelu, simulačního prostředí, sběru dat i k následné analýze. Projekt běží v prostředí Node.js s využitím frameworku Electron.

Node.js

Node.js[1] je open-source runtime prostředí pro JavaScript. Využívá událostmi řízené programování, díky událostem je většina I/O akcí neblokujících, tzn. neblokují vykonávání hlavního vlákna, vhodné využití je tedy pro vytváření škálovatelných síťových aplikací, například webových serverů.

Důležitou součástí Node.js jsou moduly, které umožňují mimo jiné získat informace o operačním systému (využití procesorů, paměti, ...), vytváření nových procesů, odesílání HTTP požadavků, čtení a ukládání do souborů. Kromě oficiálních modulů je možné přidat i externí balíčky, které mohou podporovat například generování PDF souborů, práci s obrázky, zamykání souborů atd.

Electron JS

Electron[2] je open source knihovna pro vytváření multiplatformních desktopových aplikací za pomoci HTML, CSS a Javascriptu. Electron spojuje chromium a Node.js do jednoho runtime. Výsledné programy jsou spustitelné na systémech Linux, Windows i macOS.

Mezi známé aplikace využívající knihovny Electron patří například Skype, Visual Studio Code nebo Atom.

Electron nabízí také konzoli pro ladění vyvíjených aplikací včetně analýzy výkonu atd., podobnou jako v prohlížeči Google Chrome.

Jako hlavní výhodu pro tento projekt, kvůli které jsem Electron zvolil je spustitelnost ve formě desktopové aplikace. Zároveň společně s Node.js nabízí možnost vytvoření skriptů, které se postarají o spuštění aplikace i ve více instancích zároveň. Je tedy možné spouštět simulace souběžně ve více procesech. Electron také podporuje vícevláknové aplikace s využitím tzv. web workers, to se ale v implementaci neosvědčilo vzhledem k vysoké režii synchronizace.

Plotly.js

Plotly.js[3] je open source JavaScriptová knihovna umožňující zpracovat informace do podoby interaktivního grafu. Jedná se o součást celku Plotly, což je skupina knihoven pro různé jazyky, kromě JavaScriptu podporují například Python a MATLAB.

[1] Zdroj: https://nodejs.org
[2] Zdroj: https://electronjs.org/
[3] Zdroj: https://plot.ly/javascript/

Vykreslování grafů lze lehce modifikovat, grafy jsou reprezentovány jako JSON objekty a každý z jejich prvků má atributy, které ovlivňují výsledný vzhled grafu. Grafický výstup je ve vektorovém formátu SVG, kvalitní a škálovatelný.

Plotly.js nabízí širokou škálu grafů, zahrnuje například:

Základní grafy – koláčový graf, sloupcový graf, čárový graf, ...

Statistické grafy – box plot, histogram, ...

Vědecké grafy – teplotní mapa, kobercový graf, radarový graf, ...

Finanční grafy – časová řada, ...

Ostatní grafy – mapy, 3D grafy, kombinované grafy, ...

3.3 HTML

HTML[4] (HyperText Markup Language) je značkovací jazyk využívaný pro tvorbu webových stránek. Dokumenty ve formátu HTML jsou základním kamenem webových stránek, nemusí se ale jednat o konkrétní soubory, mohou být generovány dynamicky na straně serveru. Zápis HTML se skládá ze značek, které oddělují text od HTML kódu. Značky mohou mít atributy, upravující jejich funkcionalitu. Značky mění výslednou podobu stránek, používají se například pro zvýraznění textu, přidání odkazů nebo prosté logické oddělení sekcí. Nejčastěji se ke značkám přidávají atributy pro identifikaci, ty najdou využití v JavaScriptu nebo při vytváření stylovacích pravidel v CSS.

HTML5 Canvas

HTML značka `<canvas>`[5] byla zavedena v HTML5. Tato značka se používá jako plátno k vykreslování grafický prvků, samotné vykreslování není možné provádět přímo v HTML, je nutné použít JavaScript. Většina prohlížečů implementuje 2D kontext canvasu, některé ale podporují i 3D kontext.

Canvas se v této práci používá pro vykreslení celého simulačního prostředí. Pro tento účel jsou však využity pouze základní možnosti, a to vykreslení přímek, textu a obdélníků.

3.4 Existující nástroje pro simulaci

Tato sekce krátce představuje existující nástroje pro vytváření simulací.

SimJS je JavaScriptová knihovna určená pro modelování diskrétních systémů. Knihovna je založena na událostmi řízeném návrhovém vzoru. Pro JavaScript se vývojáři rozhodli z důvodu široké dostupnosti webových aplikací. Software je volně dostupný.

Sim4edu je prostředí vytvořené v JavaScriptu, určené pro modelování a simulaci přímo na webu. Umožňuje modelovat diskrétní i spojité systémy. Jako jednu z výhod uvádí vývojáři možnost sdílení simulací a spouštění přímo v prohlížeči, ale i na tabletech a telefonech. Software je volně dostupný.

[4] Zdroj: www.yourhtmlsource.com

[5] Zdroj: https://www.zdrojak.cz/clanky/zaciname-z-html5-canvasem/

Simulink je nádstavba MATLABu pro modelování a simulaci. Prostředí běží jako desktopová aplikace a poskytuje větší výkon než JavaScriptová řešení, také umožňuje průběžně testovat validitu modelů. Je možné využít připravený balíček pro simulaci autonomních vozidel. Jedná se o placený software.

I přes existenci simulačních nástrojů jsem se rozhodl vytvořit nové simulační prostředí, hlavně proto, že jsem chtěl mít volnou ruku a nebýt nijak omezen zvoleným prostředím. Dalším z důvodů byla vizualizace simulace, kterou většina z dostupných nástrojů nepodporuje a nebo ji podporuje pouze v omezené podobě.

3.5 Shrnutí

Tato kapitola shrnula základní terminologii související s tématem práce. Stručně popisuje také technologie využité k implementaci modelu, samotné simulaci a také nástroje pro analýzu získaných dat.

Tyto technologie byly zvoleny pro jejich flexibilitu a snadné rozšíření bez limitů, které mohou obsahovat kompletní nástroje určené přímo pro vytváření modelů a simulací. Zároveň poskytují možnost vizualizace probíhající simulace, kompatibilitu napříč operačními systémy, využití více procesů a potencionálně je s použitím daných technologií možné spustit výsledný projekt přímo ve webovém prohlížeči.

Kapitola 4

Návrh modelu

Tato kapitola popisuje samotný návrh abstraktního modelu. Seznamuje čtenáře s požadavky na model autonomního vozidla a zahrnuje model rozhodování vozidla zobrazený pomocí konečných automatů.

4.1 Požadavky na abstraktní model

Tato sekce se věnuje požadavkům na model, uvádí výčet prvků systému, provedené abstrakce a také zanedbané prvky systému. Jedná se o otevřený, kombinovaný deterministický systém. Okolí modelovaného systému obsahuje nedeterministické prvky.

Prvky modelovaného systému

Reálný systém, na který se zaměříme, je autonomní vozidlo pohybující se v provozu na dvouproudé silnici. Tento systém obsahuje mimo samotné autonomní vozidlo také senzory, které jsou jeho "očima" a pomáhají počítači učinit správné rozhodnutí. Takových senzorů je v autonomním vozidle od každého druhu několik, viz. kapitola 2, je tedy nutné je zahrnout do abstraktního modelu.

Nosné prvky systému jsou:

- autonomní vozidlo
- stavový automat autonomního vozidla
- senzory

Autonomní vozidlo

Autonomní vozidlo rozhoduje deterministicky o nadcházející akci na základě informací přijatých pomocí senzorů a aktuálního vnitřního stavu.

U samotného vozidla nás zajímají pouze základní jízdní vlastnosti, které mu umožní pohyb v rámci okolí systému. Naším cílem není modelovat dokonalý abstraktní model vozidla se všemi jeho reálnými vlastnostmi, zaměřujeme se spíše na rozhodování vozidla. Pro přiblížení se reálnému okolí systému budeme v simulaci zohledňovat okolní podmínky, a to konkrétně počasí a maximální definovanou rychlost – například omezení rychlosti ve městě a námrazu na silnici. Zajímá nás tedy:

- Zrychlení za daných okolních podmínek

- Zpomalení za daných okolních podmínek
- Otáčení se za daných okolních podmínek

Zachování výše uvedených vlastností je předpokladem validity modelu, pokud nebudou odpovídat původnímu systému, je potřeba model upravit. Vzhledem k povaze simulace jsou pro nás nepodstatné například následující vlastnosti:

- Efektivita jednotlivých operací – systém ABS, druh brzdových destiček a podobně. Vozidlo by si mělo poradit bez ohledu na efektivitu daných součástí.
- Skluz při zatáčení
- Defekty, opotřebování součástek
- Vliv řidiče – autonomní vozidlo ho nepotřebuje

Senzory

Senzory jsou neodmyslitelnou částí výsledného modelu, ovšem i přes jejich důležitost není pro tuto práci podstatné modelovat všechny jejich vlastnosti. Vozidlo potřebuje pouze získat informace o tom, zda se před ním, vedle něj, nebo za ním vyskytuje nějaký prvek okolí systému, který by mohl být relevantní z hlediska kolize, jeho případnou vzdálenost a rychlost. Zaměříme se tedy na následující vlastnosti:

- Dosah senzoru
- Šířka zorného pole senzoru
- Typ detekce – vidí pouze první objekt, nebo i více objektů ve stejné linii? (rozdíl radar vs LiDAR)

Naopak jsou pro nás nepodstatné následující faktory:

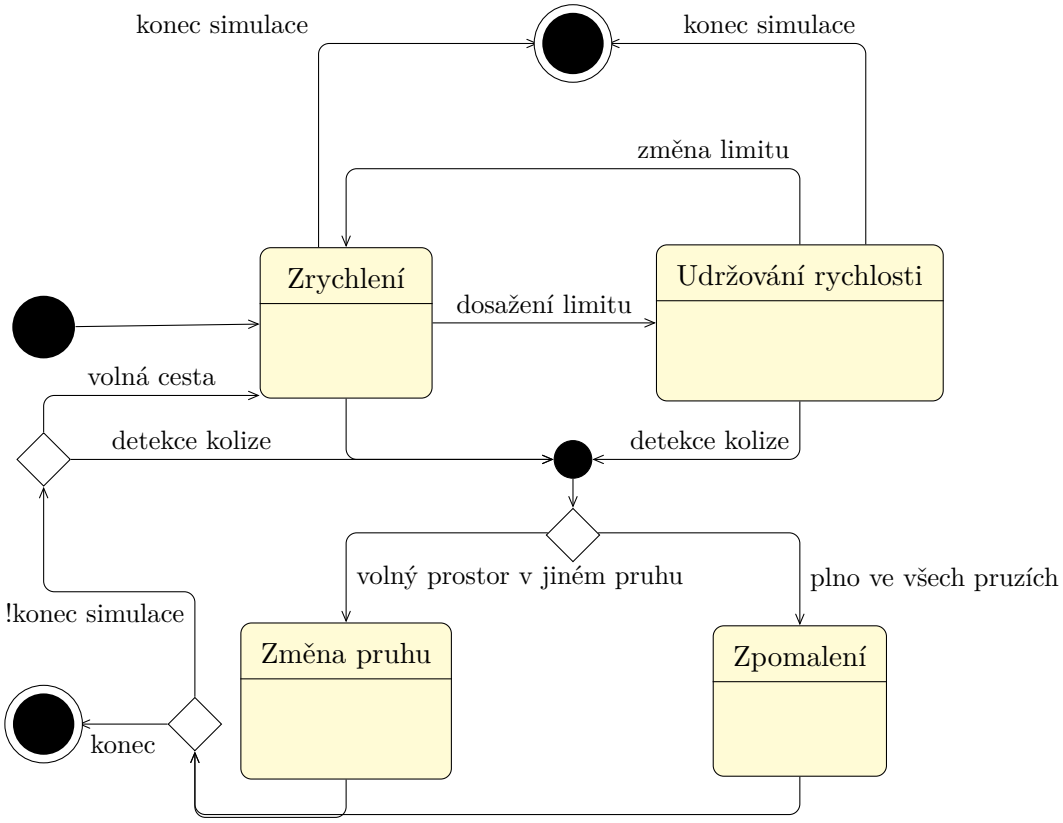
- Chyby při rozpoznávání – předpokládáme, že pokud je objekt v dosahu, bude správně rozpoznán
- Okolní podmínky a vliv počasí
- Technické detaily – nezáleží na tom, zda senzor vysílá radiové vlny, paprsky světla nebo ultrazvukové vlny.

4.2 Návrh rozhodovacího algoritmu a detekce překážek

V této sekci je popsán návrh algoritmu použitého pro rozhodování abstraktního modelu vozidla na základě aktuálního vnitřního stavu a informací získaných pomocí senzorů. Přesto, že je model vozidla i senzorů spojitý, jednotlivá rozhodnutí o změnách stavu lze považovat za diskrétní akce, proto je budu modelovat za pomoci konečného automatu obsahujícího stavy, ve kterých se vozidlo může nacházet a přechody mezi nimi.

Rozhodovací algoritmus

Níže uvedený konečný automat zobrazuje způsob rozhodování abstraktního modelu autonomního vozidla s ohledem na data získaná z jeho senzorů.



Obrázek 4.1: UML diagram konečného automatu rozhodování abstraktního modelu autonomního vozidla.

Po zahájení simulace se modelované vozidlo dostane do stavu zrychlování. Následně v tomto stavu setrvává do doby, než se jeho okamžitá rychlost vyrovná rychlostnímu omezení a nebo dojde k detekci kolize.

Pokud nastane situace, kdy vozidlo dosáhne rychlostního limitu, vozidlo již dále nezrychluje, pouze udržuje aktuální rychlost. V případě detekce relevantní kolize přejde do režimu vyhodnocení situace.

Vyhodnocení kolize má dva možné výstupy.

1. Vozidlo analýzou informací ze senzorů zjistí, že má dostatek prostoru k tomu aby předešlo kolizi změnou jízdního pruhu. V tomto případě pokračuje změnou jízdního pruhu, během změny pruhu stále kontroluje zrychlení a kolize.

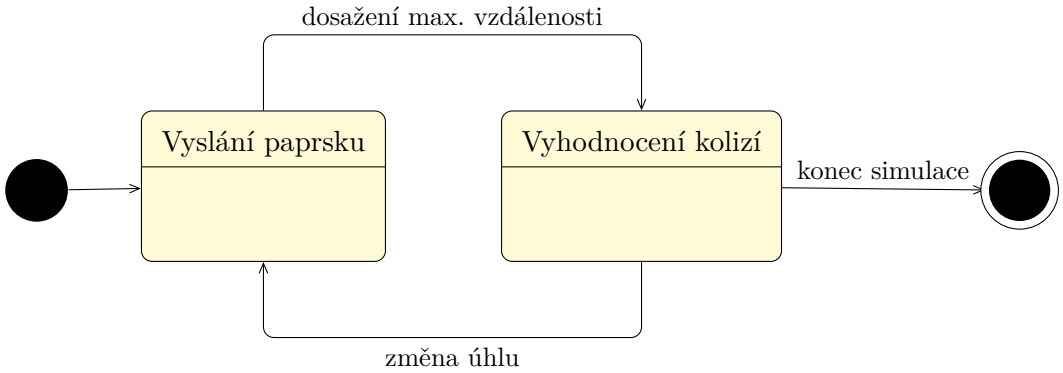
2. Vozidlo analýzou informací ze senzorů zjistí, že není volný žádný další jízdní pruh. V tomto případě mu nezbývá jiná možnost, než přejít do stavu zpomalování.

Jako relevantní kolizi uvažujeme objekt v relevantní vzdálenosti, který je ve stejném pruhu jako modelované vozidlo a zároveň má nižší relativní rychlost než modelované vozidlo a nebo objekt, který pravděpodobně modelovanému vozidlu zkříží cestu, například ve křižovatce.

Relevantní vzdálenost se skládá ze vzdálenosti potřebné k vyrovnání rychlosti kolizního objektu v součtu s bezpečnou vzdáleností, kterou by vozidlo mělo udržovat.

Detekce překážek

Pro model vozidla jsem se rozhodl využít senzory LiDAR, kvůli jejich schopnosti detekce vícero objektů ve stejné linii. Níže uvedený konečný automat zjednodušeně zobrazuje způsob funkce abstraktního modelu senzorů sloužících k detekci překážek.



Obrázek 4.2: UML diagram konečného automatu abstraktního modelu senzoru sloužícího k detekci překážek.

Model senzoru po startu simulace začíná vysláním světelného impulzu (paprsku) s daným počátečním úhlem. Po dosažení maximální vzdálenosti, jenž je určena pro každý senzor zvlášť, dojde k pomyslnému návratu paprsku a následnému vyhodnocení všech kolizí v jeho cestě.

Model senzoru má předem definovaný minimální a maximální úhel – tzv. zorné pole. Po vyhodnocení kolizí pro aktuální paprsek, dojde ke změně počátečního úhlu v rámci zorného pole senzoru a simulace senzoru pokračuje vysláním nového paprsku.

Zároveň s dosažením maximálního počátečního úhlu končí činnost senzoru pro aktuální čas simulace t. V simulačním čase $t + 1$ se opět změní počáteční úhel senzoru na minimální úhel zorného pole.

V rámci detekovaných kolizí poskytuje senzor také informace o jejich vzdálenosti od vozidla, rychlosti a směru detekovaného objektu.

4.3 Shrnutí

Tato kapitola popsala návrh abstraktního modelu, který se následně bude využívat pro implementaci simulačního modelu. Vysvětlila důvody zvolených abstrakcí a také chování důležitých prvků modelu.

Kapitola 5

Implementace

Tato kapitola se zabývá samotnou implementací simulačního modelu. Stručně popisuje zvolenou strukturu projektu, vytvořené objekty a použité metody. Vybrané části logiky popisuje do detailu i s obecným vysvětlením. Zabývá se také samotným průběhem simulace a její vizualizací.

5.1 Struktura projektu

Implementace je rozdělena do několika složek, souborů, tříd a objektů. Důležité třídy a objekty jsou popsány v této sekci.

Globální objekty a moduly

Projekt obsahuje několik důležitých globálních objektů, které jsou nezbytné pro korektní běh simulace a prostředí:

main – objekt, který se stará o řízení simulace, nastavuje prostředí, inkrementuje simulační čas a překresluje plátno.

autonomousVehicle – objekt rozšiřující logiku běžného vozidla o autonomní chování, má na starosti přechody mezi jednotlivými stavy na základě získaných informací

analyzeExporter – objekt sloužící k pravidelnému ukládání aktuálního stavu simulace a následnému exportu celého průběhu simulace do souboru.

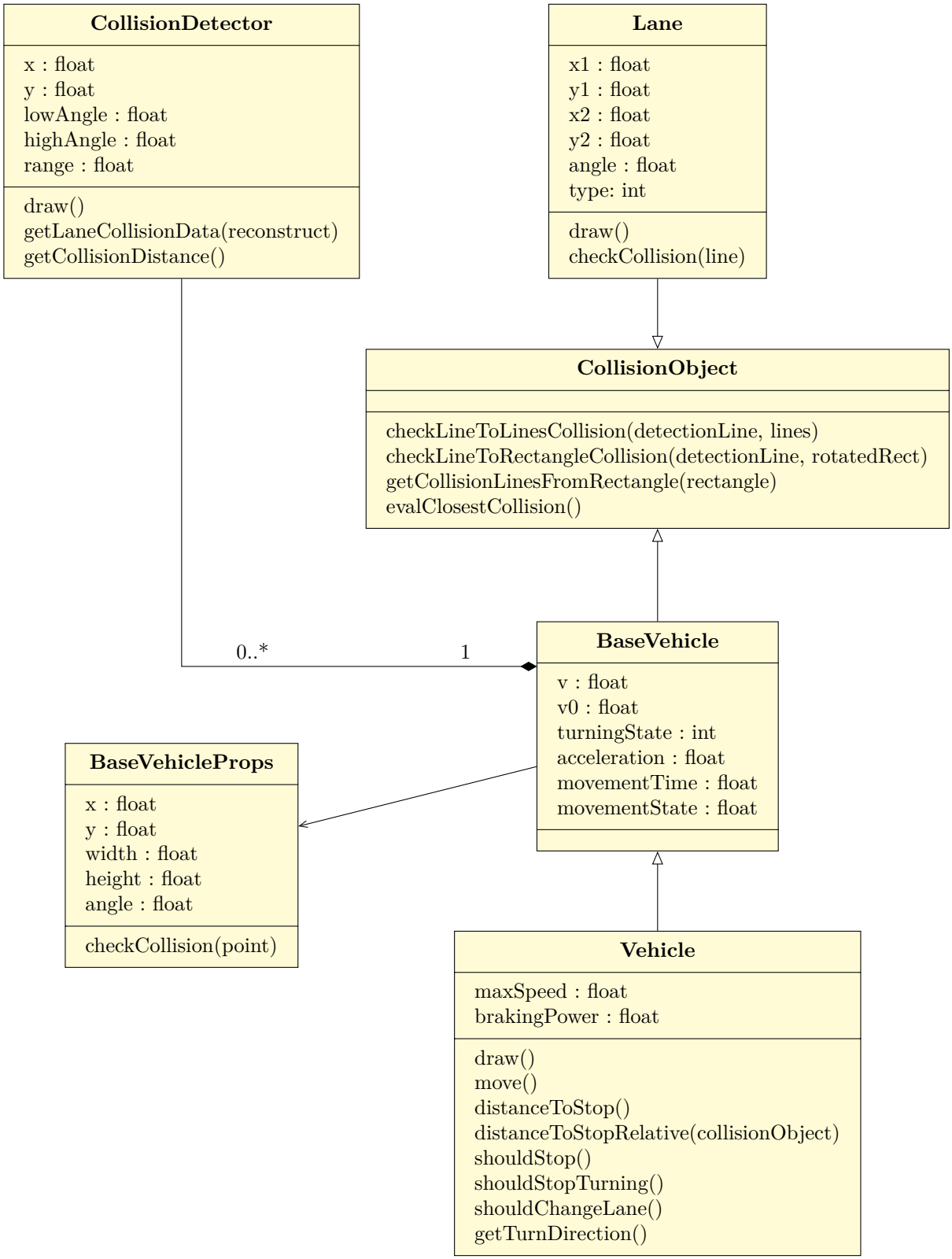
Kromě výše uvedených objektů obsahuje řešení i následující moduly:

maths – obsahuje často opakované výpočetní metody.

graphics – obsahuje metody pro vykreslení obrázků i geometrický tvarů.

Třídní diagram

Níže uvedený diagram zobrazuje vztahy mezi hlavními třídami a jejich vybrané důležité veřejné metody a atributy. Diagram nezobrazuje grafický a matematický modul ani globální objekty.



Obrázek 5.1: Diagram tříd zahrnující hlavní součásti projektu. Zobrazuje vazby mezi třídami, jejich podstatné atributy a metody.

5.2 Důležité algoritmy a rozhodovací logika

V této podsekci jsou důkladněji popsány vybrané algoritmy a logické prvky používané při simulaci. Nejedná se o kompletní výčet metod použitých v implementaci, uvedeny jsou pouze části, které zasluhují důkladnější popis než několika řádkový komentář v kódu.

Získání relativní rychlosti

Pokud vozidlo zaregistruje kolizi, může zcela zastavit a nebo zpomalit na relativní rychlost kolizního objektu. K tomu slouží metoda **getRelativeSpeed(toObject)** ve třídě Vehicle. Pokud by vektor rychlosti kolizního objektu směřoval stejným směrem jako vektor rychlosti modelovaného vozidla, stačilo by rychlosti navzájem odečíst, jenomže kolizní objekty mohou mířit různými směry, například v křižovatce může být rozdíl úhlů markantní. Pro tento účel jsem vytvořil vzorec na základě Pythagorovy věty 5.1.

$$c^2 = a^2 + b^2 \tag{5.1}$$

Senzory modelu poskytují o kolizních objektech následující informace:

- vzdálenost objektu
- rychlost objektu
- úhel objektu

Pro získání relativní rychlosti je nejprve třeba zjistit jakou rychlostí se kolizní objekt pohybuje ve stejném směru jako model autonomního vozidla. Graf 5.2 ilustruje výpočet této rychlosti, vektor rychlosti autonomního vozidla je rovnoběžný s osou x. Známé hodnoty:

- α – rozdíl mezi úhlem autonomního vozidla a úhlem kolizního objektu
- w – vektor rychlosti kolizního objektu (rychlost a úhel)

Pro získání hodnoty a je třeba nejprve dopočítat hodnotu b, tu získám pomocí goniometrické funkce sinus, kde b je délka protilehlé odvěsny a c je délka přepony:

$$\sin(\alpha) = \frac{b}{c} \tag{5.2}$$

Z rovnice 5.2 vyjádříme b:

$$b = c * \sin(\alpha) \tag{5.3}$$

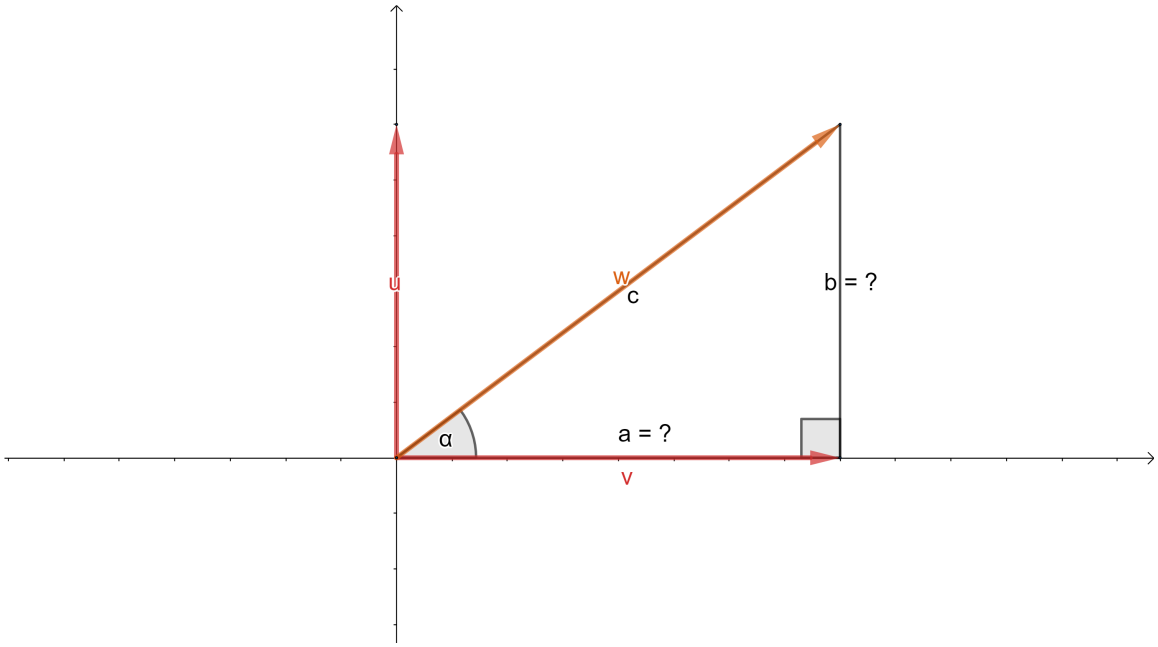
Zbývá zjistit hledanou hodnotu a. Dosadím tedy do původní rovnice 5.1 rovnici 5.3:

$$c^2 = a^2 + (c * \sin(\alpha))^2 \tag{5.4}$$

Po několika úpravách dostaneme finální rovnici pro výpočet rychlosti ve směru autonomního vozidla:

$$a = \pm\sqrt{c^2 - (c * \sin(\alpha))^2} \tag{5.5}$$

Výstup metody bude kladný, pokud má kolizní objekt namířeno stejným směrem, záporný v případě, kdy kolizní objekt míří směrem proti autonomnímu vozidlu – to znamená, že úhel α je mezi 90° a 270°.



Obrázek 5.2: Grafické znázornění mezivýpočtu relativní rychlosti objektu. Úhel α je rozdíl úhlů mezi vektory rychlosti dvou objektů. Objekt, který rychlost analyzuje má vektor rychlosti rovnoběžný s osou x. Vektor w je vektor rychlosti analyzovaného objektu. Cílem je získat rychlost analyzovaného objektu ve směru objektu, jež analýzu provádí, tedy velikost vektoru v.

Získání času do předpokládané kolize

Pro běžné kolize v jízdních pruzích před vozidlem se používá výpočet relativní rychlosti k přizpůsobení rychlosti autonomního vozidla, případně k vyhodnocení, zda má vozidlo změnit jízdní pruh.

V okolí modelovaného systému se ale vyskytují i kolizní objekty, pohybující se mimo jízdní pruhy před vozidlem – například ve křižovatkách. Pro tuto situaci senzory modelovaného vozidla odhadují předpokládaný kolizní bod za pomoci průsečíku dvou přímek, tyto přímky směrem odpovídají vektoru rychlosti autonomního vozidla, pokud je nalezen bod jejich průsečíku, vypočítá se vzdálenost a čas, za který vozidla do bodu dorazí (s rezervou pro bezpečný průjezd).

K získání vzdálenosti vozidla od průsečíku je použita Pythagorova věta viz. 5.1.

Výpočet času do bodu se liší na základě akcelerace. V případě konstantní rychlosti, tzn. nulové akcelerace, lze vypočítat čas následovně:

$$t = s/v \tag{5.6}$$

Tento výpočet se v simulaci používá u objektů v okolí, neboť mají konstantní rychlost.

Ovšem pokud se objekt pohybuje rovnoměrně zrychleným pohybem viz. vzorec 2.1, je výpočet složitější, vysvětluje ho následující algoritmus:

Result: Čas do dosažení vzdálenosti
časDoMaxRychlosti = (maxRychlost - okamzitaRychlost) / akcelerace;
drahaDoMaxRychlosti = vypoctiDrahuDosazenouZaCas(pocatecniRychlost, zrychleni, cas);
if *drahaDoMaxRychlosti == vzdalenostBodu* **then**
  return *časDoMaxRychlosti*;
end
if *drahaDoMaxRychlosti < vzdalenostBodu* **then**
  return *časDoMaxRychlosti + ((vzdalenostBodu - drahaDoMaxRychlosti) / maxRychlost)*;
end
if *drahaDoMaxRychlosti > vzdalenostBodu* **then**
  return *vypoctiCasDosazeniVzdalenostiBehemAkcelerace(pocatecniRcychlost, zrychleni, draha)*;
end

Algoritmus 1: Výpočet času pro dosažení vzdálenosti do bodu zapsané v pseudokódu. Mohou nastat tři různé situace, vypočtená dráha ujetá vozidlem při snaze o dosažení maximální rychlosti se rovná vzdálenosti bodu, v tom případě vrátíme čas potřebný k dosažení maximální rychlosti. Složitější případ je, pokud je dráha větší nebo menší než vzdálenost bodu, poté musíme použít další matematické vzorce.

Nejprve musíme dopočítat dráhu do maximální rychlosti, proto potřebujeme zjistit, za jaký čas vozidlo takovou dráhu urazí, to uděláme pomocí vzorce 2.6. Tento čas dosadíme společně s počáteční rychlostí a zrychlením do následující rovnice, čímž získáme výslednou dráhu. Jedná se o rovnici 2.5 s přidanou počáteční rychlostí.

$$s = v_0 * t + \frac{1}{2} * a * t^2 \tag{5.7}$$

V prvním případě, kdy $drahaDoMaxRychlosti == vzdalenostBodu$ postačí vrátit čas potřebný k dosažení maximální rychlosti.

V případě, kdy $drahaDoMaxRychlosti < vzdalenostBodu$ potřebujeme dopočítat čas strávený na zbývající dráze. Po ujetí vzdálenosti odpovídající dosažení maximální rychlosti se rychlost vozidla ustálí, můžeme tedy použít vzorec 5.6 k dopočtení času stráveného na zbývajícím kusu dráhy. Tyto dva časy následně sečteme.

Ve třetím případě získáme výsledný čas použitím rovnice:

$$t = \frac{-v_0 + \sqrt{v_0^2 + 2 * a * s}}{a} \tag{5.8}$$

Jedná se o diskriminant z výše uvedené rovnice 5.7. Za s dosadíme vzdálenost bodu.

Rozhodnutí o změně jízdního pruhu

Aby vozidlo mohlo změnit jízdní pruh, musí nejprve vyhodnotit, zda má změna smysl a zda je možné ji provést bez kolize. To má na starosti metoda **getTurnDirection()** ve třídě

Vehicle. Tato metoda kontroluje, zda je možné odbočit a také, zda je to výhodné pro levý i pravý jízdní pruh, její funkcionalitu si ilustrujeme algoritmem pouze pro jeden z pruhů.

```
Funkce getTurnDirection():
    Result: Směr zatáčení vyjádřený číslem
    if vnějšíLevýJízdníPruh != null then
        if jeOdbočeníVlevoBezpečné(kontrolujPouzeOstatníPruhy,
          bezpecnaVzdalenost) a máOdbočeníSmysl(kolizníObjektyPředVozidlem,
          identifikátorVnějšíhoPruhu) then
            return -1;
    if vnějšíPravýJízdníPruh != null then
        if jeOdbočeníVpravoBezpečné(kontrolujPouzeOstatníPruhy,
          bezpecnaVzdalenost) a máOdbočeníSmysl(kolizníObjektyPředVozidlem,
          identifikátorVnějšíhoPruhu) then
            return 1;
    return 0;
```

Algoritmus 2: Metoda pro vyhodnocení směru odbočení zapsaná v pseudokódu. Rozhoduje o tom, zda bude vozidlo odbočovat do levého nebo pravého pruhu, případně zda vůbec má pruh měnit.

Zda je odbočení vlevo (nebo vpravo) bezpečné se vyhodnocuje na základě splnění následujících podmínek:

- Nejbližší zadní kolize je dále než stanovená bezpečná vzdálenost (výchozí hodnota je 10 metrů).
- Rychlost všech kolizních objektů za vozidlem je vyšší maximálně o předem stanovenou hodnotu (výchozí hodnota je 0.5 m/s, spolu s předchozím předpokladem dává vozidlu v ideálním případě prostor minimálně 20 sekund na odbočení)
- Nejsou přítomny boční kolize – tzn. prostor vedle vozidla je prázdný.
- Nejbližší přední kolize je alespoň 10 metrů vzdálená (délka dvě a půl vozidla, dostatek prostoru pro případný návrat do pruhu).

Zároveň s tím se kontroluje, jestli odbočení dává smysl následovně:

```
Funkce shouldChangeLane(kolizníObjektyPředVozidlem, cílovýPruh):
    Result: Vrací hodnotu true, pokud má odbočení smysl, jinak vrací false
    if jeDetekovánaPředníKolize then
        return kolizeJeVeStejnémPruhu a
          rychlostVCílovemPruhuJeVyššíNežVAktuálním a
          rychlostPředníKolizeJeNižšíNežVozidla
    if rychlostVCílovemPruhuJeVyššíNežVAktuálním a vozidloNeníVPravémPruhu
      then
        return true
    return false;
```

Algoritmus 3: Zápis metody v pseudokódu. Metoda vrací pravdivostní hodnotu true, pokud vyhodnotí, že změna pruhu dává smysl. Používá se, aby vozidlo neprovádělo zbytečné pokusy o změnu pruhu.

Detekce kolizí

Obecně je detekce kolizí definována v sekci 4.2. V simulačním modelu je realizována pomocí průsečíků přímek. Třída collisionObject obsahuje metodu **checkLineToLineCollision(line1, line2)**, která vrací JSON výstup obsahující následující data:

- výsledek – zda byla detekována kolize
- x,y – souřadnice průsečíku přímek
- vzdálenost – vzdálenost mezi počátkem první přímky a souřadnicemi průsečíku

Protože se k výpočtu používá průsečík přímek, ale jednotlivé detekční paprsky mají omezený dosah – jsou to úsečky – musí metoda před vyhodnocením výsledku ještě zkontrolovat, zda se průsečík nachází v dosahu senzoru.

Pokud se vozidlo nachází ve stavu zrychlování, nemusí samotná detekce kolize znamenat změnu stavu na zpomalování nebo odbočování. Pokud se kolize nenachází v kritické vzdálenosti a nebo nesplňuje některou z podmínek definovaných u dalších stavů, je ignorována.

Kritická vzdálenost pro to, aby byla kolize relevantní se skládá z brzdné dráhy a bezpečné vzdálenosti, kterou vozidlo udržuje. Brzdná dráha se neliší jen na základě okamžité rychlosti, ale také na základě okolních podmínek, viz. následující tabulka, která zobrazuje délku brzdné dráhy z rychlosti 130 km/h a velikost záporného zrychlení na základě okolních podmínek.

	Reakční dráha	**Brzdná dráha**	**Dráha zastavení**	**Zrychlení**
suchá silnice	36 m	93 m	129 m	-7 m/s
mokrá silnice	36 m	130 m	166 m	-5 m/s
náledí	36 m	435 m	471 m	-1.5 m/s

Tabulka 5.1: Tabulka zobrazující rozdílné délky brzdné dráhy z okamžité rychlosti 130 km/h na základě různých okolních podmínek. Mimo jiné uvádí hodnoty záporného zrychlení využitého v simulačním modelu. Autonomní vozidlo nepočítá s reakční dráhou, jeho brzdná dráha bude tedy o něco nižší, než u vozidla řízeného člověkem.

Udržování ve středu pruhu

Model vozidla obsahuje kromě senzorů sloužících k detekci kolizí, také senzory sloužící pro detekci jízdních pruhů. Jejich dosah je oproti ostatním senzorům snížený a rozsah je omezen na jeden paprsek, navíc dokáží rekonstruovat chvilkový výpadek – například chybějící značení cesty.

Mimo ostatní účely tyhle senzory slouží k udržení vozidla uprostřed jízdního pruhu. Vozidlo spočítá vzdálenost od jízdního pruhu na obou stranách, přičemž prioritně získává data ze senzorů směřujících mírně před vozidlo, aby se dokázalo vyrovnat i s případnou zatáčkou. Poté vyhodnotí rozdíl vzdáleností, z něhož se vypočítá úhel otáčení za pomoci předdefinovaného koeficientu a započne otáčení ve směru vzdálenějšího jízdního pruhu. Otáčení je omezeno maximálním úhlem natočení předního kola. Metoda je dostupná v kódu autonomního vozidla pod názvem **balanceDistanceBetweenLanes()**.

5.3 Simulace

Kromě implementace samotného simulačního modelu jsem implementoval také samotné simulační prostředí, na které se zaměřuje tato sekce. Implementace simulačního prostředí byla z důvodu zvolených nástrojů nezbytná.

Řízení simulace

O řízení celé simulace se stará logika uvnitř souboru **main.js**, ten obsahuje následující metody:

- **init** – metoda zodpovědná za přípravu scénáře, vytvoření a inicializaci autonomního vozidla a také spuštění samotné simulace. Metoda je zavolána po načtení HTML dokumentu index.html, který tvoří základ aplikace.
- **initializeCanvas** – metoda volaná z init metody. Má na starosti vytvoření grafického prostředí pro vykreslování za pomoci HTML5 canvas.
- **redraw** – tato metoda vykresluje všechny vizuální prvky simulace a řídí také simulační čas, proto se zde volá také logika obsluhující generování okolních objektů a zpracování data kolektorů. Po vykreslení všech objektů nastaví časovač pro své opětovné zavolání.

Simulace se neřídí podle reálného času, ale podle simulačního, který se inkrementuje o předem definovaný krok po každém volání metody redraw. Tento simulační čas slouží k synchronizaci všech akcí provedených v rámci simulace, ale i akcí v rámci pomocných metod jako například sběr informací nebo generování prostředí.

Simulační čas je pro účely sledování vizuálního výstupu možné přizpůsobit reálnému času pomocí přepínače v metodě **init** nazvaného „useRealTime“.

Simulační prostředí je nastaveno tak, aby po dosažení ukončovací podmínky byla celá simulace restartována, opět s náhodným výběrem prostředí a proměnlivých hodnot u objektů okolního systému – to napomáhá jednoduššímu sběru dostatečného množství informací.

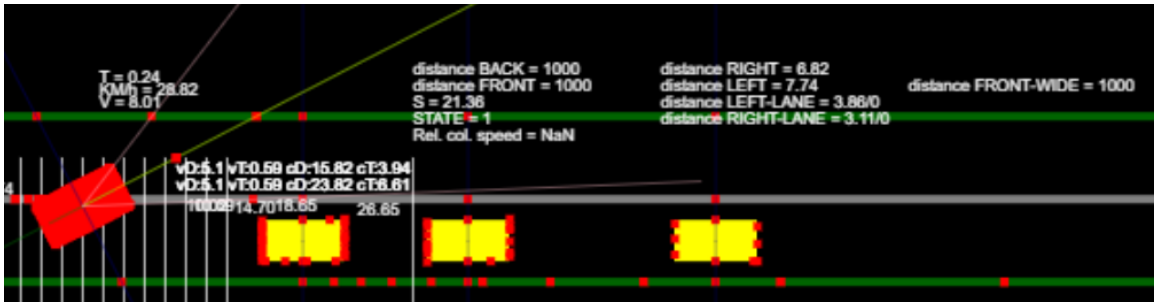
5.4 Sledování průběhu simulace

Jak jsem již zmínil v předchozí podkapitole, v pravidelných intervalech jsou vykreslovány vizuální prvky simulace. Tyto vizuální prvky nejsou pouze výpisy o stavu simulace a podobně, nýbrž se jedná o kompletní prostředí zahrnující simulační model autonomního vozidla, senzory i okolí vozidla – ostatní vozidla a jízdní pruhy.

Tuto funkcionalitu je možné využít nejen pro účely ladění, ale také pro lepší představu o tom, co se v průběhu simulace děje, získání přehledu o tom, jak vozidlo reaguje a podobně. Vizualizace je zobrazena v následujících podsekcích, model autonomního vozidla je zvýrazněn červenou barvou.

Informační výpisy

Vizuální zobrazení simulace poskytuje několik druhů výpisů důležitých proměnných pro případnou kontrolu validity modelu, výpisy jsou viditelné na obrázku:



Obrázek 5.3: Výstřižek ze simulačního prostředí. Zobrazuje tři druhy informačních výpisů přítomných pro manuální validaci a verifikaci modelu.

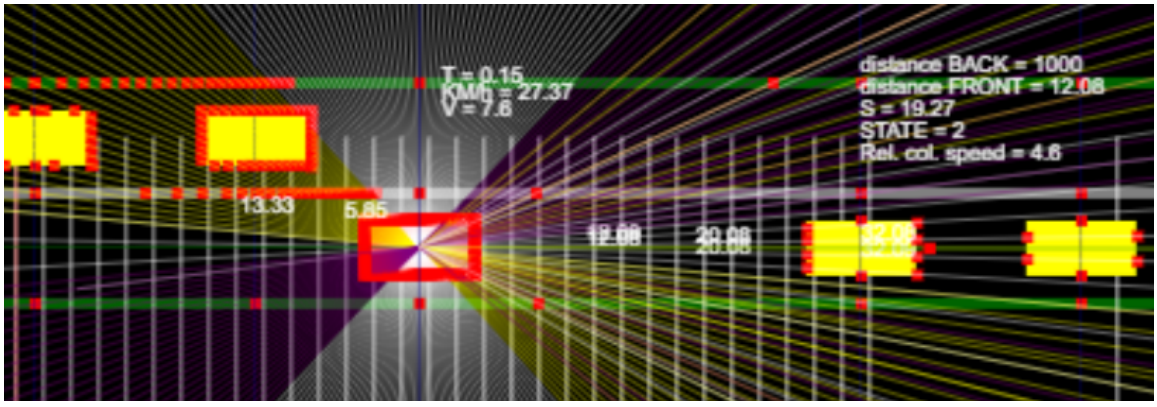
Výpisy ve vrchní části okna aplikace obsahují informace relevantní k internímu stavu vozidla. Zobrazují simulační čas, rychlost v kilometrech za hodinu i metrech za sekundu, vzdálenosti od jízdních pruhů, vzdálenosti od nejbližších kolizí, brzdnou dráhu a relativní rychlost kolizního objektu před vozidlem.

K dispozici jsou také výpisy v rámci modelu senzoru, ty se zobrazují mezi modelem autonomního vozidla a případným kolizním objektem a určují nejbližší vzdálenost kolizního objektu.

Poslední typ informačního výpisu je výpis na pozici průsečíku drah objektů – v tomto případě autonomního vozidla a jiného kolizního objektu. Tento výpis zobrazuje vzdálenost a čas autonomního vozidla k průsečíku, vzdálenost kolizního objektu od průsečíku a čas kolizního objektu potřebný k dosažení průsečíku drah.

Senzory

Pro lepší přehled o průběhu simulace je možné zapnout i vizualizaci senzorů. Senzory obsahují dva vizualizační prvky: úsečku a čtverec.



Obrázek 5.4: Výstřižek ze simulačního prostředí. Zobrazuje dva vizualizační prvky senzorů, úsečky a čtverce. Úsečky označují paprsky vyslané senzoru, červené čtverce průsečíky paprsků s kolizními objekty.

Úsečky slouží pro zobrazení zorného pole senzoru, jsou barevně odlišeny podle typu senzoru a části zorného pole – senzory mohou být rozděleny na tři části pro účely detekce kolizí. Je také možné zapnout zvýraznění jednotlivých úseček pokud dojde k detekci kolize.

Červené čtverce zvýrazňují jednotlivé průsečíky přímek mezi paprskem senzoru a kolizním objektem.

5.5 Sběr informací

Během simulace probíhá sběr informací, které se následně používají pro vyhodnocování úspěšnosti scénářů. Tato sekce se věnuje právě nastavení sběru informací a krátkému vysvětlení funkcionality sběru.

Nastavení monitorování

Informace získané ze simulace by měly být vždy relevantní a užitečné. To může znamenat, že bude potřeba v různých situacích a scénářích sbírat rozdílné informace.

Proto jsem implementoval možnost nastavení takzvaných **data kolektorů**. Nastavení kolektoru je metoda, která vrací JSON objekt složený ze dvou hlavních prvků: *name*, neboli jméno objektu, pod kterým bude každý záznam kolektoru uložený v celkovém exportu a *value*, což je objekt obsahující libovolné hodnoty.

Ke sběru informací z kolektorů dochází pravidelně v předdefinovaném intervalu, výchozí hodnota je 0.5 sekundy v simulačním čase. Při každém sběru informací se provolají všechny kolektory, je tedy možné nastavit kolektorů více, jako vidíme na uvedeném příkladu:

```
analyzeExporter.setStopCondition(
    function () {
    return simulationTime > 60
    });
// nastaveni ukoncujici podminky

analyzeExporter.addDataCollector(function ()
{
    return {
        name: watchedVehicle.id,
        value: {
            x: watchedVehicle.x,
            y: watchedVehicle.y,
            simulationTime: simulationTime,
            turnDirection: watchedVehicle.turnDirection,
            turningState: watchedVehicle.turningState
        },
    }
});

analyzeExporter.addDataCollector(function ()
{
    return {
        name: "autonomous-additional",
        value: {
```

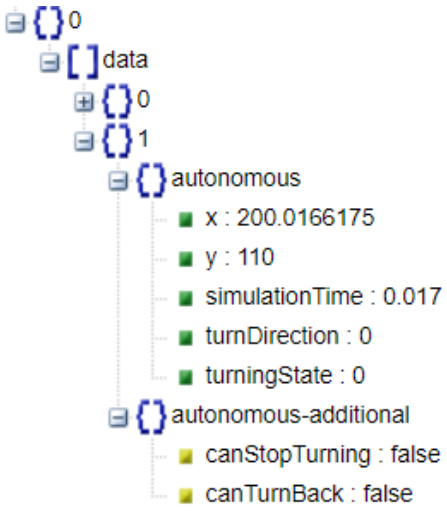
```
            canStopTurning: watchedVehicle.canStopTurning(),
            canTurnBack: watchedVehicle.canTurnBack()
        },
    }
});
```

Výpis 5.1: Ukázka nastavení sběračů dat pro získávání informací během simulace. První řádek ukazuje jakým způsobem lze nastavit ukončující podmínku pro simulaci.

Při vytváření objektu obsahujícího hodnotu kolektoru lze použít libovolné metody, ty se vyhodnotí při vyvolání sběru informací a jejich hodnota se uloží ve vyhodnocené podobě do kolekce určené pro finální export získaných informací, není tedy třeba mít obavy z případného vyhodnocení metod až před ukončením simulace, což by mohlo způsobit problémy v případě, že by metody obsahovaly referenci na objekty.

Ukázka výstupních dat

Výstupní data jsou kolekcí všech jednotlivých záznamů ze všech nastavených kolektorů pro daný scénář. Pokud předpokládáme nastavení kolektorů stejně jako ve výše uvedeném příkladu, bude struktura informací uložená ve výsledném souboru vypadat následovně:



Obrázek 5.5: Zobrazení datové struktury exportovaných informací z dokončené simulace. Vybrány pouze dva záznamy, reálně bude záznamů desítky z jediné simulace. Na obrázku lze vidět jakým způsobem se ukládají data jednotlivých kolektorů do finální struktury uložené v souboru.[2]

Ukládání informací

Posbírané informace se po dosažení ukončující podmínky simulace automaticky exportují do předem definovaného adresáře – ve výchozím nastavení je to adresář “CollectedData”

[2] Vygenerováno nástrojem: http://jsonviewer.stack.hu

s názvem podle aktuálního scénáře simulace. Ukládají se dvě verze souboru, jeden podrobný pro zpětnou analýzu informací. Ten obsahuje informace ze všech nastavených data kolektorů. Druhá verze souboru obsahuje pouze informace o autonomním vozidle a to ve formě jednorozměrných polí. K tomu ještě obsahuje základní informace o nastavení a průběhu simulace.

Při ukládání bylo z důvodu souběžného běhu několika simulačních instancí ve stejný čas nutno řešit také výhradní přístup k zápisu do souboru. I přesto, že se každý scénář ukládá do vlastního souboru docházelo občas k přepsání souborů, pokud se ve stejný čas spustilo více instancí se stejným scénářem. Příčinou bylo využívání stejného file descriptoru. Pokud některý běžící proces již inkrementoval offset file descriptoru, druhý proces uložil nová data na pozici daného offsetu, tím mohl vzniknout poškozený JSON a nebo úplně prázdný soubor, pokud k tomu došlo již při načítání souboru.

Z toho důvodu jsem se rozhodl využít rozšíření prostředí Node.js - “proper-lockfile” v kombinaci s “fs-extra”. Tyto dva moduly umožňují soubor při zápisu uzamknout a také kontrolovat, zda je soubor, ke kterému přistupuji uzamknutý. Pokud tedy nastane výše zmíněná situace, nedojde k poškození posbíraných dat.

5.6 Shrnutí

Tato kapitola se zabývala popisem struktury projektu, detailnějším popisem důležitých metod a použitých vzorců v rámci simulačního modelu. Také se zaměřila na popis simulačního prostředí a nastavení monitorování simulačních běhů.

Kapitola 6

Vyhodnocení modelu

Kromě simulačního prostředí je k dispozici také nástroj pro analýzu získaných informací. Ten zpracovává data ve formátu JSON, uložené v souborech se suffixem "–autonomousData" a zobrazuje je za pomoci knihovny Plotly.js viz. podkapitola 3.2. Použití nástroje bude vysvětleno v této kapitole. Následně bude provedena analýza získaných dat s využitím tohoto nástroje.

6.1 Nastavení nástroje

V horní sekci nástroje se nachází políčka pro vložení a úpravu vstupních dat. Mimo možnost "omezit na trace" vyžadují opětovné zpracování vstupních dat kliknutím na tlačítko „Zpracovat".

Vstupní data – vstup pro vložení souboru s JSON výstupem ze simulací.

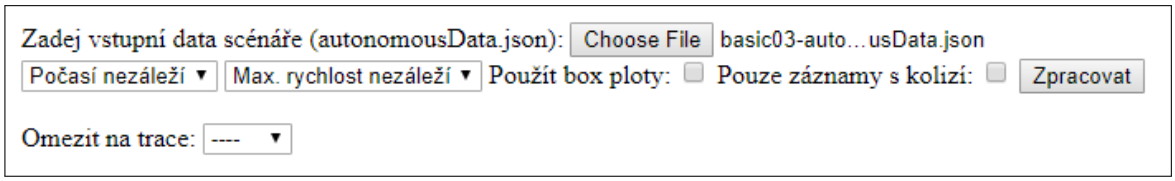
Počasí – omezí vstupní data pouze na vybrané počasí.

Max. rychlost – omezí vstupní data na vybranou maximální rychlost.

Použít box ploty – místo spojnicového grafu zobrazí graf s box ploty.

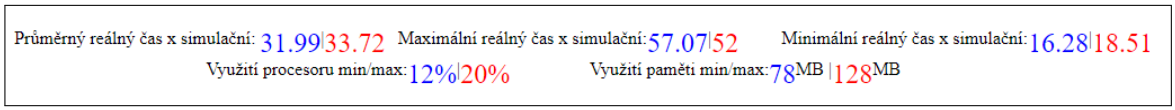
Pouze záznamy s kolizí – omezí vstupní data na ty, které obsahují alespoň jednu potencionální kolizi.

Omezit na trace – upraví zobrazení grafu pouze na jeden vybraný záznam ze vstupního souboru, výběr je ovlivněn dalšími nastaveními (počasí, rychlost, ...)



Obrázek 6.1: Vkládání vstupních hodnot a nastavení filtrů nad vstupními hodnotami. Pro aplikaci filtrů je třeba vstup znovu zpracovat pomocí tlačítka Zpracovat.

Pod nastavením nástroje se nachází stručný souhrn informací o provedených simulacích. Zobrazuje rozdíly mezi simulačním a reálným časem, paměťovou náročnost a využití procesoru.



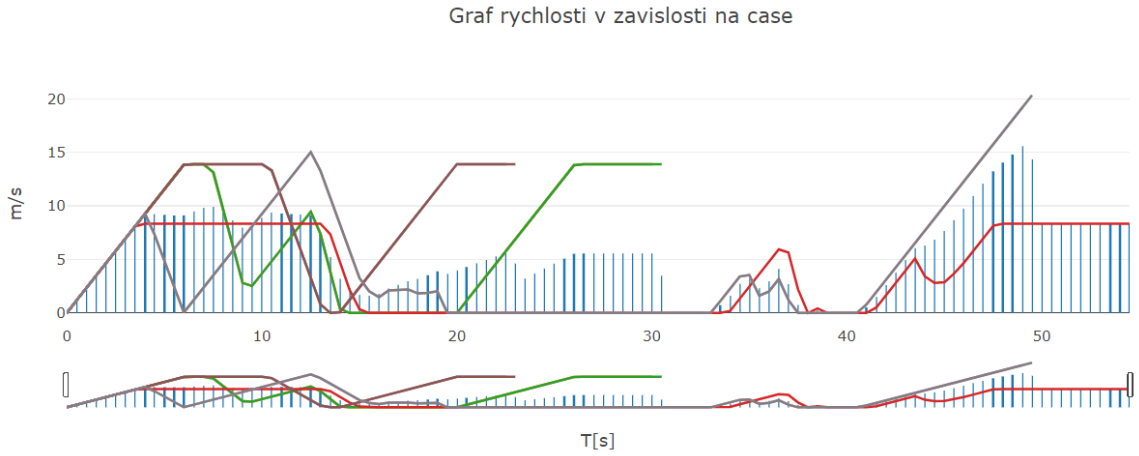
Obrázek 6.2: Zobrazení souhrnu základních informací o simulacích v nástroji pro analýzu výstupních dat ze simulací.

6.2 Výstup nástroje

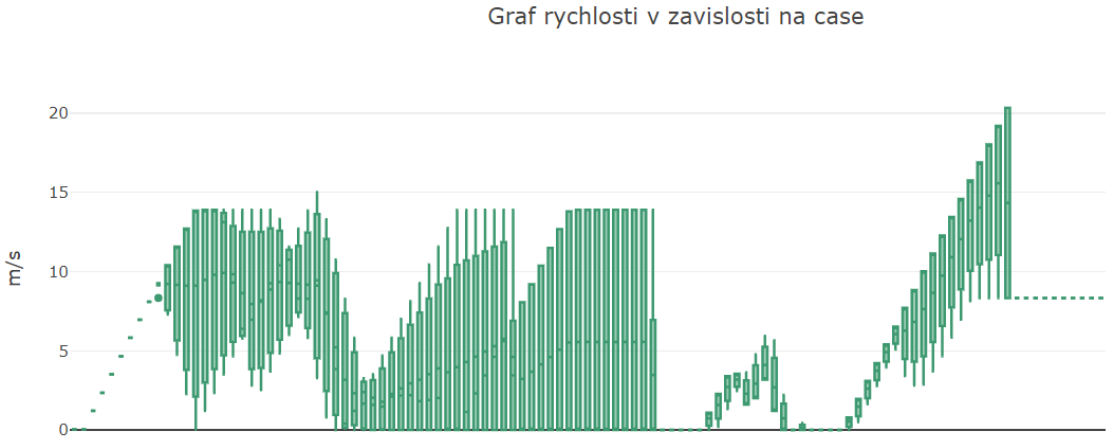
Po zpracování vstupních dat se zobrazí výstup ve formě grafů. S grafy lze dále interaktivně pracovat, nabízí například následující možnosti:

- Zobrazení detailů o jednotlivých záznamech
- Přiblížení a posun v grafu
- Uložení grafu ve formátu PDF

V případě zobrazení spojnicového grafu je na pozadí přidán také sloupcový graf zobrazující průměrnou rychlost všech běhů simulace. Kopie grafu pod osou zobrazující čas slouží pro výběr oblasti, kterou chceme zobrazit, je možné ji tažením myši omezit.



Obrázek 6.3: Vizualizace výstupních dat pomocí knihovny Plotly.js. Na obrázku lze vidět spojnicový graf, barevně rozlišené spojnice označují jednotlivé záznamy simulace. V pozadí je vidět také sloupcový graf zobrazující průměrnou rychlost.



Obrázek 6.4: Zobrazení výstupních dat pomocí knihovny Plotly.js. Jedná se o graf formou tzv. box plotů, které obsahují informace o průměrné hodnotě, maximální a minimální hodnotě, hodnotě prvního a třetího kvartálu i mediánu.

6.3 Vyhodnocení jednotlivých scénářů

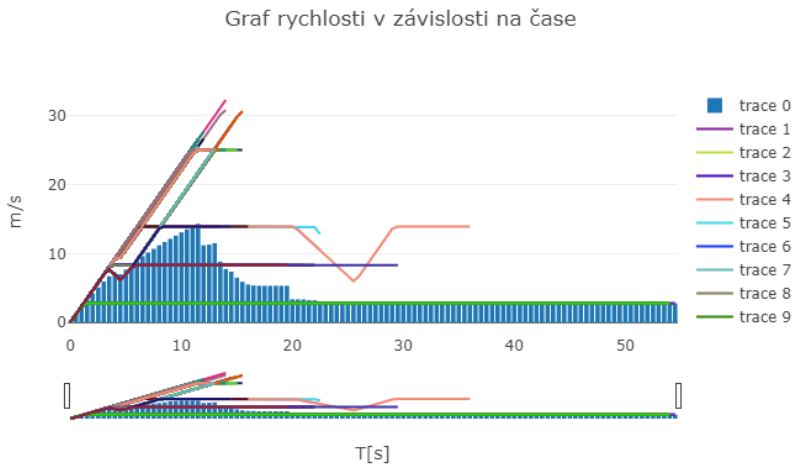
Simulace obsahuje celkem pět scénářů, na kterých byly prováděny a vyhodnocovány testy vozidla. První dva scénáře obsahují jen základní situace bez náhodných veličin, proto je v této sekci uveden podrobněji pouze druhý scénář. Další tři scénáře se již soustředí na složitější situace včetně náhodných veličin.

2. scénář

První dva scénáře obsahují jednoduché situace pro ověření validity základních vlastností vozidla – zrychlení, brzdné dráhy, otáčení. Z toho důvodu není v těchto simulacích zavedeno náhodné chování okolních prvků a jejich výsledky se liší víceméně jen maximální dosaženou rychlostí z důvodu omezení rychlosti. Níže je zobrazen druhý scénář na obrázku.

Obrázek 6.5: Ukázka druhého simulačního scénáře. V tomto scénáři je modelována situace, kdy vozidlo může na základě jeho rychlosti setrvat ve svém jízdním pruhu a nebo změnit pruh. Při změně pruhu musí kontrolovat, zda svou akcí neohrozí bezpečnost vozidel v cílovém jízdním pruhu.

Vzhledem k povaze těchto dvou scénářů je níže uveden pouze graf rychlosti druhého scénáře pro ilustraci jak vypadají data ze simulace bez náhodného chování. Tato simulace proběhla celkem 1231-krát, bez jakékoli kolize nebo odchylky od očekávané trasy.



Obrázek 6.6: Graf rychlosti v závislosti na čase zobrazující data získaná z druhého scénáře. Tento scénář neobsahuje náhodné prvky, proto je graf ustálený a hodnoty se liší pouze na základě rychlostního omezení. Sloupcový graf na pozadí zobrazuje průměrnou rychlost ze všech běhů simulace.

3. scénář

Ve třetím scénáři je zavedena náhodná rychlost u objektu před autonomním vozidlem v druhém pruhu. Tento scénář simuluje situaci, kdy vozidlo zpočátku nemá dostatek prostoru ke změně pruhu, ale objekt v druhém pruhu v určitém okamžiku zvýší rychlost a uvolní tak prostor autonomnímu vozidlu. Zbývající objekty mají stálou rychlost.

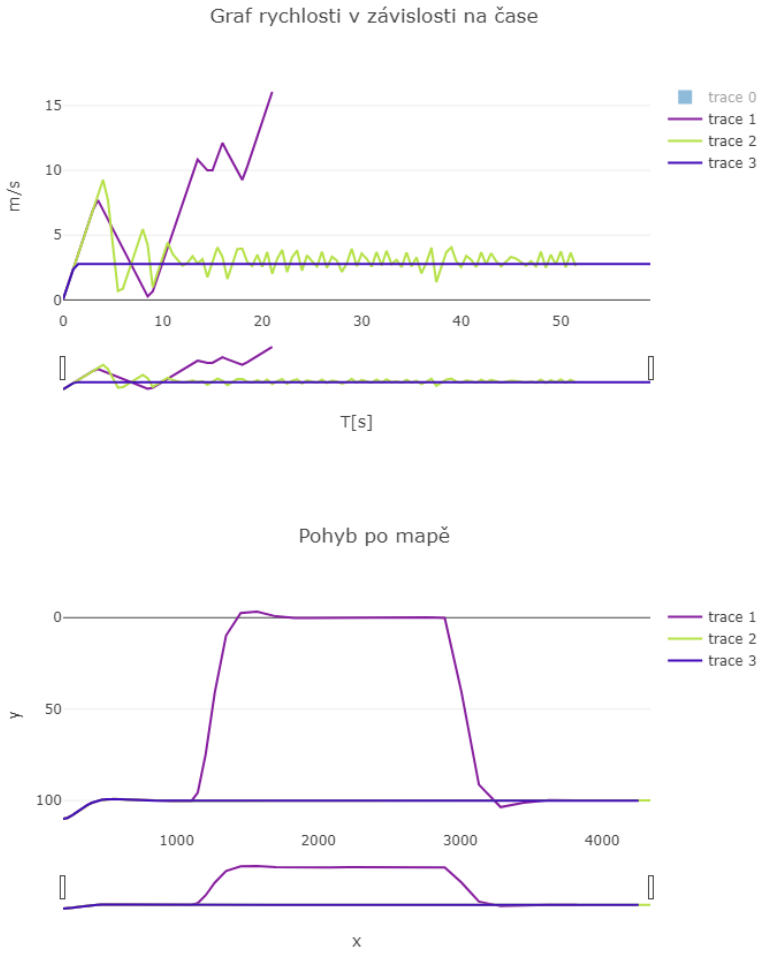
Obrázek 6.7: Ukázka třetího simulačního scénáře. V tomto scénáři je modelována situace, kdy vozidlo musí mimo kontroly blížících se objektů za ním, kontrolovat také dostatečný prostor v pruhu, kam by mohlo odbočit. Pokud nemá smysl odbočovat (tzn. rychlost vozidla v dalším pruhu je stejná nebo menší, nebo není dostatečně daleko), mělo by vozidlo setrvat ve stávajícím pruhu.

Grafy této simulace jsou o něco zajímavější. Pro tuto simulaci bylo provedeno celkem 2685 simulačních běhů, během kterých nebyla zaznamenána žádná kolize. Níže uvedené grafy jsou omezené pouze na 3 simulační běhy z důvodu přehlednosti. Zobrazují chování vozidla v následujících situacích:

1. Autonomní vozidlo má vyšší rychlost než objekt před ním, současně dojde k dostatečnému zvýšení rychlosti objektu ve druhém jízdním pruhu
2. Autonomní vozidlo má vyšší rychlost než objekt před ním, ale objekt ve druhém jízdním pruhu nezíská dostatečnou rychlost, aby měla změna pruhu smysl

3. Autonomní vozidlo má nižší rychlost než objekt před ním

První graf zachycuje okamžitou rychlost vozidel v závislosti na čase a na druhém grafu jsou tyto situace zachyceny v podobně bodů, ve kterých se vozidlo vyskytovalo, lze je porovnat s obrázkem 6.7, hodnota 100 na ose y značí střed původního jízdního pruhu, hodnota 0 poté střed druhého jízdního pruhu.



Obrázek 6.8: Ukázka výstupu ze třetího simulačního scénáře. V tomto scénáři je modelována situace, kdy vozidlo musí mimo kontroly blížících se objektů za ním, kontrolovat také dostatečný prostor v pruhu, kam by mohlo odbočit. Pokud nemá smysl odbočovat (tzn. rychlost vozidla v dalším pruhu je stejná nebo menší, nebo není dostatečně daleko), mělo by vozidlo setrvat ve stávajícím pruhu. Tyto situace jsou zobrazeny na vybraných třech záznamech.

První situaci lze pozorovat v grafech pod označením „trace 1“. Vozidlo z počátku zrychluje, následně vyhodnotí, že pro vyhnutí se kolizi musí začít brzdit a ve druhém pruhu ještě není dostatek prostoru. Když se ve druhém pruhu uvolní dostatek prostoru, vozidlo zároveň se zrychlováním změní pruh (to lze vidět v prvním grafu po 10 sekundě, ve druhém grafu cca v bodě 1100 na ose x). Od 13. do 18. sekundy simulace se pohybuje za objektem ve

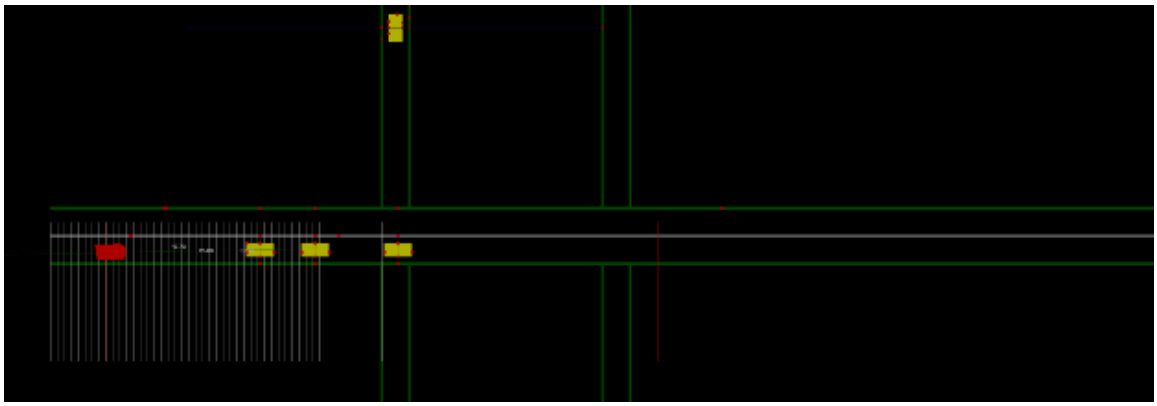
druhém jízdním pruhu, poté se dostane před skupinu objektů (vozidel) v prvním pruhu a vrátí se do něj, protože je to výhodné – nejsou tam další blokující objekty.

Druhá situace je označena v grafech jako „trace 2“. Lze pozorovat, že vozidlo nabere rychlost a zpomalí podobně jako v první situaci, nenásleduje pak ale změna pruhu, protože objekt v druhém pruhu neuvolní dostatek prostoru. Na grafu jsou vidět výkyvy zrychlení, to je způsobeno pokusy o získání rychlosti, které jsou následně blokovány kolizemi.

Třetí situace, v grafech jako „trace 3“, zobrazuje situaci, kdy je vozidlo omezeno maximální rychlostí nižší, než je rychlost objektů před ním. Vozidlo tak zrychlí na svou nejvyšší rychlost a pokračuje stejnou rychlostí až do konce scénáře.

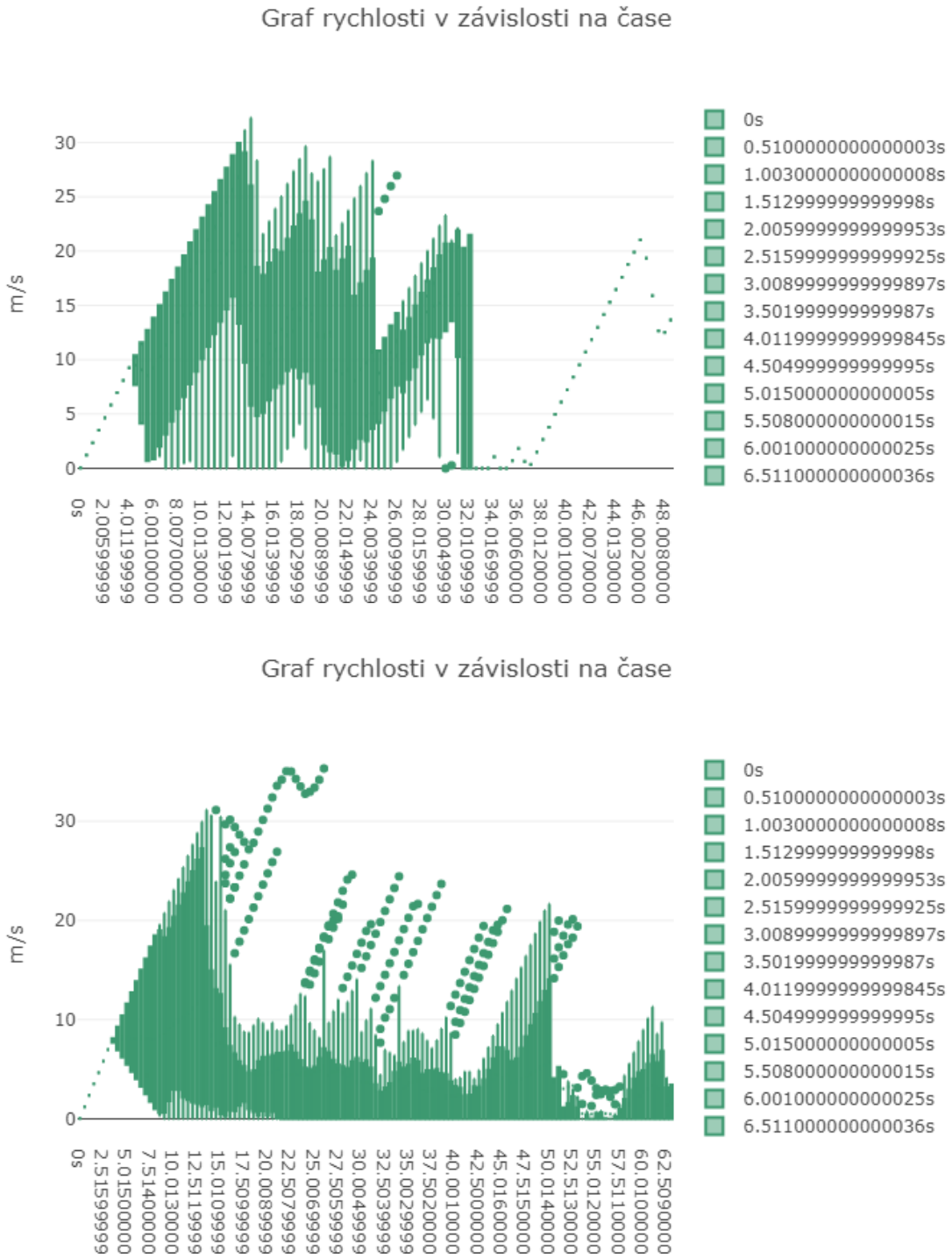
4. scénář

Čtvrtý scénář obsahuje kolonu vozidel jedoucích na počátku simulace před autonomním vozidlem. Tato kolona vozidel má stálou rychlost a rozestupy. Ze stran přijíždějí do trasy vozidla, která jsou generována v náhodných časech a s náhodnou rychlostí. Tato vozidla jsou potencionální riziko a autonomní vozidlo by mělo podle toho adekvátně zareagovat.



Obrázek 6.9: Simulační prostředí čtvrtého scénáře. Tento scénář modeluje situaci, kdy vozidlo musí vyhodnocovat případné kolize pocházející z křižovatek a zároveň reagovat na vozidla jedoucí před ním.

Celkem bylo analyzováno 2579 simulačních běhů s tímto scénářem. Níže uvedené grafy porovnávají rozdíly rychlosti mezi simulacemi na suché silnici a náledí v rychlosti 130 km/h, lze na nich pozorovat následky rozvážnějšího chování vozidla z důvodu nižšího brzdného účinku a tím pádem delší brzdné dráhy.



Obrázek 6.10: Porovnání výsledků simulace vozidla s rychlostí omezenou na 130 km/h s využitím box plotů. První graf zobrazuje výsledky na suché silnici, druhý na náledí. Na výsledcích lze vidět, že v případě suché silnice dosáhlo vozidlo cíle ve většině případů do 30-té sekundy simulace. V případě náledí, z důvodu delší brzdné dráhy a rozvážnějšího řízení, může být simulace ukončena až časovým limitem.

Ve 14 % zaznamenaných simulačních běhů se vyskytla kolize. U záznamů s kolizí v globálním hledisku nezáleželo na počasí, ale spíše na rychlostním omezení, 36 % záznamů s kolizí bylo s omezením na maximální rychlost 10 km/h, zbytek je rozložen rovnoměrně. Po hlubší analýze posbíraných dat jsem zjistil následující důvody kolizí:

- Autonomní vozidlo se rozhodlo pro průjezd křižovatkou ve chvíli kdy to bylo bezpečné. Než stihlo průjezd dokončit, v okolí se vygenerovalo vozidlo s vysokou rychlostí. Z důvodu nedostatečné rychlosti autonomního vozidla, nemělo vozidlo dostatek času na průjezd, rozhodlo se tedy zvolit cestu minimalizaci rizik a zpomalit.
- Autonomní vozidlo nemělo dostatečnou rychlost pro změnu pruhu a nebo nebyl dostupný volný pruh. Z toho důvodu se drželo za kolonou vozidel. Když došlo k vygenerování vozidla s vysokou rychlostí v blízkém okolí, nemělo již dostatek času na zastavení, a nemohlo ani zrychlit, protože před ním byla kolona vozidel.
- Autonomní vozidlo se zařadilo před kolonu vozidel příliš brzy. Když se před ním vygenerovalo větší množství vozidel projíždějících křižovatkou, stálo na místě tak dlouho, až do něj nabourala kolona vozidel.

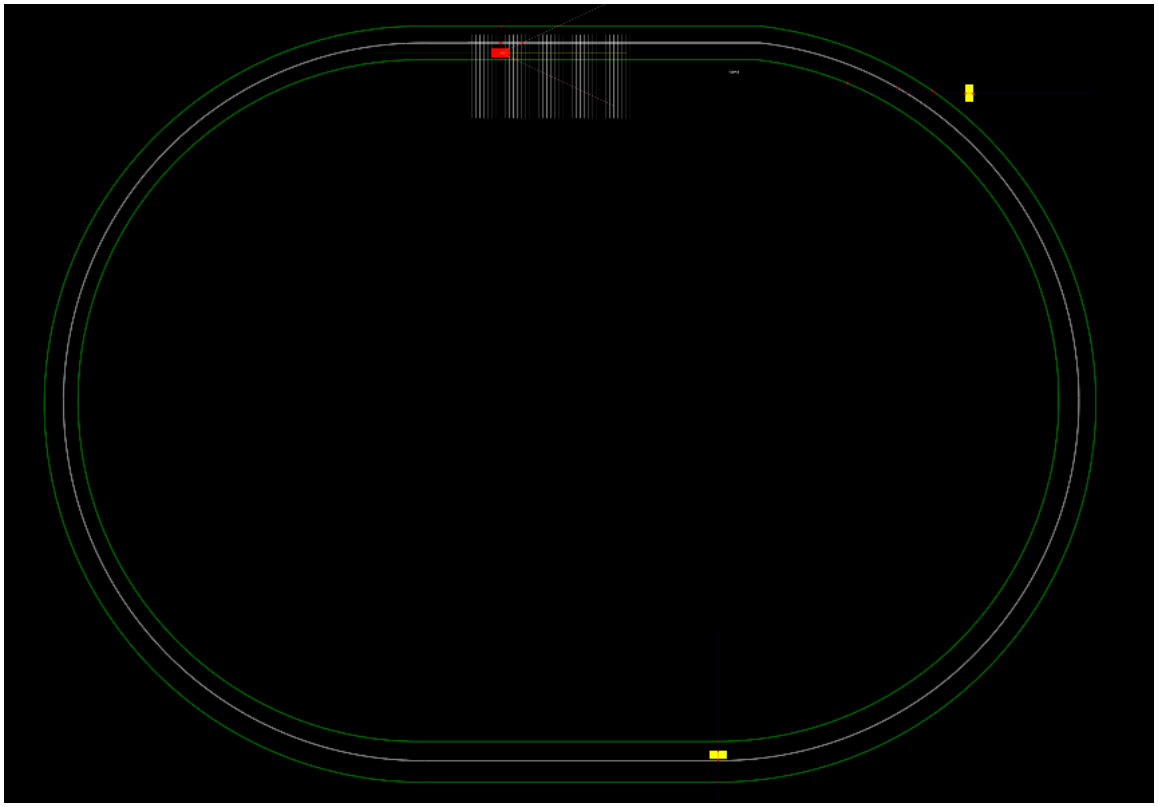
K žádné z výše uvedených situací by nemuselo dojít, pokud by vozidla v okolí reagovaly na autonomní vozidlo v jejich trase. V uvedených situacích již nemělo autonomní vozidlo další možnosti. V ojedinělých případech se vozidlo při vyhýbání kolizi dostalo mimo svou dráhu s následným návratem zpět (méně než 1 % případů).

5. scénář

Poslední scénář modeluje uzavřený okruh, vozidlo je zde na rozdíl od ostatních scénářů omezeno pouze časovým limitem. Tento scénář obsahuje dva body, ve kterých se generují vozidla, viz. obrázek 6.11.

Žluté vozidlo v pravém horním rohu představuje situaci neočekávané kolize. Generuje se v takovém místě, kde vozidlo při vjezdu do zatáčky nevidí, navíc s malým odstupem od silnice. U této situace nepředpokládám vysokou úspěšnost, vytvořil jsem ji, abych zjistil jak rychle zvládne model zareagovat a v jaké rychlosti dojde ke kolizi. Tato vozidla jsou generována s náhodnou rychlostí a rozestupy.

Druhé žluté vozidlo ve spodní části simulace má „nepředvídatelné“ chování. Jeho rychlost i rozestupy generování jsou náhodné, navíc ale napodobuje funkcionalitu autonomního vozidla pro udržování se v jízdním pruhu – s tím rozdílem, že pravidelně přesahuje hranici svého jízdního pruhu a může z něj i vybočit.

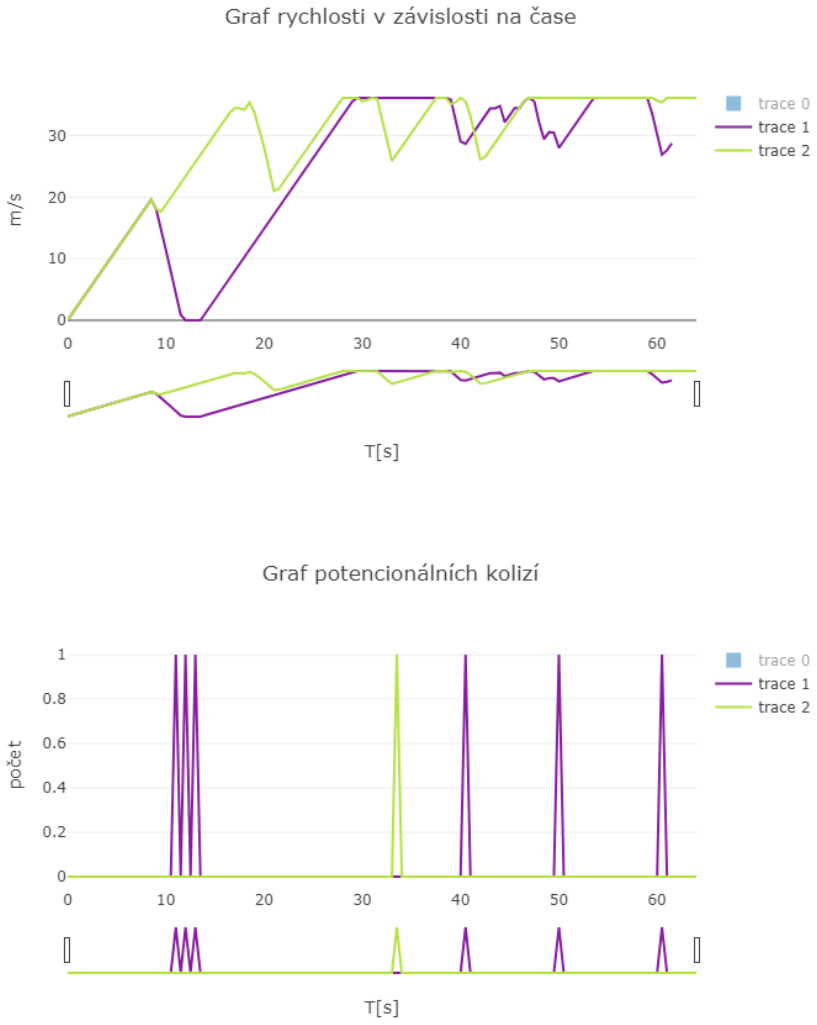


Obrázek 6.11: Ukázka pátého simulačního scénáře. V tomto scénáři se vozidlo vyskytuje na uzavřeném okruhu a do cesty mu přichází kolizní objekty. První druh kolizního objektu se generuje příliš blízko jízdního pruhu, navíc v nepřehledné zatáčce do které autonomní vozidlo s předstihem nevidí. Druhý typ kolizního objektu náhodně mění pruhy a tím nutí autonomní vozidlo reagovat změnou pruhu nebo zpomalením.

Celkem bylo analyzováno 1750 záznamů, z toho bylo 72 % záznamů s kolizí. Podle předpokladu byl hojný výskyt kolizí v místě prvního náhodného objektu. Počet kolizí narůstá s maximální rychlostí autonomního vozidla, zatímco v simulačních bězích omezených na 10 km/h se vyskytuje 14 % ze všech kolizí, ve 130 km/h je to 26 %. Podle počasí jsou kolize rozděleny následovně:

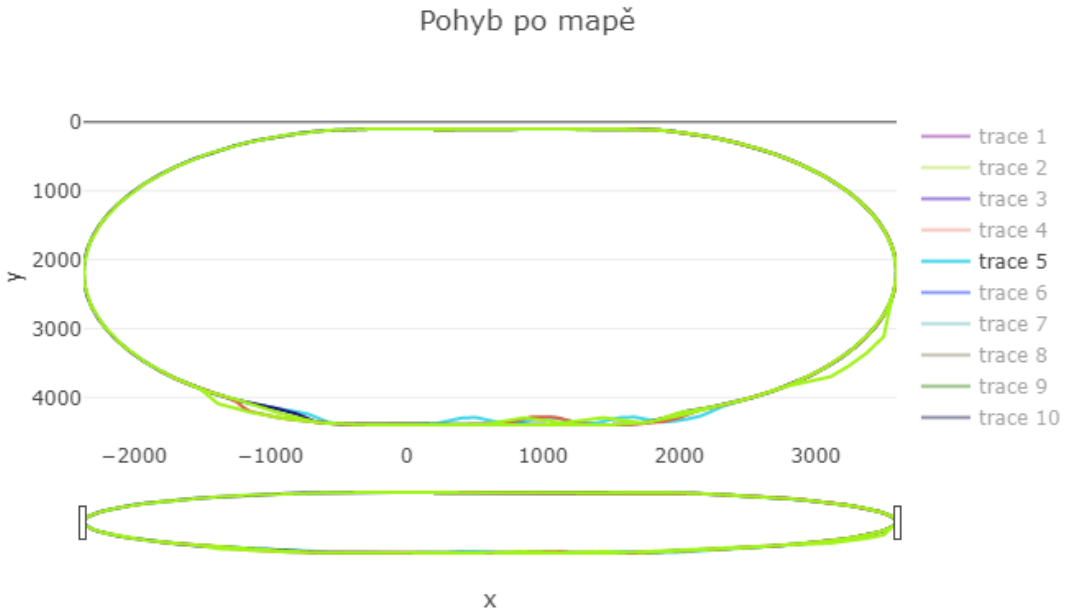
- Suchá vozovka – 29.4%
- Mokrá vozovka – 32.8%
- Náledí – 37.8%

Na grafech uvedených níže jsou zobrazeny dva simulační běhy s omezením maximální rychlosti na 130 km/h a se suchou vozovkou. Tyto běhy zobrazují kolize s objektem v pravém horním rohu.



Obrázek 6.12: Porovnání dvou simulační běhů s omezením maximální rychlosti na 130 km/h a se suchou vozovkou. Na běhu označeném jako trace 1 lze vidět, že vozidlu se nepodařilo kolizi předejít ani v jednom případě. Pokaždé se pokusilo zpomalit, ale bylo již příliš pozdě. Oproti tomu se vozidlu v běhu označeném jako trace 2 nepodařilo zamezit pouze jedné kolizi a to v čase 33 vteřin po začátku simulace. Tento rozdíl je dán pozicí objektu při vjezdu do zatáčky.

V místě generování druhého kolizního objektu ke kolizím došlo ve 25 % případů, a to za situace, kdy objekt s náhodným chováním změnil pruh těsně před vozidlem. V méně než 1 % případů vozidlo reagovalo na nenadálou kolizi vyjetím ze silnice. Ve zbylých případech se úspěšně vyhýbalo nečekaným změnám směru vozidla, to lze vidět na níže uvedeném grafu.



Obrázek 6.13: Graf zobrazující změnu pozice vozidla v reakci na nečekané změny směru vygenerovaného kolizního objektu. Lze pozorovat úhybné manévry provedené autonomním vozidlem.

6.4 Shrnutí

Tato kapitola provedla čtenáře základním nastavením nástroje sloužícího pro analýzu získaných dat. Následně bylo provedeno vyhodnocení testovaných scénářů a byly zde popsány také situace, které během simulace jednotlivých scénářů nastávaly. Testování dopadlo podle předpokladů, níže je uvedena souhrnná tabulka provedených simulací.

Scénář	**Provedené simulace**	**Simulace s kolizí**	**Kolize na suché silnici**	**Kolize na mokré silnici**	**Kolize na náledí**
1	955	-	-	-	-
2	1231	-	-	-	-
3	2685	-	-	-	-
4	2579	368	127	131	110
5	1750	1258	370	413	475

Tabulka 6.1: Souhrnná tabulka jednotlivých simulačních běhů a detekovaných kolizí za různých podmínek. Zobrazuje rozdíly mezi různými druhy počasí. V průběhu testování bylo získáno vzorků ještě více.

Kapitola 7

Závěr

Cílem této práce bylo analyzovat požadavky, problémy, pojmy a principy související s autonomními vozidly a následně vytvořit výpočetní model chování a rozhodování autonomního vozidla.

Záměr práce byl naplněn navržením a implementací modelu v jazyku JavaScript, model byl validován oproti poznatkům o fyzikálních vlastnostech vozidel a testován, zda jeho chování odpovídá předpokladům, které jsem v řešení práce určil.

Práce nabízí také simulační prostředí, ve kterém lze model testovat, sbírat informace o jeho chování a také sledovat průběh simulačních běhů.

Získané informace jsou exportovány do souborů a je možné nad nimi provádět analýzu pomocí nástroje, vytvořeného k tomuto účelu. Popis práce s tímto nástrojem je taktéž obsažen v práci.

Simulační model byl vsazen do pěti různých scénářů. Tyto scénáře modelují situace, kdy se vozidlo vyskytuje na dvouproudé silnici a do cesty mu přichází kolizní objekty (vozidla). Vozidlo musí správně reagovat na objekty před ním, za ním i objekty, které by se mohly vyskytnout před vozidlem po příjezdu z jiných pruhů (například v křižovatkách).

Ze simulací bylo získáno více než 9200 vzorků. Tyto vzorky se dělí podle počasí a maximální povolené rychlosti v daném simulačním běhu. Při porovnávání výsledků lze zjistit, že se navrhovanému modelu vozidla podařilo až v 83 procentech případů úspěšně zabránit kolizím. Některé scénáře byly navrženy tak, aby vozidlo mělo šanci na zabránění kolizi minimální, tyto scénáře testují, zda vozidlo stihne minimalizovat riziko a kolize jsou tedy očekávané, vozidlo v těchto případech správně reagovalo zpomalením.

V rámci budoucího rozvoje bych si dokázal představit vytvoření pokročilejšího modelu okolí vozidla. Nyní okolní vozidla nereagují na autonomní vozidlo, ani na další prvky okolí. V reálném systému je možné předpokládat, že se i řidič dalšího vozidla pokusí vyhnout kolizi, což modelovaný systém nezahrnuje. Dále bych rád do simulačního prostředí přidal více typů objektů, aby bylo možné modelovat i situace, kdy do cesty vozidlu vstoupí člověk, nebo zvíře. Také by bylo možné do simulace zahrnout selhání jednotlivých prvků autonomního systému, například senzorů.

Literatura

[1] Cambou, P.: In high end imaging, in the end Sony always wins. [Online; navštíveno 18.2.2019].
URL https://www.i-micronews.com/in-high-end-imaging-in-the-end-sony-always-wins/

[2] Campbell, S.: *Sensor Technology in Autonomous Vehicles : A review*. IEEE, 2018, ISBN 978-1-5386-6046-1.

[3] Digi-Key's European Editors: Radar Sensing for Driverless Vehicles. [Online; navštíveno 15.2.2019].
URL https://www.digikey.com/en/articles/techzone/2016/nov/radar-sensing-for-driverless-vehicles

[4] Hikita, M.: An introduction to ultrasonic sensors for vehicle parking. [Online; navštíveno 15.2.2019].
URL http://www.newelectronics.co.uk/electronics-technology/an-introduction-to-ultrasonic-sensors-for-vehicle-parking/24966/

[5] Lambert, F.: Tesla reveals how it hides its ultrasonic sensors in a new patent application. [Online; navštíveno 20.2.2019, dostupno také jako patent 20170059697 A1].
URL https://electrek.co/2017/03/09/tesla-ultrasonic-sensors-patent/

[6] Petr PERINGER, Martin HRUBÝ: *Modelování a simulace, prezentace*. FIT VUT, 2018.

[7] Reese, H.: Tesla's Autopilot: A cheat sheet. [Online; navštíveno 18.2.2019].
URL https://www.techrepublic.com/article/teslas-autopilot-the-smart-persons-guide/

[8] SAE International: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. *J3016, SAE International Standard*, 2018.