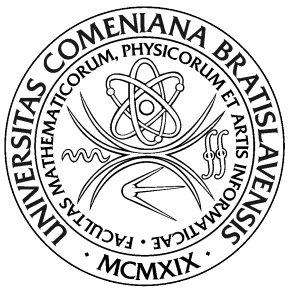
UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

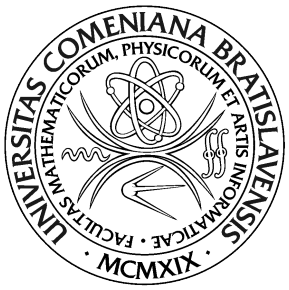


SIMULÁTOR NAVIGÁCIE V AUTOMATICKOM TRANSPORTNOM SYSTÉME

Diplomová práca

2020 Bc. Tomáš Bočinec

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



SIMULÁTOR NAVIGÁCIE V AUTOMATICKOM TRANSPORTNOM SYSTÉME

Diplomová práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Vedúci záverečnej práce:	Mgr. Pavel Petrovič, PhD.

Bratislava, 2020 Bc. Tomáš Bočinec

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

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

Názov: Simulátor navigácie v Automatickom transportnom systéme
Simulator of navigation in Automatic Transport System

Anotácia: V Laboratóriu autonómnej mobility na FMFI UK prebieha dlhodobý vývoj Automatického transportného systému založeného na autonómnych elektrických vozidlách jazdiacich po označenej dráhe. Vozidlá získavajú spätnú väzbu čítaním značiek na vozovke, na základe ktorej v spolupráci s virtuálnou mapou riadia pohyb vozidla. Počas vývoja systému je potrebné vykonať veľa testov a naladiť množstvo parametrov, čo je pri spúšťaní reálnych vozidiel náročný proces, hoci viaceré parametre a algoritmy by sa mohli najskôr testovať v simulácii. Ušetrilo by sa tým jednak množstvo času a naopak simulátor umožňuje oveľa lepšie monitorovať priebeh jazdy a dáva priestor na odskúšanie viacerých - potenciálne lepších algoritmov riadenia. Jazda v simulácii umožní tomu istému riadiacemu softvéru riadiť simulované vozidlo ako keby išlo o riadenie reálneho vozidla. Simulátor bude naviac viesť podrobné záznamy a umožní analyzovať a vizualizovať prebehnutú jazdu. Simulátor bude dobre konfigurovateľný pre rozličné verzie vozidiel i rozličné situácie vo virtuálnej mape. Súčasťou práce bude spolupráca na práci v tíme laboratória autonómnej mobility - overenie funkčnosti simulátora na reálnych problémoch, ktoré tím pri vývoji rieši.

Literatúra: K.Šimnová: Riadenie vozidla po dráhe osadenej 3D značkami, diplomová práca, FMFI UK Bratislava, 2018.
G.E. Prince, S.P. Dubois: Mathematical models for motion of the rear ends of vehicles, Mathematical and Computer Modelling, vol. 49, p. 2049–2060, by Elsevier Ltd., 2009.
James C. Alexander, J. H. Maddocks: ON THE MANEUVERING OF VEHICLES, SIAM J. AppL. MATH., Vol. 48, No. February 1988.

Kľúčové slová: automatický transportný systém, simulácia, navigácia

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

Dátum zadania: 22.10.2018

Dátum schválenia: 22.10.2018 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

Čestne prehlasujem, že túto diplomovú prácu som vypracoval samostatne len s použitím uvedenej literatúry a za pomoci konzultácií u môjho školiteľa.

. . . . . .

Bratislava, 2020 Bc. Tomáš Bočinec

Poďakovanie

Ďakujem vedúcemu záverečnej práce Mgr. Pavel Petrovič, PhD. za jeho ochotu, častú a rýchlu pomoc, bez ktorej by táto práca nebola zrealizovaná. Taktiež by som chcel poďakovať mojej priateľke Nikolke a bratovi Filipovi, ktorí ma pri tvorbe diplomovej práce podporovali.

Abstrakt

Na FMFI UK prebieha dlhodobý vývoj Automatického transportného systému. Cieľom tohto výskumu je vytvoriť globálne riadený autonómny systém, ktorý je založený na jazde autonómnych elektrických vozidiel. Tieto vozidlá budú jazdiť po špeciálnej inteligentnej vozovke tvorenej špeciálnymi riadiacimi značkami. Cieľom tejto diplomovej práce je vytvoriť simulátor vozidla, ktorý bude simulovať autonómne vozidlo, teda pohyb vozidla v priestore podľa Ackermannovho kinematického modelu a simulované čítanie značiek z inteligentnej vozovky v zodpovedajúcom čase a na zodpovedajúcom mieste. Hlavnou motiváciou je urýchliť vývoj riadiaceho systému, konkrétne algoritmy optimálnej dráhy vozidla. Vďaka simulátoru bude jednoduchšie nastaviť jednotlivé parametre vozidla a testovať rôzne druhy dopravných situácií. Prvá časť práce sa zaoberá nevyhnutným teoretickým základom ako je problematika simulovania, informácie o autonómnych vozidlách a bližší popis projektu Autonómny transportný systém. Druhá časť práce sa zaoberá vývojom aplikácie simulátora od návrhu až po jeho testovanie. Súčasťou práce je aj matematický opis správania simulovaného automobilu. Výsledkom práce je funkčná aplikácia simulátora, s ktorou vie komunikovať potrebný zvyšok projektového softvéru.

Kľúčové slová: automatický transportný systém, simulácia, navigácia

Abstract

At FMFI UK, the long-term development of the Automatic transport system is ongoing. The aim of this research is to create a globally controlled autonomous system that is based on the driving of autonomous electric vehicles. Those vehicles will drive on a special intelligent road consisting of special control signs. The aim of this diploma thesis is to create a vehicle simulator that will simulate an autonomous vehicle, thus, the movement of the vehicle in space according to the Ackermann kinematic model and the simulated reading of marks from the intelligent road at the corresponding time and corresponding place. The main motivation is to accelerate the development of the control system, specifically the algorithms of the optimal vehicle path. Thanks to the simulator, it will be easier to set individual vehicle parameters and the possibility of testing various traffic situations. The first part of the thesis deals with the necessary theoretical basis such as simulation, information about autonomous vehicles and a detailed description of the project Autonomous transport system. The second part deals with the development of the simulator application from design to testing. The result of the diploma thesis is a functional application of a simulator. With final application can communicate the necessary rest of the project software.

Keywords: automatic transport system, simulation, navigation

Obsah

Kapitola 1

Úvod

Autonómne vozidlá už nie sú iba otázkou budúcnosti, ale prítomnosti, ktorú môžeme pozorovať aj na našich cestách. Jednotlivé krajiny sa postupne pripravovali na príchod technológie autonómnych vozidiel, ktoré im zabezpečia bezpečnosť na cestách a hlavne aj vyššiu mobility starších ľudí. Aj napriek tomu, že systémy ovládania vozidiel je potrebné dôkladne otestovať, ich vývoj je veľmi rýchly. Väčšina veľkých výrobcov dopravných prostriedkov sa zameriava na investovanie do vývoja autonómnej technológie.

V Laboratóriu autonómnej mobility na FMFI UK vznikol projekt Automatický transportný systém (ďalej "ATS"), ktorého cieľom je vytvoriť globálne riadený autonómny systém. Tento systém je založený na jazde autonómnych elektrických vozidiel. Projekt vznikol ako alternatíva súčasným drahým a komplikovaným systémom. Vozidlo sa nepohybuje voľne po verejnej komunikácii ale po špeciálnej inteligentnej vozovke, ktorá je tvorená riadiacimi značkami slúžiacimi k navigácii vozidla. V súčasnej dobe sa na projekte vyvíja nový algoritmus riadenia. Konkrétne sa jedná o algoritmus najoptimálnejšej dráhy. Pre účely vývoja bolo potrebné vytvoriť simulátor, ktorý bude pomocou matematických modelov simulovať správanie automo-

bilu. Úlohou simulátora je urýchliť vývoj a umožniť niektoré testy robiť iba virtuálne. Simulátor umožní jednoduchšie nastavenie rôznych vstupných parametrov pre rôzne typy vozidla a zároveň bude možné vyskúšať rozličné dopravné situácie. Dôležitou súčasťou simulácie je matematický model, ktorý vyjadruje správanie sa automobilu s Ackermannovým riadením.

Na začiatku je práca venovaná teoretickým východiskám, ktoré boli potrebné na celkové zrealizovanie projektu. V rámci tejto časti sú vysvetlené pojmy ako experiment a modelovanie. Súčasne sa kapitola venuje podrobnejšiemu popisu simulácie, jej využitia a klasifikácie a taktiež aktuálnej problematike autonómnych vozidiel. Zároveň kapitola popisuje ovládanie pohybu vozidla, ktorý slúži ako východisko pre matematický model simulácie. Záver kapitoly je venovaný bližšiemu popisu projektu Automatický transportný systém, jeho technickému vybaveniu a opisom softvéru, na ktorý sa bude simulátor integrovať.

Nasledujúca kapitola sa zaoberá vymedzeniu a opísaniu cieľov diplomovej práce.

Štvrtá a piata kapitola sa venuje metodike práce a postupu vývoja aplikácie simulátora. V tejto časti sú uvedené jednotlivé kroky popisujúce tvorbu aplikácie.

V ďalšej kapitole sú podrobne popísané jednotlivé funkcie aplikácie, ktoré boli implementované v rámci projektu spolu s vysvetlením ich využitia.

Súčasťou siedmej kapitoly je opis matematického modelu pohybu auta a rozpoznávania značiek z virtuálnej mapy. Kapitola vysvetľuje aj implementáciu týchto modelov v kóde aplikácie simulátora.

Záverečná kapitola sa zaoberá manuálom k aplikácií, požiadavkami na spustenie aplikácie a návodu k použitiu aplikácie v základnom scenári.

Kapitola 2

Prehľad problematiky

Úvodná kapitola sa zameriava na vymedzenie teoretických východísk, ktoré predstavujú nevyhnutnú súčasť celej diplomovej práce. Sú základným zdrojom informácií potrebných na pochopenie riešenej problematiky.

2.1 Experiment

Experiment je proces, pri ktorom získavame nové informácie o skúmanom systéme s možnosťou testovania rôznych vstupných atribútov, ktoré systém ovplyvňuje. Proces, pri ktorom meníme rôzne vstupné atribúty nazývame experimentovanie. Dobré experimentálne riešenie je také, na ktorom môžeme vyskúšať čo najviac možných vstupov a zároveň vieme pozorovať a zaznamenávať správanie sa skúmaného systému. Ideálne v reálnom čase [18].

Problémy, ktoré môžu vzniknúť pri experimentovaní:

1. experiment môže byť nákladný - napríklad crash-test experiment,
2. experiment môže byť nebezpečný - napríklad experimentovanie správanie posádky vozidla pri havárií,
3. skúmaný problém neexistuje - napríklad pre systémy, ktoré iba navrhujeme alebo ich aktuálne nie je možné vytvoriť.

Nedostatky experimentov viedli k vzniku konceptu Modelu.

2.2 Model

Model je hocičo, čo nám umožňuje vykonávať experiment. Je to väčšinou reprezentácia určitého systému. Systém je súbor organizovaných udalostí alebo objektov, ktoré pozorujeme alebo zasahujú do pozorovania [13]. Definujeme štyri hlavné triedy systémov, a to:

- prírodné systémy (počasie),
- vytvorené fyzikálne systémy (auto),
- abstraktné systémy (matematika),
- ľudské aktivity (politika).

Môže existovať viacero modelov reprezentujúcich ten istý systém. To znamená, že test nevykonávame na reálnom systéme, ale na zjednodušenom modeli. Pri tomto zjednodušenom modeli aproximujeme reálne správanie a napodobňujeme jeho skutočné vlastnosti [13]. Najčastejšie sa stretávame s pojmami Fyzický a Matematicky model.

Fyzický model

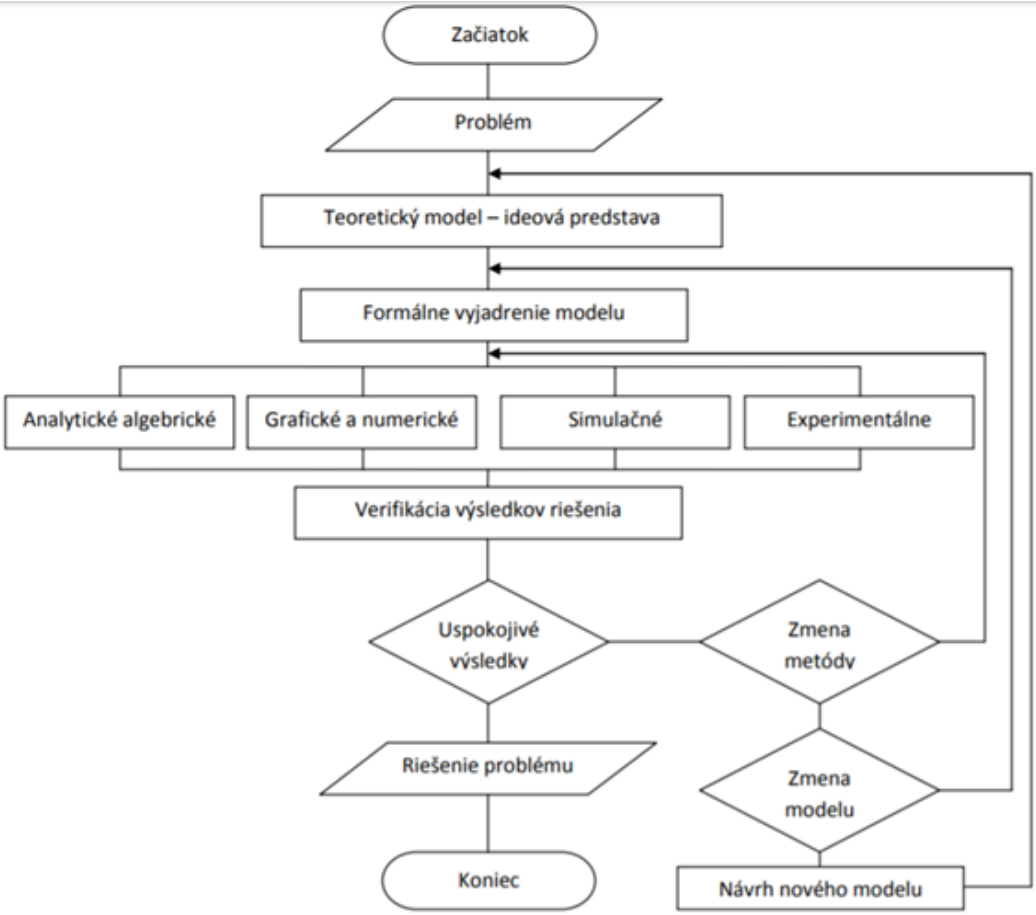
Fyzický model napodobňuje vlastnosti reálneho fyzického objektu. Napríklad model karosérie auta, na ktorom môžeme testovať aerodynamické vlastnosti.

Matematicky model

Matematický model je popis systému, kde pomocou matematiky znázorňujeme vzťahy a závislosti medzi premennými. V tomto prípade premenné predstavujú rôzne vlastnosti, ako napríklad veľkosť, hmotnosť, teplota a rýchlosť. Matematické modely najčastejšie popisujú rôzne prírodné zákony [13].

2.3 Simulácia

Simulácia je napodobenina činnosti procesu alebo systému, ktorá predstavuje jeho fungovanie v čase [7]. Teoreticky sa pojmy model a simulácia líšia. Model predstavuje nástroj, zatiaľ čo simulácia je činnosťou použitia tohto nástroja [12]. Tieto pojmy sa však v praxi často zamieňajú. Výsledok simulácie sa často nazýva simulácia.



Obr. 2.1: Riešenie vedecko-technických zadaní [13]

Vďaka simulácii vieme riešiť vedecko-technické úlohy. Príklad takéhoto riešenia je schematicky znázornený na Obr. 2.1.
Pri tejto tématike je dobre si ozrejmiť nasledujúce pojmy:

- statická simulácia - imitácia sôch, budov,
- dynamická simulácia - imitácia určitého systému vrátane jeho prechodu v čase,
- numerická (počítačová) simulácia - imitácia systému pomocou počítačového softvéru.

Hlavné dôvody využitia počítačovej simulácie:

- lacnejšie a bezpečnejšie ako experimenty,
- vieme meniť niektoré parametre - napríklad urýchliť čas,
- vieme dosiahnuť podmienky, ktoré by sme nevedeli dosiahnuť pri experimentoch,
- rýchlejšia a jednoduchšia zmena atribútov modelov,
- potlačenie porúch reálneho zariadenia,
- vieme zanedbať nežiadané atribúty.

2.3.1 Využitie simulácie

Simulácia sa môže využívať takmer vo všetkých odvetviach, pri rôznom plánovaní, optimalizácii, vývoji, analýze atď.

Príklady aplikovanej simulácie:

- počítačové hry,
- simulácia vývoja ekonomických ukazovateľov,
- simulácia počasia,
- simulácia pohybu auta,
- simulácia požiaru.

2.3.2 Modelovanie a numerická simulácia

Numerický model je model implementovaný do numerického nástroja. V praxi ide o prepis a naprogramovanie matematického modelu do numerického programu alebo programovacieho jazyka. Vďaka tomu vieme aj zložitejšie výpočty počítať s tou istou alebo podobnou presnosťou. Avšak tento proces je oveľa rýchlejší, čo nám často umožňuje vyskúšať oveľa viac možných parametrických vstupov.

2.3.3 Reprezentácia stavu

Je to nejaká stavová premenná, ktorá sa môže počas simulácie meniť a vieme ju sledovať.

2.3.4 Limity a nevýhody simulácie

Vyššie uvádzame najmä výhody simulácie, avšak môžu nastať aj situácie, kedy nie je vhodné simulovať. Môže sa stať, že nie každá simulácia môže byť z technických dôvodov uskutočniteľná.

Matematické a numerické limity: Môže nastať situácia, kedy niektoré zložité matematické rovnice alebo funkcie ešte neboli vyriešené, prípadne na ich počítanie by sme potrebovali výpočtový výkon, ktorý nám súčasné počítače ešte neumožňujú.

Nedostatočné znalosti: Príkladom môže byť situácia, kedy niektoré znalosti fyziky môžu byť nedostatočné na to, aby sme vedeli daný jav dostatočne simulovať.

2.3.5 Rozhodovanie sa o simulácii

Pri rozhodovaní sa o numerickej simulácii je hlavnou otázkou, či je investícia na jej vytvorenie zisková alebo nie. Pri ziskovosti treba posudzovať aj dlhodobé aspekty a výhody simulácie. Až na základe týchto aspektov je potrebné sa rozhodnúť, či je vhodné použiť simuláciu alebo nie.

2.3.6 Rôzne druhy numerických simulácií

Existujú rôzne klasifikácie simulačných modelov. Pre každý druh boli navrhnuté špeciálne techniky a osvedčené postupy ako k ním pristupovať a ako ich vytvárať [13].

Dynamická alebo statická

Model je dynamický, keď sa jeho stavové premenné časom menia - napríklad teplota ľadu v horúcej vode. Tieto modely sa väčšinou opisujú pomocou diferenciálnych rovníc. Model je statický, ak nezávisí od času. Príkladom môže byť model hodnotiaci intenzitu prechádzajúceho elektrického prúdu cez elektrický odpor, ktorý sa podľa Ohmovho zákona nemení.

Diskrétne alebo súvislé

Model je diskrétny vtedy, keď jeho vnútorné premenné sa menia len určitý konečný počet krát. Príkladom môže byť evakuačná simulácia, kedy sa môže evakuovať iba určité množstvo ľudí a nemôže sa evakuovať určitá časť jedného človeka. Naopak topenie kocky ľadu je súvislé.

Determistické alebo stochastické

Model je deterministický v prípade, ak sa nespolieha na náhodné javy. Keď simuláciu pustíme viackrát s tými istými vstupmi, vždy dostaneme také isté výsledky. Naopak model je stochastický v prípade, ak zahŕňa náhodný alebo pravdepodobnostný výber. Napríklad simulácia pohybu áut cez križovatku.

Počet dimenzií

0D model, nazývaný aj systémový model, nezohľadňuje rozmerné premenné. Z toho dôvodu je často iba abstraktný. Pojem 1D využíva iba jednu dimenziu (1D geometrický tvar). Najčastejšie sa stretávame s 2D a 3D simuláciou.

Monofyzikálny a multifyzikálny

Monofyzikálne modely sa vzťahujú na jednu špeciálnu oblasť fyziky (termodynamika, dynamika). Naopak, model je multifyzikálny, ak zahŕňa viacero fyzikálnych zákonov z rôznych oblastí. Takéto modely sú často komplexnejšie. Napríklad môže ísť o model veternej elektrickej turbíny.

Procesný alebo funkčný

Procesný model predpovedá produkčnú funkcionalitu vo fáze vývoja z dôvodu správnej optimalizácie reálneho systému. Účelom funkčného modelu je predpokladať efektívnosť systému počas používania.

Poznáme päť krokov na vytvorenie numerickej simulácie:

1. definovanie cieľa simulačnej štúdie,
2. navrhnutie a vytvorenie matematicko-fyzikálneho problému,
3. konverzia modelu na numerický model,
4. samotné simulovanie a vytváranie výstupných dát,
5. uloženie modelu, spracovanie a uloženie výsledkov.

2.3.7 CAS (Computer Algebra System)

Skratka CAS označuje počítačový softvér vhodný na manipuláciu a počítanie matematických výrazov. Takýto softvér vie napríklad spracovať a zjednodušiť matematický výraz, vypočítať integrál, vypočítať lineárne rovnice a pod. Niektoré CAS vedia okrem výpočtov aj rôzne matematické modely vizualizovať a aj preto sú vhodné na tvorbu simulácií. Medzi najznámejší CAS patria aplikácie: Axiom, Maxima, Magma, Maple, Mathematica, SageMath. Matematická disciplína, ktorá sa zaoberá softvérovými matematickými výpočtami sa nazýva numerická algebra.

2.3.8 Java Algebra Systém (JAS)

JAS je knižnica, ktorá umožňuje objektovo-orientovaný, typovo-bezpečný a multivláknový prístup k počítačovej algebre. Knižnica je voľne šíriteľná a môže byť použitá ako súčasť inej knižnice alebo aplikácie. Pomocou tejto knižnice môžeme jednoducho počítať aj zložitejšie matematické problémy, ako napríklad komplexné a reálne korene polynómov, Legendrove polynómy,

mocninové rady. Knižnica je vhodná aj na zložitejšie výpočty, pretože podporuje aj 64-bitovú architektúru a možnosť výpočtu na viacerých jadrách v procesore alebo aj viacerých procesorov súčasne [5].

2.3.9 Budúcnosť numerickej simulácie

Odhaduje sa, že numerická simulácia sa bude rozširovať vo všetkých odvetviach. Čoraz väčšia výpočtová sila nám umožňuje detailnejšie simulácie, a teda aj ich pokročilejšie a lepšie využitie. Aplikácie na tvorbu simulácií sú čoraz jednoduchšie. Nakoľko je v nich pripravených veľa pomocných fyzikálnych modelov, programátor nemusí mať pokročilé fyzikálne znalosti. Veľký rozruch pri tvorbe simulátorov vidíme aj pri rozmachu umelej inteligencie[14].

2.3.10 Umelá inteligencia a simulácia

Umelá inteligencia a simulácia sa čoraz viac prekrývajú a dopĺňajú. Problémom strojového učenia sú najčastejšie dáta. Ak chcete naučiť jazdiť autonómne vozidlo, budete potrebovať tisíce kilometrov ľudskej jazdy, čo je veľmi nákladné. Avšak ak by ste mali dobrý numerický simulátor, mohli by ste si nagenerovať ľubovoľné množstvo dát bez námahy. Takýto model strojového učenia môžeme následne použiť v komplexnejších simulačných systémoch. Vďaka metódam strojového učenia vieme vytvárať aj simulátory na komplexné problémy, pri ktorých nemáme dostatočné vedecké vedomosti, alebo výpočty by boli príliš zložité. Naopak vo väčšine prípadov potrebujeme veľké množstvo pozorovaných dát. Metódy umelej inteligencie nám môžu taktiež simulovanie zrýchliť, a tým spraviť menej nákladné. Napríklad simulácia molekulárnych reakcií vyvolaných svetlom je doteraz časovo náročná a preto aj nákladná. Vedci z Viedenskej univerzity teraz predstavili metódu využívajúcu hlboké neurónové siete, ktorá drasticky urýchľuje simuláciu potrebných

procesov. V rámci štúdie uskutočnili fotodynamickú simuláciu, ktorá trvala dva mesiace. Ak by vedci využili predchádzajúce metódy tak by simulácia trvala 19 rokov [17].

2.3.11 Simulácia dráhy vozidla

Simulátor dráhy vozidla je simulátor, ktorý predpovedá pohyb a trasu auta. Zohľadňuje rôzne atribúty medzi ktoré zaraďujeme druh vozidla, veľkosť vozidla, počet náprav, rýchlosť vozidla a iné. Často sú zohľadnené aj parametre vozovky, po ktorej sa vozidlo pohybuje. Príkladom môže byť povrch alebo sklon vozovky. Ide o dynamický, súvislý, deterministický, multifyzikálny simulátor, ktorý väčšinou počíta s dvoma alebo troma dimenziami prostredia. Simulátor môže využívať rôzne evolučné algoritmy, algoritmy umelej inteligencie, prípadne analytické riešenia.

2.4 Autonómne vozidlá

Autonómne vozidlo je vozidlo, ktoré je schopné sa pohybovať s malým alebo žiadnym ľudským vstupom. Jednou z hlavných úloh autonómnych vozidiel je správne rozpoznanie okolia, načo sa využívajú rôzne senzory, radary, sonary, GPS. Tie majú za úlohu nahradiť ľudské pozorovanie [1].

V označovaní autonómnych vozidiel nastal zmätok. V dôsledku toho vznikla presná norma SAE (J3016), ktorá klasifikovala jednotlivé autonómne vozidlá do piatich skupín podľa miery automatizácie. Pri úrovni autonómie 0 systémy automobilu nemajú nad vozidlom žiadnu kontrolu, ale môžu spustiť aspoň varovanie, napríklad pípnuť, ak sa druhé vozidlo dostane do nášho mŕtveho uhla. Pri druhej úrovni už vie vozidlo vykonávať niektorú činnosť samo, ale vodič musí byť pripravený kedykoľvek prevziať situáciu do vlastných rúk, na-

príklad adaptívny tempomat. V prípade druhého levelu už automobil dokáže zrýchliť, brzdiť aj zatáčať, ale vodič musí stále sledovať okolie a v prípade potreby prevziať riadenie. Do tejto kategórie patrí napríklad prvá generácia autopilota firmy Tesla. Pri úrovni autonómie 3 nie je vodič nútený sledovať okolie v známom prostredí a auto dokáže jazdiť samo. Napríklad auto je schopné na diaľnici fungovať bez zásahov vodiča. Autonómne vozidlá štvrtej úrovne dokážu jazdiť, až na špeciálne výnimky, bez zásahu vodiča. Medzi takéto výnimky môžeme zaradiť napríklad mimoriadne zhoršené povetornostné podmienky. Pri úrovni 5 je už vozidlo plne autonómne a dokáže robiť všetku potrebnú funkcionalitu bez ľudského zásahu[10]. Prehľad sa nachádza na Obr. 2.2.

SEA level	Názov	Vykonanie zrýchlenia riadenia a spomalenia	Monitorovanie jazdného prostredia	Dynamické jazdne vlastnosti	Vodické režimy jazdy
Človek monitoruje prostredie					
0	Bez automatizácie	Ľudský vodič	Ľudský vodič	Ľudský vodič	bez
1	Jazdná asistencia	Ľudský vodič a systém	Ľudský vodič	Ľudský vodič	niekoľko
2	Čiastočná automatizácia	Systém	Ľudský vodič	Ľudský vodič	niekoľko
Automatizovaný systém monitoruje prostredie					
3	Podmienená automatizácia	Systém	Systém	Ľudský vodič	niekoľko
4	Vysoká automatizácia	Systém	Systém	Systém	niekoľko
5	Plná automatizácia	Systém	Systém	Systém	všetky jazdné modely

Obr. 2.2: Prehľad úrovní autonómnych vozidiel [10]

Dynamické jazdné vlastnosti zahŕňajú prevádzkové úlohy (riadenie, brzdenie, zrýchlenie, monitorovanie vozidla a vozovky) a taktické úlohy (reagovať

na udalosti, reagovať na vzniknuté náhodné situácie, správne určiť moment zmeny jazdného pruhu, správne vyhodnotiť potrebu odbočenia či použitia signálov a iné).

Režim jazdy je špeciálnym typom scenára jazdy s charakteristickými požiadavkami dynamickej jazdy. Príkladom môžu byť dopravné zápchy, jazda v uzavretom areáli, jazda v pešej zóne a iné.

2.4.1 Testovanie autonómnych vozidiel

Testovanie autonómnych vozidiel je veľmi náročné, často nebezpečné a navyše testovanie na verejnej komunikácií nám nemusí umožňovať ani legislatíva. V Spojených štátoch amerických je síce povolené testovanie, ale vo vozidle musí byť nonstop kontrolujúca a zodpovedná osoba, ktorá v prípade zlyhania systému vie prebrať riadenie. Aj kvôli týmto nedostatkom sa kladie veľký dôraz na vývoj simulátorov autonómnych vozidiel. Jedným z takýchto simulátorov je aj opensource systém CARLA, ktorý umožňuje testovanie autonómnych vozidiel vo virtuálnej realite [4].

2.4.2 Výhody a nevýhody autonómnych vozidiel

Výhody autonómnych vozidiel:

- vyššia bezpečnosť,
- nižšie náklady spojené s infraštruktúrou a spotrebou paliva,
- nižšia potreba miest na parkovanie,
- pohodlie,
- nižší počet nehôd.

Nevýhody autonómnych vozidiel:

- zaradenie autonómnych vozidiel do bežnej dopravy,
- problém súvisiaci s právom na súkromie, nakoľko vozidlá nepretržite snímajú vonkajšie prostredie,
- zníženie dopytu po práci šoférov,
- zneužitie technológie, či samotná jej bezpečnosť.

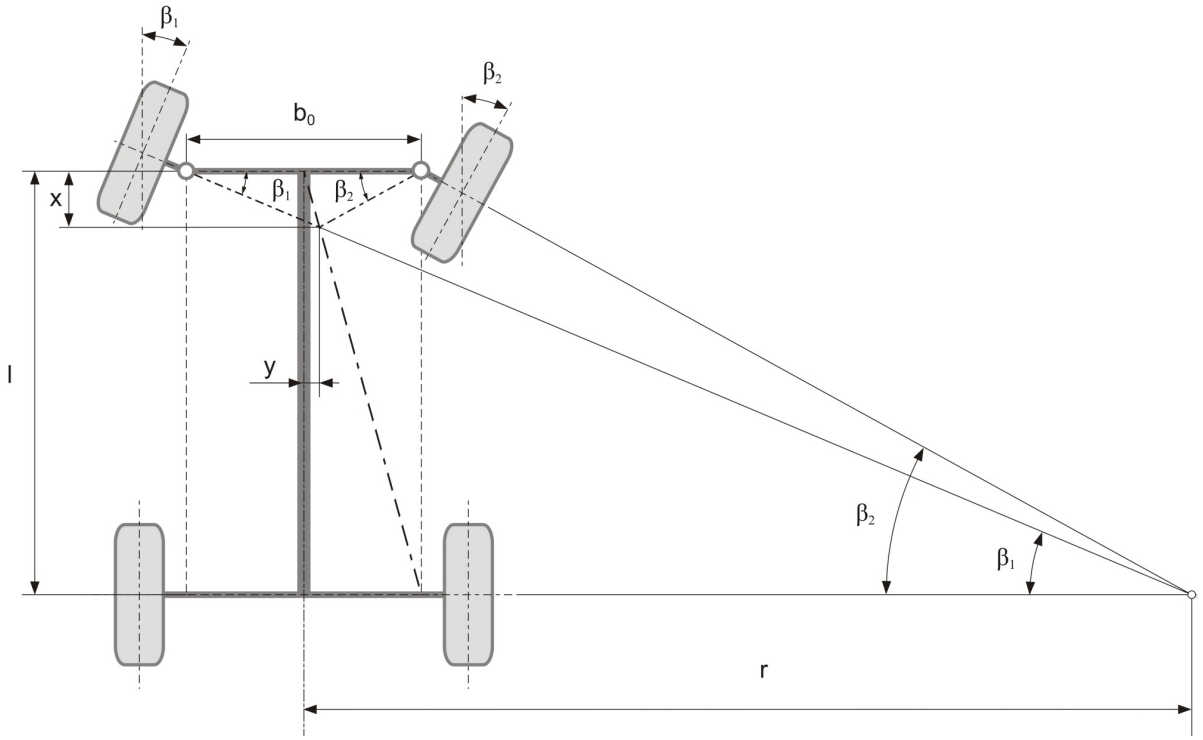
2.5 Ovládanie vozidla

Podkapitola opisuje teoretické poznatky z oblasti mechaniky a matematiky, ktoré sú podstatné na pochopenie problémov riadenia vozidla.

2.5.1 Ackermannove riadenie

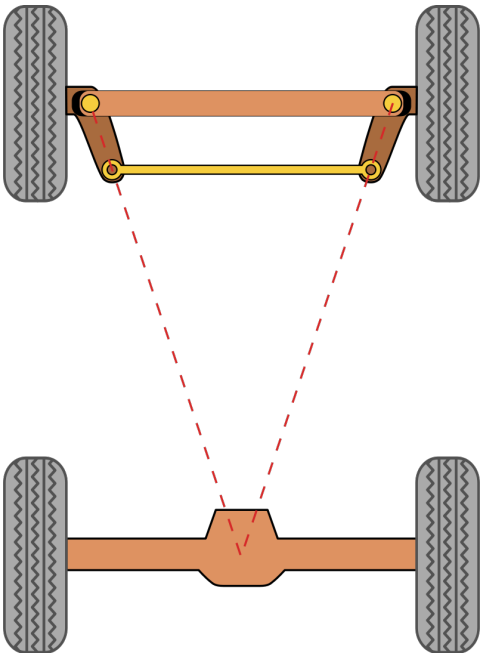
Ackermannov model riadenia definuje geometriu, ktorá rieši problém dráhy kolies na vnútornej a vonkajšej strane zákruty, ktoré prechádzajú rôzne vzdialenosti po kružniciach rôznych polomerov [19]. Koleso, ktoré je vo vnútornej strane zákruty sa musí pootočiť o väčší uhol ako vonkajšie. Výsledkom je, že kolesá na aute neprešmykujú. Ackermannove riadenie sa používa od začiatku 19. storočia až dodnes. Tento model riadenia využívajú všetky modely áut navrhnuté pre potreby projektu.

Na Obr. 2.3 je znázornená vzdialenosť l, ktorá predstavuje vzdialenosť medzi prednou a zadnou nápravou. Vzdialenosť b0 popisuje vzdialenosť medzi osami pneumatík na náprave. Vzdialenosť r predstavuje polomer otáčania. Zadná náprava smeruje priamo do stredu otáčania. Inak povedané, vozidlo je vždy kolmo na stred otáčania. Uhly $\beta 0$ a $\beta 1$ predstavujú uhly otočenia predných kolies vzhľadom na aktuálne natočenie vozidla.



Obr. 2.3: Geometria Ackermannovho riadenia [16]

Na Obr. 2.4 je approximačne znázornená konštrukcia vozidla s Ackermannovým riadením, ktorá umožňuje lepšiu predstavu o spôsobe zatáčania vozidla.



Obr. 2.4: Aproximácia Ackermannoveho modelu riadenia [3]

2.5.2 Diferenciálny pohon

Keďže pri pohybe vozidla v zákrute vykonáva každé koleso inú dráhu, tým pádom sa rozdielne kolesá musia točiť rozdielnou rýchlosťou, aby nedochádzalo k prešmykovaniu. To znamená, že kolesá nemôžu byt pevne spojené. Na modeli auta v laboratóriu máme taktiež diferenciálny pohon. [11].

2.5.3 The BUS PROBLEM

Článok Mathematical models for motion of the rear ends of vehicles sa venuje matematickým modelom pohybu viacnápravových vozidiel. Opisuje pohyb vozidiel po rovných, kruhových a kľukatých cestách. V článku matematicky dokázali, že priama a kruhová dráha zadnej časti vozidla je nezávislá od rýchlosti vozidla počas celej dráhy. Tento výsledok je dôležitý, pretože v

numerickej aj symbolickej simulácii môžeme simulovať rôzne parametre bez toho, aby sme museli zohľadňovať rýchlosť vozidla [15].

2.6 Projekt Automatický transportný systém

Cieľom projektu je zostrojiť Automatický transportný systém založený na autonómnych elektrických vozidlách. Tieto vozidlá jazdia po označenej dráhe za pomoci virtuálnej mapy. Celý systém je riadený globálne. Projekt sa snaží poukázať na možnú alternatívu dopravy, aktuálny trend autonómnych vozidiel a tým zostrojiť systém, ktorý bude bezpečný, plynulý a cenovo dostupný. Systém sa vyvíja a testuje na Fakulte matematiky, fyziky a informatiky Univerzity Komenského v Bratislave, kde sa nachádza aj vývojové laboratórium. Aktuálne sa tím zaoberá navigáciou automobilu a stým spojeným výberom optimálnej trasy.

2.6.1 Technické a softvérové vybavenie laboratória

Medzi technické a softvérové vybavenie laboratória ATS patrí: vozidlo, Gocator, softvérové riešenie, rozpoznávací algoritmus, testovacia trať.

Vozidlo

V laboratóriu máme 2 dvojnápravové RC vozidlá. Prvé vozidlo je v mierke 1/10 ku reálnemu vozidlu. Parametre vozidla sú:

- vzdialenosť medzi nápravami je 260 mm,
- maximálny uhol zatočenia vnútorného kolesa je 24 ∘ - zodpovedá minimálnemu polomeru otáčania asi 665 mm,
- rozpätie laserovej čiary (pozri časť Gocator) je 206 mm,

- hmotnosť je 7,1 kg,
- rozchod - vzdialenosť medzi ľavým a pravým kolesom je 163 mm.

Druhé vozidlo je v mierke 1/8 ku reálnemu vozidlu a aktuálne sa využíva na vývoj. Parametre vozidla sú:

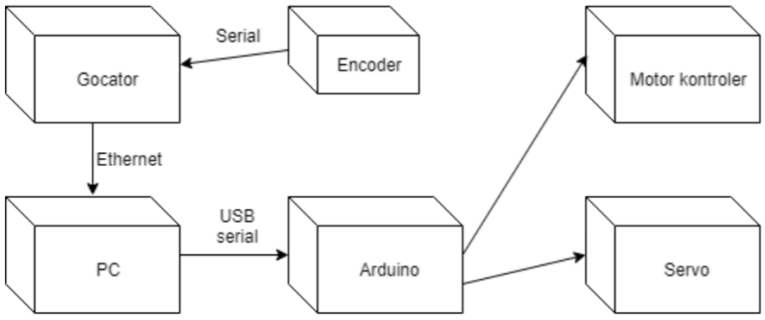
- maximálna rýchlosť je 25 km/h,
- vzdialenosť medzi nápravami je 305 mm,
- vzdialenosť medzi kolesami na náprave je 162 m,
- maximálny uhol zatočenia vnútorného kolesa je 20 ∘ - zodpovedá minimálnemu polomeru otáčania asi 892 mm,
- vzdialenosť čítacej čiary snímača od prednej nápravy je 189 mm,
- rozchod - vzdialenosť medzi ľavým a pravým kolesom je 188 mm.

Obr. 2.5: RC model [19]

Vozidlo využíva nasledujúci hardware:

- Kontrolér motora,
- Encoder na meranie aktuálnej rýchlosti,
- Servomotor,
- Mikrokontrolér Arduino,
- PC na lokálne výpočty (riadenie automobilu),
- 3D laser (Gocator) - zariadenie na rozpoznávanie značiek pomocou merania výškových profilov.

Predpokladá sa, že systém bude možné využívať na všetkých druhoch vozidiel. Pri tomto druhu autonómnych vozidiel je výhoda najmä ich nižšia cena.

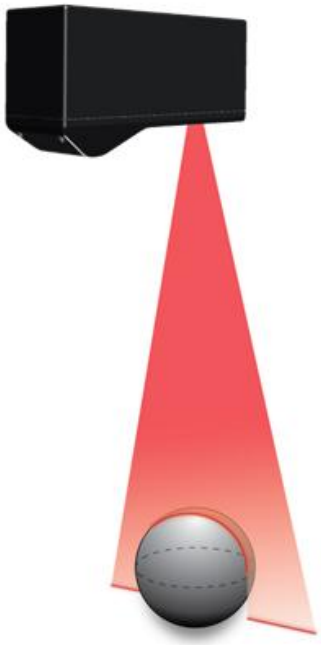


Obr. 2.6: Architektúra jednotlivých komponentov [19]

Gocator

Je 3D laser na meranie výškových profilov. V projekte používame zariadenie na rozpoznávanie značiek, ktoré označujú dráhu. Zariadenie je vysokorýchlostne (32 kHz), čo umožňuje čítanie značiek aj pri vysokých rýchlostiach.

Gocator je umiestnený na prednej časti auta a sníma vozovku pred vozidlom. Prístroj funguje na báze svetelných lúčov, ktoré sa odrážajú od povrchu a následne ich zachytáva vysokorýchlostná kamera. Medzi výhody zariadenia patria jeho kompaktné rozmery, nízka váha, a to že zariadenie nevyžaduje žiadne ďalšie riadiace jednotky. Gocator je dodávaný spolu s Gocator SDK, čo je knižnica, ktorá umožňuje prístup a kontrolu nad senzorom.



Obr. 2.7: Gocator [2]

Softvérové riešenie

Prevažná časť systému je naprogramovaná v programovacom jazyku C a menšie časti v jazyku Javascript. Kód je rozdelený do logických modulov. Používateľ môže zariadenie ovládať priamo z konzoly, prípadne z webového rozhrania.

Rozpoznávací algoritmus

Dôležitou časťou aplikácie je algoritmus na rozpoznávanie vodiacich značiek. Algoritmus na vstupe dostáva dáta zo senzora Gocator. Keďže Gocator produkuje veľké množstvo dát, musíme dáta vedieť správne a rýchlo spracovať. Prvým krokom algoritmu je nájsť v dátach značky a vyselektovať ich hrany. Druhým krokom je klasifikácia značiek. Bolo otestovaných viacero druhov značiek i viaceré verzie rozpoznávacích algoritmov. Napríklad algoritmus vyvinutý v predchádzajúcej diplomovej práci [19] dosahoval úspešnosť až 95%, neskôr boli značky zjednodušené a úspešnosť rozpoznania sa ďalej zlepšila.

Testovacia trať

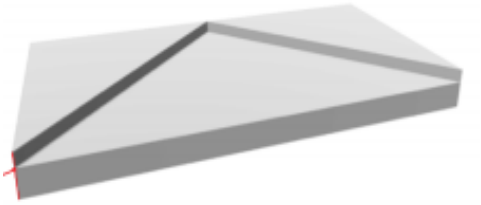
Na testovanie jednotlivých algoritmov a funkčnosti celého systému je v laboratóriu vytvorená testovacia dráha, viď. Obr. **??**. Trať pozostáva z kružnice s 3 m priemerom a 5 m širokého oválu. Na trati je rozmiestnených 10 druhov značiek vo vzdialenosti 25 cm.

Obr. 2.8: Testovacia trať [19]

Obr. 2.9: Testovacia trať z blízka [19]

Vodiaca značka

3D vodiaca značka je znázornená na Obr. 2.10. Jej rozmery sú 20 x 30 x 3 mm. Výškový rozdiel na identifikáciu konkrétnej značky je 1 mm. Značky sú navrhnuté tak, aby ich bolo možné čo najrýchlejšie zosnímať a rozoznať.



Obr. 2.10: Značka použitá v [19]

Virtuálna mapa

Systém využíva pri riadení virtuálnu mapu. V tejto mape sa nachádzajú informácie o trase, ako napríklad rozmiestnenie jednotlivých značiek. Ďalšou súčasťou mapy sú informácie, ako možné smery jazdy, typ prechodu, maximálna rýchlosť a iné. Vozidlo využíva spomínanú virtuálnu mapu, aby vedelo vyhodnotiť miesto, kde sa aktuálne nachádza, prípadne akú ďalšiu značku má na zvolenej trase očakávať. Mapa je rozdelená do menších logických celkov nazývaných segmenty. Mapa využíva formát JSON.

Kapitola 3

Cieľ práce

Cieľom diplomovej práce je zostrojiť simulátor vozidiel nachádzajúcich sa v laboratóriu. Projektom sa zaoberá tím nazývaný Automatický transportný systém. Hlavnou úlohou simulátora je simulovať vozidlo. Pomocou matematických modelov je simulátor schopný napodobňovať správanie vozidla pri pohybe po dráhe a rozpoznávanie riadiacich značiek. Tieto značky sa nachádzajú na mape a riadiaca časť vozidla podľa nich rozhoduje o nasledujúcom pohybe. Simulované vozidlo sa bude pohybovať po virtuálnom priestore vytvoreného z virtuálnej mapy, ktorá bude predstavovať reálnu testovaciu mapu vozidla z laboratória. Samozrejme bude možné využívať aj iné mapy. Vyššie opísaná simulácia bude slúžiť najmä na zdokonalenie samotných riadiacich algoritmov a urýchlenie samotného vývoja. Simulátor umožní testovať viaceré dopravné situácie, ktorých variabilitu by bolo náročne zostrojiť priamo v laboratóriu. Simulátor tak umožní lepšie a rýchlejšie nastavenia jednotlivých parametrov vozidla a parametrov riadenia. Taktiež zabezpečí rýchlejšiu adaptáciu systému na nový typ vozidla s odlišnými veľkosťami a jednotlivými komponentmi. Pôjde o desktopovú aplikáciu jednoduchú na používanie, ktorá bude simuláciu aj vizualizovať. Táto aplikácia bude komunikovať s riadiacim

systémom ATS. Podľa požiadaviek zadania to znamená, že ten istý kód, ktorý beží priamo na hardvéri vozidla, sa môže namiesto na reálny hardvér pripojiť na vytvorený simulátor bez toho, aby v ňom bolo potrebné urobiť akékoľvek zmeny. Simulátor má teda plne nahradiť fyzické vozidlo jeho simulovaným modelom vrátane zabezpečenia všetkých potrebných softvérových adaptérov na automatickú integráciu s riadiacim softvérom vozidla. Pri vývoji je potrebné brať ohľad na otvorenosť a všestrannosť kódu, aby bolo jednoduché vyriešiť nový problém, ktorý môže nastať pri vývoji riadiaceho algoritmu. Vo všeobecnosti to znamená, že sa kladie dôraz na jednoduchosť a čitateľnosť kódu.

Kapitola 4

Metodika práce a metódy skúmania

Konkrétnu diplomovú prácu na tému Simulátor navigácie v transportnom systéme som si vybral na základe môjho záujmu o technológie, ktoré projekt využíva a vysokému potenciálu projektu konkurovať momentálnym transportným systémom. Verím, že výsledok diplomovej práce bude pre projekt prínosom.

Teoretický základ informácií potrebných k diplomovej práci som čerpal hlavne z rôznych vedeckých článkov, predchádzajúcich diplomových prác venujúcim sa rovnakej problematike, ale aj z cenných informácií a dát od členov tímu Automatického transportného systému.

Na začiatku samotného spracovania diplomovej práce bolo potrebné preskúmať kód, na ktorý sa bude simulátor integrovať a súčasne pochopiť algoritmus riadenia a význam jednotlivých komponentov vozidla. Súčasťou prípravy bolo aj samotné zoznámenie sa s Laboratóriom autonómnej mobility na FMFI UK. Táto príprava na samotné písanie diplomovej práce bola pre mňa prínosná. Následne bolo nevyhnutné pripraviť matematický model po-

hybu vozidla a rozpoznávania značiek. Pri tejto časti mi pomohli vedomosti z Goniometrie a Lineárnej algebry.

Je potrebné položiť si otázku: "Prečo bolo treba simulátor?"Vývoj riadiaceho algoritmu pre vozidlo bol pomalý, z dôvodu že po každej zmene bolo treba nanovo odštartovať nový experiment, prichystať auto a vyskúšať rôzne dopravné situácie. Jednotlivé experimenty mohli byť navyše nepresné, napríklad z dôvodu slabej batérie. Simulátor tieto nedostatky odstraňuje a minimalizuje počet experimentov na minimum.

Samotný vývoj simulátora bol riadený podľa požiadaviek tímu ATS a jeho postup je znázornený v priatej kapitole.

Kapitola 5

Postup tvorenia simulátora

Táto kapitola opisuje postupný vývoj aplikácie, jednotlivé procesy a rozhodnutia, ktoré bolo nevyhnutné vykonať počas vývoja simulátora.

5.1 Základný návrh

Na začiatku bolo treba vypracovať čo najpodrobnejší softvérový návrh aplikácie. Medzi základné podúlohy spadalo stanovenie technológií, návrh predbežného grafického návrhu a softvérovej architektúry aplikácie.

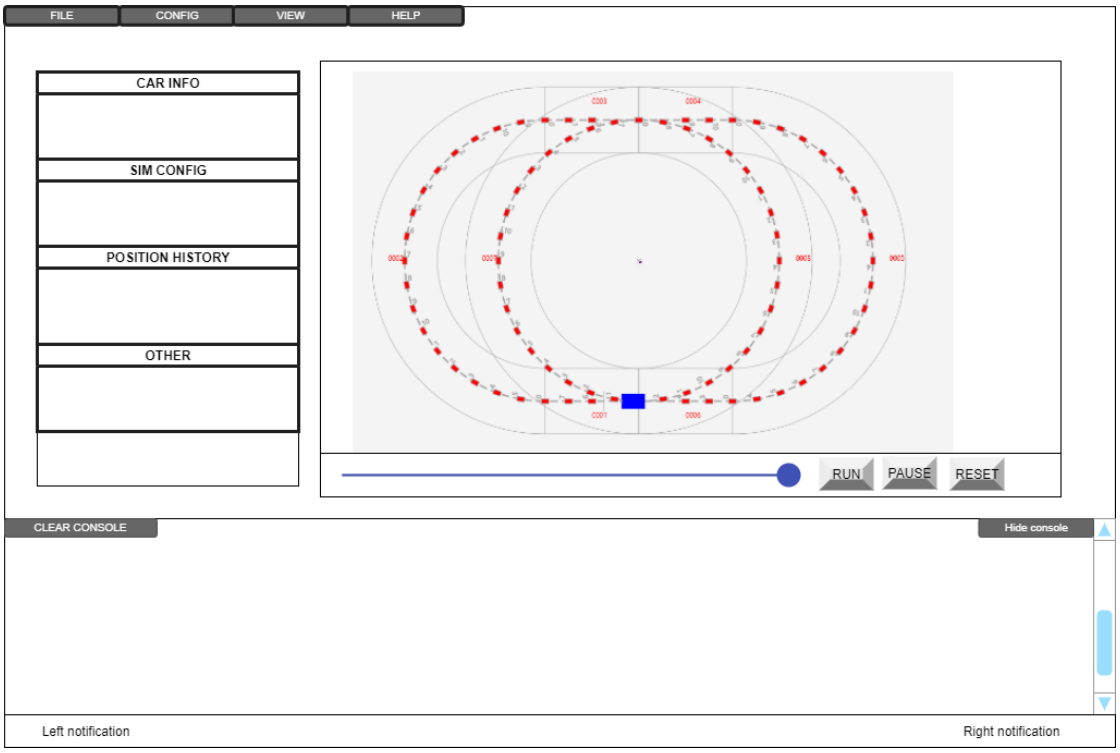
5.1.1 Technológia

Simulátor je naprogramovaný v programovacom jazyku Java. Táto technológia bola vybraná z dôvodu jej popularity, veľkej komunity a množstva dostupných knižníc vysokej platformovej nezávislosti a kvalitnej správy pamäte. Ako buildovací a balíčkovací softvér bol vybraný Maven. Užívateľské rozhranie je naprogramované pomocou knižnice JavaFX a pomocnej aplikácie Scene Builder. Na správu externých súborov XML a JSON softvér používa knižnicu Jackson. Na priamu a rýchlu komunikáciu simulátora s riadiacim

systémom je využitá bežná komunikácia cez TCP/IP sockety. Tento spôsob bol vybraný aj z dôvodu, že riadiaci systém už mal vybudované rozhranie pre danú technológiu. Je vysoko efektívny, keďže nie je obalený zbytočnými vrstvami, je najštandardnejším a najrozšírenejším protokolom a dovoľuje beh simulátora kdekoľvek na Internete, čiže aj inde ako beží riadiaci softvér vozidla. Celý softvérový vývoj bol verziovaný pomocou verziovacieho nástroja GIT a verziovacieho repozitára Github. Odkaz sa nachádza v prílohách.

5.1.2 Grafický návrh

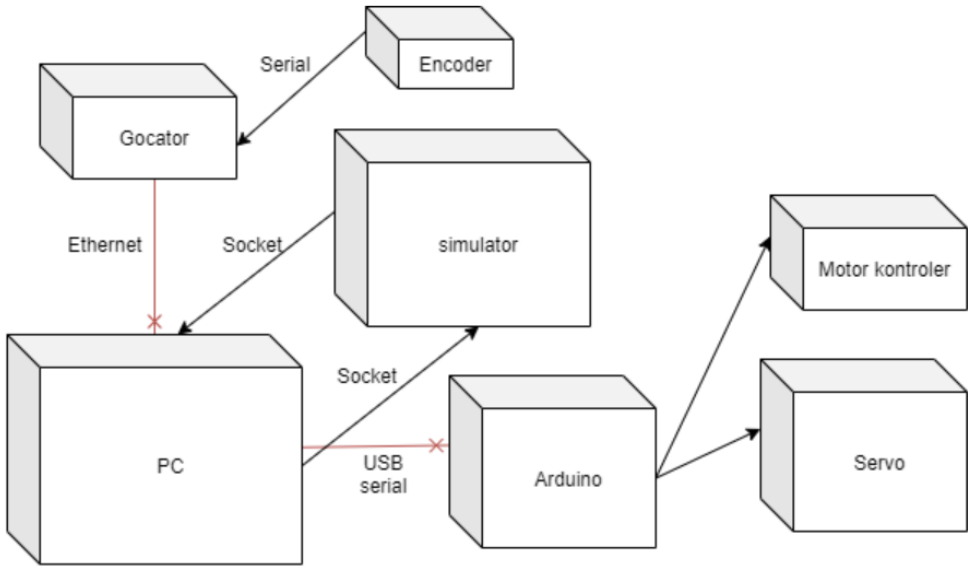
Obr. 5.1 znázorňuje prvotný grafický návrh aplikácie. Dôraz bol kladený najmä na to, aby bolo možné sa dostať ku všetkým potrebným veciam na jeden klik a súčasne bolo vidieť čo najviac informácií. Najväčšiu časť tvorí vizualizácia simulácie. Pod ňou sa nachádza časová os so základným ovládacími tlačidlami simulácie. Medzi základné ovládacie tlačidlá patri run, pause, reset. Naľavo je zobrazené kontextové rozbaľovacie menu. Jednotlivé položky sa dajú zobraziť a skryť. Nižšie sa nachádza scroolovacia konzola na zobrazovanie všetkých logov z aplikácie. Na spodu aplikácie sa nachádzajú identifikačné lišty.



Obr. 5.1: Grafický návrh užívateľského rozhrania

5.1.3 Objasnenie simulátora ako komponentu

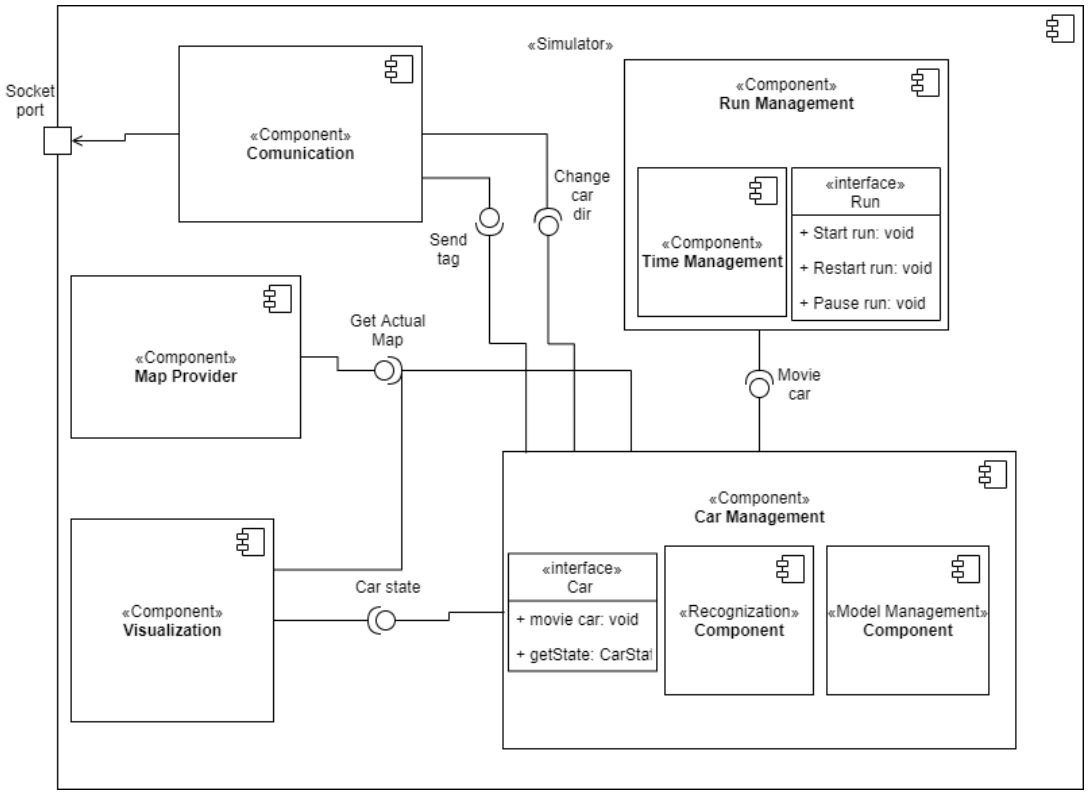
Nasledujúci Obr. 5.2 znázorňuje deployment diagram, ktorý objasňuje, ktoré komponenty simulátor nahrádza. V komponente PC je spustená samotná aplikácia riadenia vozidla. Táto aplikácia prijíma údaje potrebné na rozpoznanie značky z Gocatora. V simulačnom móde je Gocator odpojený a informácie o rozpoznaní značky posiela Simulátor. Riadiaci systém následne ovládal pohyb vozidla tým, že posiela potrebné informácie do Arduina. V simulačnom móde sa tieto údaje posielajú do simulátora.



Obr. 5.2: Deployment diagram

5.1.4 Architektúra

Aplikácia má monolitickú architektúru a skladá sa z menších modulov, ktorých základná funkcionalita je znázornená na nasledujúcom zjednodušenom diagrame zloženej štruktúry.



Obr. 5.3: Diagram zloženej štruktúry

Komponent Visualization má na starosti vykreslovanie mapy, vozidla a správu užívateľského prostredia. Komponent Run Management má na starosti správu aktuálneho zbehu simulácie, jej spustenie, zastavovanie a ukladanie. Tento modul obsahuje modul na správu času simulácie. Komponent Car Managment sa stará o všetko spojenie s modelom a simuláciou vozidla. Obsahuje subkomponenty, ktoré sa starajú o rozpoznávanie tagov a komponent, ktorý spravuje modely automobilu. Komunikačný modul má za úlohu komunikovať s aplikáciou riadenia vozidla.

Vlákna aplikácie

Základný beh aplikácie prebieha na 3 vláknach:

1. Hlavný časovač : tento časovač beží počas celej aplikácie a je navrhnutý formou návrhového vzoru Observer, kde sa zvyšné triedy môžu registrovať na tick buď každú sekundu, alebo každých 100 milisekúnd.

2. JavaFx vlákno (užívateľské prostredie). Vlákno, ktoré sa stará o chod užívateľského prostredia a užívateľský vstup.

3. Vlákno slúžiace na príjem správ v Comunication komponente. Toto vlákno neustále počúva na príjem nových dát.

Ďalšie vlákna má aplikácia pripravené v tzv. Threed Poole, ktoré sú pripravené na nové časovače potrebné pri rozpoznávaní - pozri časť Analytické rozpoznávanie tagov.

5.1.5 Context Aplikácie

Na jednoduchší prístup medzi jednotlivými modulmi v aplikácii vznikol tzv. Context. Vznikol kombináciou z navrhovaného vzoru prototypu a využíva princíp singletonu na prístup k jednotlivým komponentom. Týmto spôsobom sa dá napríklad jednoducho dostať k jednotlivým komponentom užívateľského prostredia. Stačí aby definovali interface Contextable. Priamo z Contextu je možné pristúpiť aj ku globálnemu nastaveniu aplikácie, ktoré sa načítava z konfiguračného súboru.

5.2 Vytvorenie prvého prototypu

Prvá verzia simulátora nekomunikovala zo systémom ATS. Jej úlohou bolo iba simulovať dráhu auta a vizualizovať. Táto verzia už vedela správne spracovať virtuálnu mapu. Automobilu sa nastavovali jednotlivé parametre ručne.

5.3 Mock vstupu

Následne vznikol mock riadiaceho systému ATS. Išlo o nezávislú aplikáciu, ktorá sa vedela pripojiť na simulátor pomocou socket spojenia, ako to bolo navrhnuté pre spojenie s riadiacim systémom. Išlo o konzolovú aplikáciu, kde užívateľ vedel zadať správu o zmene rýchlosti alebo smeru pre simulátor. Táto aplikácia je momentálne neudržiavaná.

5.4 Prepojenie so systémom

Systém riadenia bolo treba preprogramovať tak, aby vedel nahradiť komunikáciu s Gocatorom a Arduinm za simulátor. Na túto komunikáciu bolo použité spojenie cez socket. Používateľ v riadiacom systéme môže vstupom v konzole automobilu v simulácií nastaviť rýchlosť, zastaviť ho alebo reštartovať. Riadiaci systém počúva na správu o rozpoznaní tagu zo simulátora, následne spracuje všetky informácie a pošle do simulátora signál ako má zmeniť natočenie prednej nápravy.

5.5 Testovanie

Simulátor bol testovaný na operačnom systéme Windows 10 a Ubuntu 18.04.4 LTS. Manuálne sa vyskúšali všetky požadované funkcie a potvrdila sa správnosť návrhu.

5.6 Pridanie dodatočnej funkcionality

Na záver vývoja aplikácie boli pridané dodatočné funkcie pridávania šumu - pozri podkapitolu 6.7. Súčasťou ďalších funkcionalít bolo doladenie grafického rozhrania aplikácie a pridanie možnosti viacerých parametrov, ktoré bolo možné nastavovať priamo v konfiguračnom súbore.

Kapitola 6

Funkcie aplikácie

Nasledujúca kapitola sa venuje popisu jednotlivých funkcií a využitia aplikácie simulátora. Všetky následujúce funkcie boli odskúšané a aj úspešne implementované.

6.1 Načítanie a vykreslenie mapy

Simulátor spracuje virtuálnu mapu, ktorá je uložená vo formáte JSON. Simulátor vykreslí jednotlivé komponenty virtuálnej mapy. Aplikácia je schopná vykresliť k mape aj pomocné objekty ako je napríklad stred cesty (konkrétne spojovacia čiara medzi tagmi) alebo pomyslenú cestu. Používateľ má možnosť pre sprehľadnenie a zjednodušenie skryť niektoré časti. Napríklad si môže používateľ zobraziť iba samotné identifikačné tagy alebo skryť jednotlivé identifikátory. Jednotlivým častiam je možné zmeniť farbu v konfiguračnom súbore. Počas behu aplikácie je možné zmeniť mapu, prípadne nastaviť defaultnú mapu v nastaveniach aplikácie. Aplikácia umožní spustiť všetky validné mapy, ktoré sa nachádzajú v priečinku data/maps. Aplikácia sa na začiatku pokúsi mapu naškálovať na čo najväčšiu plochu. Stred mapy má

súradnice 0.0. Po prechode myšou ponad mapu sa zobrazuje ľavom dolnom rohu aplikácie vzdialenosť od stredu po kurzor. Po kliknutí sa zobrazí do aplikačnej konzoly najbližší tag, ak je dostatočne blízko.

6.2 Približovanie

Pre lepšiu čitateľnosť jednotlivých častí mapy a miest, kde sa aktuálne nachádza automobil, sa dá celá mapa približovať pomocou kolieska na myši.

6.3 Pohyb auta

Model auta sa pohybuje po sústredných kružniciach alebo po priamke. Jedna kružnica pre zadné koleso a jedna pre predné koleso. Tieto kružnice sú znázornené aj na mape a je možné ich vypnúť v konfiguračnom súbore. Pohyb auta je závislý od času a rýchlosti a na jeho vzdialenosť nevplýva čas potrebný na iné výpočty v aplikácii. Pohyb sa počíta podľa natočenia prednej nápravy. Informáciu ako zatočiť nápravu dostáva simulačný systém z externej aplikácie, ktorá riadi aj reálne vozidlo. Vozidlo je možné manuálne po mape posúvať a natáčať pomocou technológie drag and drop. Na začiatku nového pokusu sa auto vždy umiestni na začiatok. Aktuálne informácie o stave vozidla sa dajú pozrieť v kontextovom menu v záložke CarInfo. Model auta môže mať viacej rýchlostných stupňov. Informácie o zmene medzi jednotlivými rýchlosťami prichádza taktiež z externého systému.

6.4 Rozpoznávanie tagov

Rozpoznávanie značiek simuluje zariadenie Gocator. Na mape je znázornený lúč, ktorý keď pretne značku tak zaznamená jej rozpoznanie. Následne

pošle do výpočtového systému informácie o rozpoznaní, a to v akom čase ju rozpoznal, pod akým uhlom, a v akej vzdialenosti od stredu lúča. Históriu rozpoznaných značiek je možné pozrieť v kontextovom menu v záložke Recognized History. V aplikácií sú dva druhy rozpoznávania tagov, ktoré je možné prepnúť v konfiguračnom súbore. Prvý režim rozpoznáva značky v každom kroku pohybu auta a druhý ich rozpozná vždy dopredu a vypočíta za aký čas sa k tagu dostane. Bližší popis je uvedený v podkapitole 7.4.

6.5 Možnosť výberu áut

Aplikácia je univerzálna a vie pracovať s rôznymi modelmi áut. Na výber sú autá, s ktorými sme pracovali v laboratóriu plus fiktívne vozidlo, ktoré sa načítava z konfiguračného súboru. Na pridanie nového vozidla stačí implementovať potrebný interface. Jednotlivé modely sa líšia:

- svojimi rýchlostnými stupňami,
- vzdialenosťou medzi nápravami,
- vzdialenosťou medzi nápravou a lúčom Gocatora,
- šírkou lúča Gocatora,
- silou servo motora na otáčanie kolesa.

6.6 Časová os

Jednotlivé spustenia je možné sledovať na časovej osi. Každý beh má svoj vlastný čas, ktorý je možné zastaviť, zrýchľovať alebo posunúť. Keď si potrebujeme overiť, kde sa autíčko nachádzalo v minulosti, vieme sa na toto

miesto jednoducho pozrieť pomocou časového posuvníka. Čas vieme zrýchliť pomocou posuvníka, ktorý je umiestnený v kontextovom menu v záložke App.

6.7 Pridávanie šumu

Model pohybu a rozpoznávania je síce deterministický, ale môžu sa pridať náhodné javy formou pridaného šumu. Toto umožňuje simulovať rôzne nedostatky reálneho vozidla, dráhy alebo vplyvy okolia. Príkladom môže byť slabá batéria, prešmyknutie kolesa, nečistoty na vozovke atď. Pri pridávaní šumu máme na výber šum s Uniformným rozdelením a šum s Gaussovým rozdelením.

Simulátor dodáva nasledovné možnosti šumu:

- náhodná zmena smeru vozidla (vozidlo sa v ľubovoľný čas pootočí do ľubovoľného smeru),
- náhodné pribrzdenie (vozidlo sa v náhodný čas pohne pomalšou rýchlosťou),
- náhodné zoslabenie sily signálu na pootočenie kolies (simulácia transformuje PWM signál na stupne vynásobené náhodným zmenšujúcim koeficientom),
- náhodné vynechanie tagu (vozidlo preskočí tag),
- náhodné zaslanie chybného tagu (vozidlo pošle simulátoru iný tag ako sa aktuálne nachádza pod skenerom),
- náhodné zaslanie chybných doplnkových informácií o tagu (simulátor pošle pozmenenú informáciu o tagu, ktorá slúžia na dopočítanie novej

dráhy. Pozmení uhol pod akým naskenoval značku alebo zmení vzdialenosť značky od stredu lúča).

Všetky náhodné hodnoty sa dajú škálovať pomocou posuvníka od hodnoty 0 (vypnuté) až po hodnotu 100 (veľmi časté).

6.8 Logovanie do konzoly

Všetky dôležité informácie o behu aplikácie a jej fungovaniu sa zobrazujú v konzole aj s časom ich vyvolania. Konzola sa nachádza na spodnej strane aplikácie. Existujú tri druhy správ - Error, Warning, Info. Konzola sa dá priebežne kvôli prehľadnosti premazať, zväčšiť alebo skryť. V prípade potreby je možné obsah stiahnúť do textového súboru.

6.9 Ukladanie a spúšťanie jednotlivých zbehov

Jednotlivé testovania v aplikácií sú rozdelené do tzv. spustení (Run). Jedno spustenie obsahuje informácie o tom, v akom čase a konfigurácií sa nachádzalo autíčko na jednotlivých tagoch a stavoch vozidla po prijatí informácie od výpočtového systému. Po kliknutí na tlačidlo finish sa aktuálny zbeh uloží a je možné sa k nemu vrátiť v kontextovom menu v záložke Run History. Celý aktuálny set zbehov je možné uložiť, alebo načítať zo súboru.

6.10 Jednotky v simulátore

Simulátor počíta s nasledovnými jednotkami:

- vzdialenosť v metroch,
- uhly v stupňoch,
- čas v sekundách (prípadne v milisekundách),
- rýchlosť v m/s.

6.11 Konfigurovateľnosť

Aplikácia je naprogramovaná čo najuniverzálnejšie a väčšina vecí sa dá zmeniť bez zásahu v kóde pomocou konfiguračného súboru. Hodnoty z konfiguračného súboru je možné si pozrieť v hlavnom menu v záložke config/view all properties.

6.12 Prepojenosť so systémom

Na našu simuláciu sa prepája riadiaci systém. Simulačná aplikácia je pripravená na pripojenie od spustenia. Úspešné spojenie môžeme sledovať pomocou správy v konzole, prípadne v kontextovom menu pod záložkou Connection Info. V tejto záložke sa nachádzajú aj informácie, na akom porte aplikácia čaká na pripojenie a kedy prebehla posledná komunikácia. V prípade straty spojenia, aplikácia reštartuje svoj komunikačný modul a je pripravená na nové spojenie.

Kapitola 7

Simulácia vozidla

Nasledujúca kapitola sa venuje spracovaniu modelu auta. Objasňuje spôsob výpočtu pohybu vozidla a spôsob simulácie rozpoznávania značiek. V kapitole je vysvetlená aj reprezentácia modelu.

7.1 Model vozidla

Aplikácia je schopná pracovať s viacerými modelmi vozidiel. Pri zaradení nového modelu vozidla je potrebné pridať novú triedu, ktorá implementuje rozhranie modelu. Metódy tohto rozhrania vracajú hodnoty:

- šírka lúča - táto hodnota znázorňuje aké rozpätie v metroch dokáže daný model naskenovať,
- čas potrebný na skenovanie tagu a jeho spracovanie,
- vzdialenosť medzi prednou a zadnou nápravou v metroch,
- vzdialenosť medzi prednou nápravou a lúčom,
- maximálne možný uhol natočenia kolesa,

- rýchlosť pre zadaný rýchlostný stupeň - rýchlosť sa udáva v metroch za sekundu,
- hodnoty natočenia kolesa pre zadaný signál - vstup je PWM signál a výstup je uhol zatočenia v stupňoch (táto funkcia má simulovať servo motor, ktorý ohýba prednú nápravu).

7.2 Stav vozidla

Aktuálny stav vozidla znázorňuje trieda CarState, ktorá spĺňa konvencie JavaBeam a jej parametre sú:

- pozícia referenčného bodu vozidla (stred prednej nápravy), relatívne k stredu mapy - vzdialenosť sa udáva v metroch,
- uhol natočenia celého vozidla voči severu (vrch) - používané jednotky sú stupne a uhol sa počíta proti smeru hodinových ručičiek,
- uhol natočenia prednej nápravy voči vozidlu - používané jednotky sú stupne a hodnota 0 značí, že automobil je nasmerovaný rovno, kladná hodnota predstavuje doľava a záporná hodnota doprava,
- aktuálny stupeň rýchlosti,
- bod zadnej nápravy, ktorý slúži pre pomocné výpočty a vizualizáciu.

7.3 Model a funkcie pohybu

Pohyb vozidla zabezpečuje metóda movie (double actualTime, double passingMovementTime) triedy CarManagement. Táto metóda posunie pozíciu vozidla po dráhe o takú vzdialenosť, akú by vozidlo prešlo za čas rovný

rozdielu vstupných časov metódy. Toto nám zabezpečuje to, že aj keď na tom istom výpočtovom vlákne prebiehajú aj iné výpočty, tak neovplyvňujú daný pohyb.

Aktuálne je pohyb zjednodušený na pohyb bicykla, čiže nie je potrebná šírka nápravy. Toto zjednodušenie je vhodné z dôvodu, že dráha bicykla sa v zákrute správa z hľadiska prejdenej trajektórie identicky ako auto jazdiace podľa Ackermannovho riadenia [15].V prípade potreby iného pohybového modelu stačí v aplikácií prepísať spomínanú metódu movie().

Následne body sa venujú jednotlivým výpočtom potrebných na simulovanie pohybu modelu vozidla.

7.3.1 Posun rovno

Keď má vozidlo vyrovnanú nápravu jeho pohyb simulujeme lineárnym posunom, kde prejdená vzdialenosť sa rovná zmene času vynásobeného rýchlosťou auta.

$$x' = x + (prejdenaVzdialenost \cdot sin(uhol_auta_voci_severu))$$

y'= y - (prejdenaVzdialenost $\cdot cos(uhol_auta_voci_severu)$)

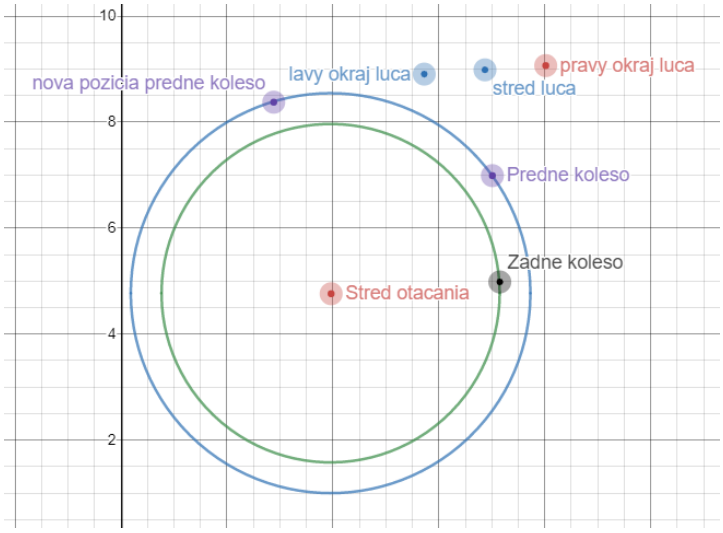
7.3.2 Pohyb zatočeného vozidla

Keď má vozidlo pootočenú nápravu tak sa pohybuje po sústredných kružniciach. Jedna kružnica prechádza stredom prednej nápravy a druhá zadnej. Tieto body sú kolmé na stred otáčania.

Vizualizácia pohybu

Pohyb automobilu bol vymodelovaný v modelovacom nástroji Desmos. Link môžeme vzhliadnuť v prílohe. Na snímke obrazovky je vidieť ukážku.

Vďaka Plotrovaciemu nástroju si vieme lepšie predstaviť pohyb a závislosti medzi jednotlivými vstupmi. V ukážke je jasne vidieť ako sa vozidlu mení dráha na základe zmeny vstupných parametrov. Súčasne je možnosť si pozrieť o aký uhol sa vozidlo posunie a akú vzdialenosť vozidlo prejde. V aplikácií Desmos je možné nastaviť aj vykreslenie polohy lúča pomocou jeho krajných bodov. Ukážka je na Obr. 7.1.



Obr. 7.1: Desmos vizualizácia

Výpočet polomeru otáčania

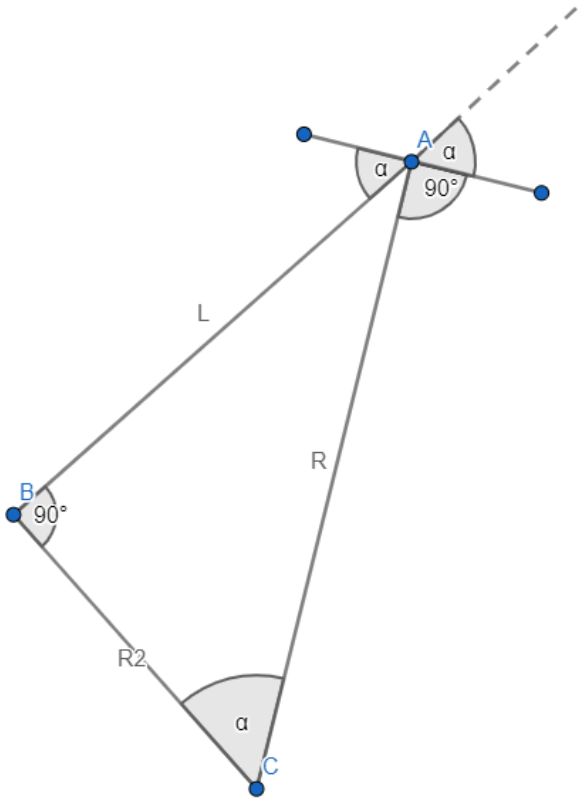
Polomer otáčania prednej nápravy sa počíta v metóde computeWheelRadius(). Na Obr. 7.2 je zreteľne vidieť nasledujúce body:

A - stred prednej nápravy (referenčný bod vozidla),

B - stred zadnej nápravy,

C - stred otáčania (A, B sú kolmé body na C),

α - uhol natočenia kolesa prednej nápravy.



Obr. 7.2: Výpočet polomeru

Tým, že sa obidve kolesá pohybujú po kružnici otáčania majú rovnaký smer ako dotyčnica ku kružniciam, a teda sú kolmé na svoju spojnicu zo stredom. Uhol ∠ BCA je totožný s uhlom zatočenia kolies, čo vyplýva z vlastností trojuholníkov. Obr. 7.2 znázorňuje vznik pravouhlého trojuholníka, pri ktorom platí:

$$sin(\alpha) = \frac{protilahlaStrana}{prepona} = \frac{L}{R}$$

Z vyššie uvedeného vieme odvodiť vzdialenosť bodov C a B, ktorá sa rovná polomeru otáčania prednej nápravy:

$$R = \left| \frac{L}{sin(\alpha)} \right|$$

Výpočet stredu otáčania

Súradnice stredu otáčania sa počítajú v metóde computeCenterOfTurn(). Ako nám obrázok znázorňuje, γ sa rovná súčtu uhlov α (zatočenia nápravy auta) a β (natočenia auta voči severu). Podľa vlastnosti trojuholníka sa γ rovná aj $\angle$ ASX.

$$\gamma = \alpha + \beta$$

Aby sme získali súradnice stredu otáčania tak potrebujeme zistiť vzdialenosti |SX| a |AX|. Z definície cosinusu platí rovnosť:

$$\triangle SAX : cos(\gamma) = \frac{|SX|}{R}$$

Za R (polomer) dosadíme L/sin(α) z predchádzajúceho výpočtu a dostaneme nasledujúci vzorec:

$$cos(\gamma) = \frac{|SX|}{L} \cdot sin(\alpha)$$
$$\Rightarrow$$
$$|SX| = L \cdot \frac{cos(\gamma)}{sin(\alpha)} = L \cdot \frac{cos(\alpha+\beta)}{sin(\alpha)}$$

Z definície sinusu platí rovnoť:

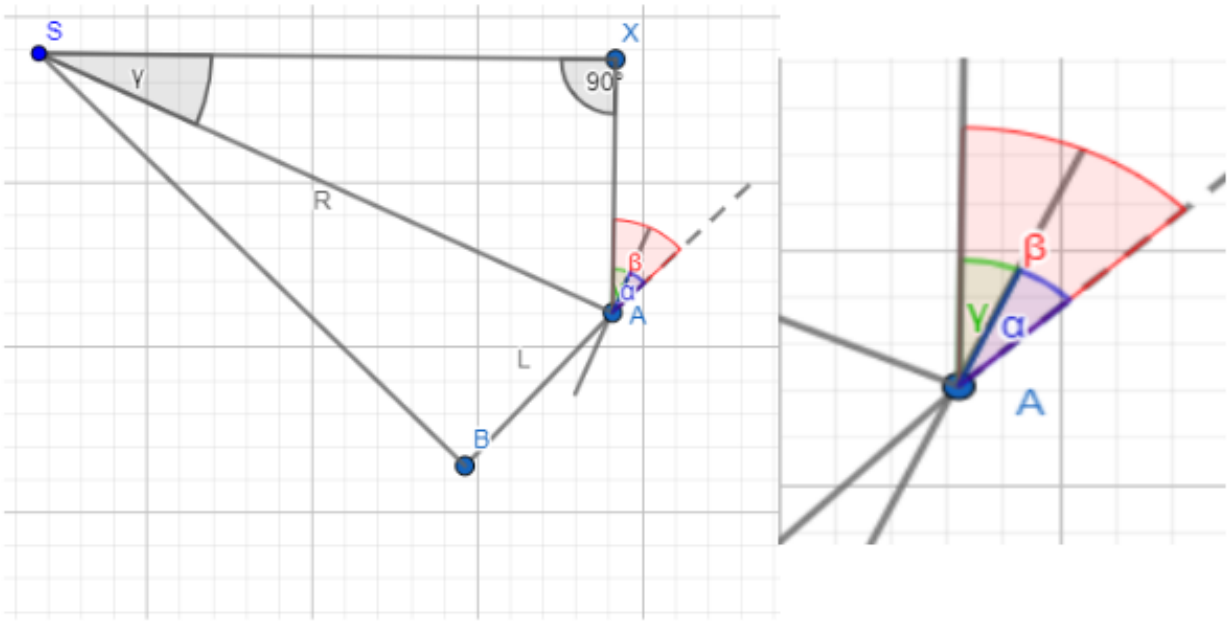
$$\triangle SAX : sin(\gamma) = \frac{|AX|}{R}$$

Za R (polomer) dosadíme L/sin(α) z predchádzajúceho výpočtu a dostaneme nasledujúci vzorec:

$$\triangle SAX : sin(\gamma) = \frac{|AX|}{L} \cdot sin(\alpha)$$
$$\Rightarrow$$
$$|AX| = L \cdot \frac{sin(\gamma)}{sin(\alpha)} = L \cdot \frac{sin(\alpha+\beta)}{sin(\alpha)}$$

Keď máme vzdialenosti |SX| a |XA|, potom vieme vyjadriť súradnice stredu. Od súradnice X odpočítame |SX| a od súradnice Y odpočítame |AX|.

$$S[x] = A[x] - L \cdot \frac{cos(\alpha+\beta)}{sin(\alpha)}$$
$$\text{S[y]} = \text{A[y]} - \text{L} \cdot \frac{sin(\alpha+\beta)}{sin(\alpha)}$$



Obr. 7.3: Výpočet Stredu

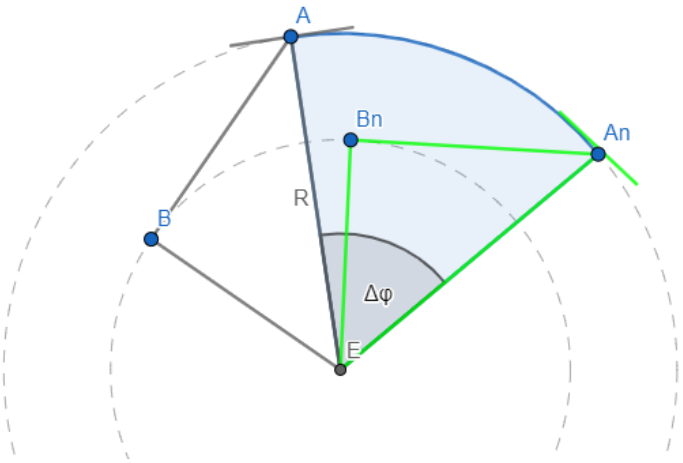
Výpočet uhlu

Na výpočet uhlu, ktoré vozidlo prejde po kružnici použijeme nižšie uvedený vzorec. Platí, že prejdená vzdialenosť sa rovná aktuálnej rýchlosti vozidla vynásobenej časom.

$$\Delta\varphi = \frac{prejdenavzdialenost \cdot 180}{\pi \cdot R}$$

Výpočet novej pozície

Výpočet novej pozície vychádza z rotácie polohového vektoru predného kolesa okolo stredu otáčania.



Obr. 7.4: Výpočet posunu

Použijeme nasledujúci vzorec, ktorý upravíme tak aby zodpovedal rotácií okolo stredu otáčania.

x,y = súradnice bodu A (stred nápravy)
x',y' = nové súradnice bodu A

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{pmatrix} cos(\Delta\varphi) & -sin(\Delta\varphi) \\ sin(\Delta\varphi) & cos(\Delta\varphi) \end{pmatrix} \cdot \begin{bmatrix} X \\ Y \end{bmatrix}$$

$$\Rightarrow$$

$$\text{x'} = \text{x} \cdot cos(\Delta\varphi) - y \cdot sin(\Delta\varphi)$$
$$\text{y'} = \text{x} \cdot sin(\Delta\varphi) + y \cdot cos(\Delta\varphi)$$

Nová pozícia referenčného bodu vozidla A v čase t je:

$$\mathrm{A(t)} = \begin{pmatrix} (X - Xs) \cdot cos(\Delta\varphi) + sin(\Delta\varphi) \cdot (Y - Ys) \\ (Y - Ys) \cdot cos(\Delta\varphi) - sin(\Delta\varphi) \cdot (X - Xs) \end{pmatrix} + \begin{pmatrix} Xs \\ Ys \end{pmatrix}$$
$$\Rightarrow$$
$$\begin{pmatrix} (X - Xs) \cdot cos(\Delta\varphi) + sin(\Delta\varphi) \cdot (Y - Ys) + Xs \\ (Y - Ys) \cdot cos(\Delta\varphi) - sin(\Delta\varphi) \cdot (X - Xs) + Ys \end{pmatrix}$$

7.4 Rozpoznávanie značiek

Rozpoznávanie značiek slúži ako simulácia zariadenia Gocator. V simulátore sú naprogramované dve verzie rozpoznávania, a to: Iteratívna verzia rozpoznávania a Analytická verzia rozpoznávania. Je možnosť prepnutia sa medzi týmito spomínanými nastaveniami v konfiguračnom súbore.

7.4.1 Iteratívna verzia rozpoznávania

Táto metóda je jednoduchšia, ale výpočtovo náročnejšia. Pri každej zmene vozidla sa kontroluje, či sa aktuálne lúč nenachádza nad stredom ľubovoľného tagu. K tomuto výpočtu potrebujeme vypočítať body hranice lúča.

1. $\mathrm{L} = \left(Ax - Z \cdot \sin(D) + \frac{l}{2}\cos(D), Ay + Z \cdot \cos(D) + \frac{l}{2}\sin(D)\right)$
2. $\mathrm{P} = \left(Ax - Z \cdot \sin(D) - \frac{l}{2}\cos(D), Ay + Z \cdot \cos(D) - \frac{l}{2}\sin(D)\right)$
3. $\mathrm{S} = (Ax - Z \cdot \sin(D), Ay + Z \cdot \cos(D))$

Z vyššie uvedených vzorcov predstavuje písmeno L ľavý okraj lúča, P pravý okraj lúča a S stred lúča. Ďalej súčasťou vzorcov je písmeno A predstavujúce stred prednej nápravy, Z vzdialenosť lúča od nápravy a D uhol natočenia vzhľadom na sever.

Následne potrebujeme určiť vzdialenosť medzi úsečkou definovanú krajnými bodmi lúča (L, P) a stredom tagu (bod T). Pri výpočte som sa inšpiroval matematikom Paul Bourke [8]. Rovnica úsečky je znázornená nasledovne:

$$p = L + u \cdot (P - L)$$

Bod T je najbližšie k úsečke p v dotyčnici priamky p, ktorý prechádza bodom T. To znamená, že bodový súčin dotyčnice a priamky je 0.

$$(T - p) \cdot (P - L) = 0$$

Po substitúcií dostaneme nasledujúci vzorec:

$$[T - L - u(P - L)] \cdot (P - L) = 0$$
$$\Rightarrow u = \frac{(T[x]-L[x])\cdot(P[x]-L[x])+(T[y]-L[y])\cdot(P[y]-L[y]))}{\|P-L\|^2}$$

Substitúciou do rovnice priamky dostaneme bod priesečníka dotyčnice. Označíme ho bodom M.

$$M[x] = L[x] + u \cdot (P[x] - L[x])$$
$$\mathrm{M[y]} = \mathrm{L[y]} + \mathrm{u}\cdot(P[y] - L[y])$$

Vzdialenosť medzi Lúčom a Stredom značky (T) sa rovná vzdialenosti medzi bodmi M a T.

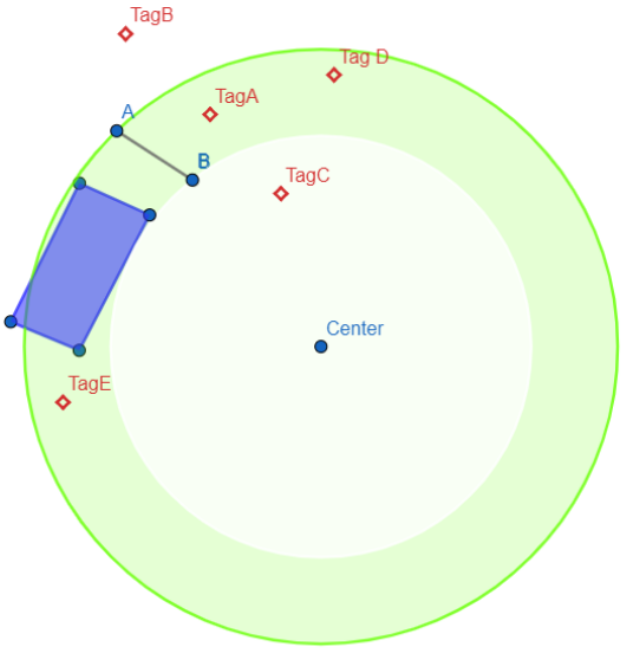
Zoznam tagov vieme načítať z virtuálnej mapy. Táto metóda rozpoznávania má najväčšiu nevýhodu v tom, že pokiaľ ide autíčko príliš rýchlo môže sa stať, že v jednom kroku pohybu preskočí celú značku. Z toho vyplýva, že sa simulácie zložitejšie škálujú.

7.4.2 Analytická verzia rozpoznávania

Analytické riešenie je vhodnejšie pretože výpočet sa vykonáva menej krát. Podstata riešenia je, že automobil si vždy pri zmene smeru jazdy vypočíta najbližšie tagy, ktoré pretne a spracuje ich až v čase keď sa k nim priblíži. Výpočet opakuje len vtedy keď užívateľ manuálne zmení parametre auta, alebo príde nová informácia z riadiaceho systému.

Výpočet najbližších tagov

V prvom rade vyfiltrujeme, ktoré tagy sa nachádzajú pred automobilom. Príklad je popísaný na Obr. 7.5. Body, ktoré sa rozpoznajú pri tejto konfigurácii zatočenia sú v zelenej časti a musia sa teda nachádzať medzi kružnicami s polomerom |Center, A| a |Center, B| so stredom Center. Taktiež musíme vyfiltrovať tagy, ktoré sa nachádzajú za vozidlom. Pre zjednodušenie nám stačí zahodiť všetky body, ktoré sú bližšie k autu ako k lúču a tie, ktoré sa nachádzajú medzi autom a lúčom, čiže ich vzdialenosť k autu je menšia ako vzdialenosť auta od lúča.



Obr. 7.5: Rozpoznávanie 1

Zisťovanie uhlu medzi vozidlom a tagom

V ďalšom kroku, je potrebné zistiť uhol medzi vozidlom a tagom. Príklad je popísaný na Obr. 7.6. Na obrázku vidíme bod T, ktorý znázorňuje stred lúča. Využijeme trojuholník so stranami a, b, c, ktorý nám vo vnútri vznikol a rovnosť $c^2 = a^2 + b^2 - 2 \cdot a \cdot b \cdot cos(\alpha)$.

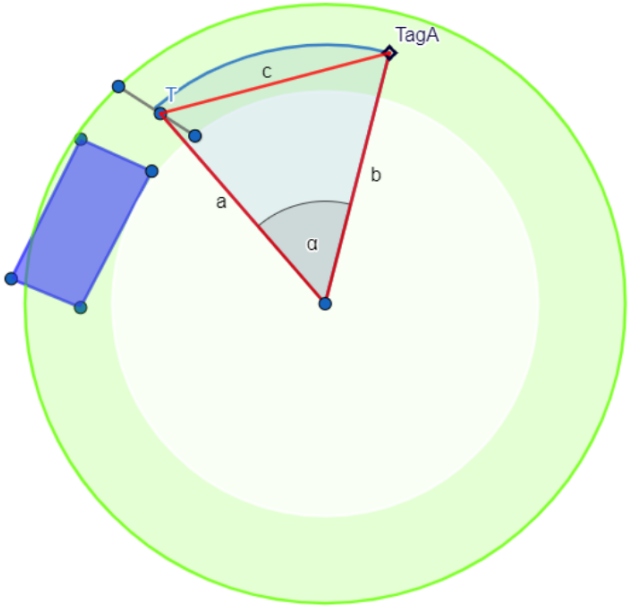
Z uvedeného vzorca si následne odvodíme vzorec:

$$\alpha = acrcos((a^2 + b^2 - c^2)/a \cdot b)$$

Nakoniec si zo vzorca na výpočet kružnicového oblúka odvodíme vzorec na čas, za ktorý sa dostane auto k tagu:

$$t = \frac{(\pi \cdot \alpha \cdot a)}{(180 \cdot rychlost)}$$

Jednotlivé časti pridáme do utriedenej dátovej štruktúry. Následne pre najbližší tag spustíme na novom vlákne časovač, ktorý odošle v správnom čase informácie o rozpoznaní značky. Jednotlive voľné vlákna sú pripravené v tzv. ThreedPoole.



Obr. 7.6: Rozpoznávanie 2

7.5 Prepočet PWM signálu

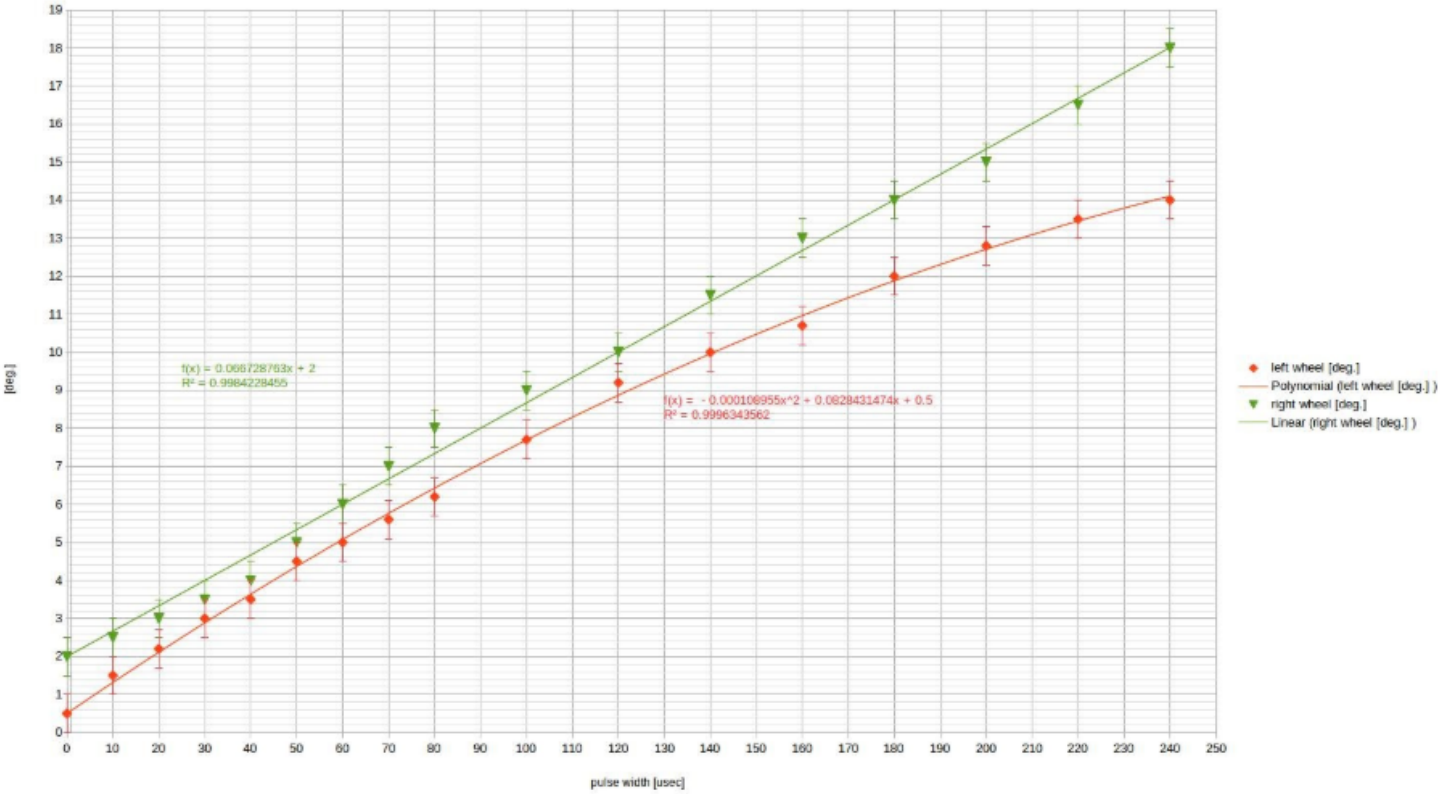
Výstupom riadiaceho algoritmu nie je uhol o aký sa má náprava natočiť, ale PWM signál, ktorý je zaslaný servo motoru na zatočenie nápravy. Z tohto dôvodu musíme tento signál v simulátore transformovať na stupne. Simulátor povoľuje dve možnosti. Prvá možnosť je lineárna funkcia, ktorej parametre sa dajú nastaviť v konfiguračnom súbore. Druhou možnosťou je použiť slovník konkrétnych asociácií medzi signálom a natáčaním. Pre tento

slovník sme využili ručne namerané údaje pre aktuálny model 1/10, ktorý sa nachádza v laboratóriu.

pulse width [usec]	**0**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**
right wheel [deg.]	2	2.5	3	3.5	4	5	6	7	8
left wheel [deg.]	0.5	1.5	2.2	3	3.5	4.5	5	5.6	6.2
pulse width [usec]	**100**	**120**	**140**	**160**	**180**	**200**	**220**	**240**	
right wheel [deg.]	9	10	11.5	13	14	15	16.5	18	
left wheel [deg.]	7.7	9.2	10	10.7	12	12.8	13.5	14	

Tabuľka 7.1: Namerané hodnoty pootočenia

Z Obr. 7.7 vyplýva, že aktuálne vozidlo zatáčalo na obe kolesá s miernou odchýlkou. Avšak hodnota pravého kolesa je takmer lineárna. 1 stupeň sa rovná približne 15 PWM. Táto funkcia musí počítať aj s tým, že každý model má iný maximálny stupeň zatočenia. Meranie vykonali: B. Sitár, M. Mereš.



Obr. 7.7: Meranie prepočtu zatočenia

Kapitola 8

Manuál

Kapitola je venovaná návodu na používanie simulátora. Popisuje minimálne požiadavky žiadúce na používanie aplikácie a zobrazuje reálnu ukážku z naprogramovanej aplikácie simulátora.

Minimálne požiadavky

Na spustenie aplikácie simulátora je potrebné mať:

- nainštalovanú Java 1.8 a novšiu,
- nastavenú premennú JAVA_HOME,
- operačný systém podporujúci grafickú knižnicu JavaFX - napríklad Windows alebo Ubuntu,
- voľnú pamäť viac ako 1,5 GB.

Návod na spustenie

Aplikáciu spustíme pomocou spúšťacieho scriptu run.sh pre operačný systém Linux alebo run.bat pre operačný systém Windows.

V prípade potreby zmeny kódu

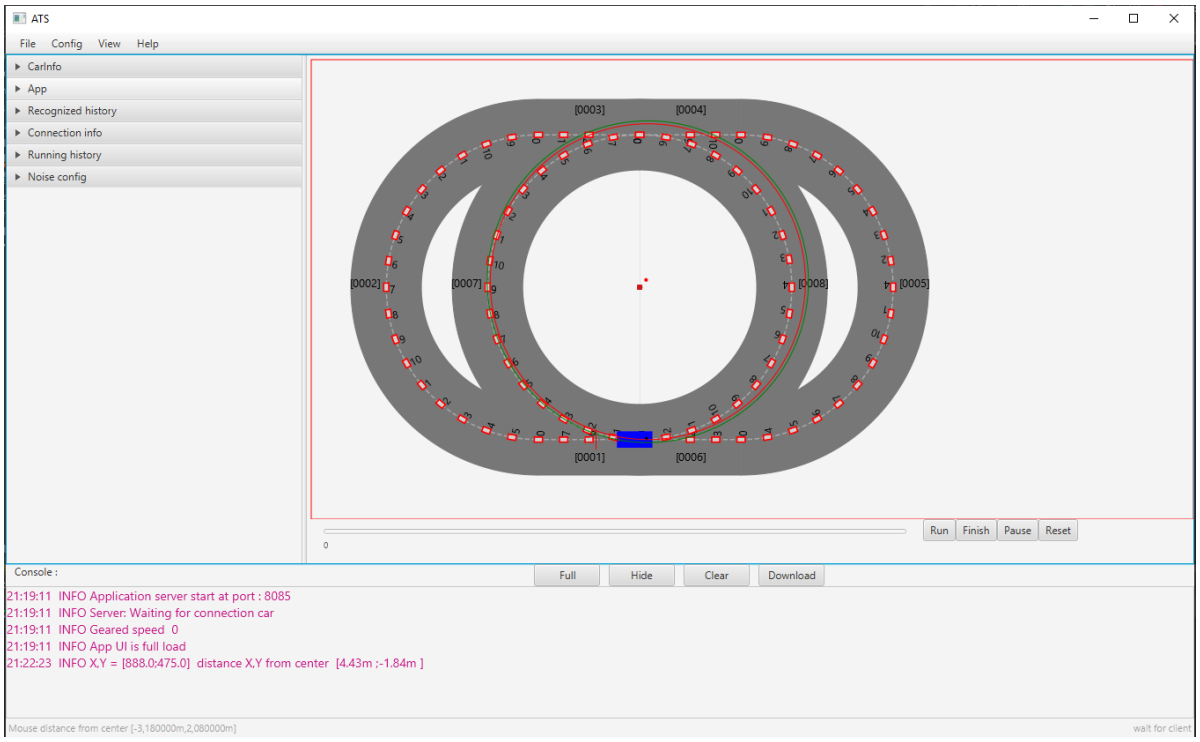
Buildovanie aplikácie je zabezpečené pomocou nástroja Maven, ktorý je potrebné mať nainštalovaný. Následne po zmene kódu stačí spustiť skript build.sh (build.bat) a nová verzia sa vybuilduje a prekopíruje do priečinka release.

Základný scenár

Pred spustením aplikácie si nastavíme konfiguračný súbor app.properties, ktorý sa nachádza v priečinku data. Dôležitou premennou je socket.port, ktorý nám nastaví na akom porte sa má na aplikáciu pripojiť riadiaci systém. Aplikáciu spustíme spúšťacím skriptom. Následne spustíme aplikáciu riadiaceho systému. O úspešnom prepojení medzi systémami nás oboznámi správa v konzole. V simulačnej aplikácií si vyberieme potrebnú mapu a model auta, prípadne nastavíme časové zrýchlenie a iné premenné. Následne v riadiacom systéme odštartujeme auto. Ak je v konfiguračnom súbore nastavená premenná app.startAfterSetPower = true tak sa simulácia automaticky spustí. Ak nie tak ju treba naštartovať tlačidlom run. Po ukončení zbehu ho uložíme tlačidlom finish. Následne môžeme zmeniť nastavenia, prípadne pustiť novú verziu riadiaceho systému a pokus opakovať. Aktuálnu prácu si môžeme uložiť v hlavnom menu pod položkou File/SaveRunningHistory.

Ukážka aplikácie

Na nasledujúcom Obr. 8.1 je zobrazená aplikácia verzie 1,5 ihneď po zapnutí.



Obr. 8.1: Ukážka aplikácie

Kapitola 9

Záver

Hlavný cieľom tejto diplomovej práce bolo podporiť vývoj Autonómneho transportného systému v Laboratóriu autonómnej mobility UK. Podarilo sa to vytvorením funkčnej aplikácie, ktorá je určená na simuláciu automobilu. Tento simulátor má tendenciu značne napomôcť pri ladení aktuálneho riadiaceho algoritmu a taktiež urýchliť vývoj nového algoritmu.

Súčasťou práce bola aj potreba upraviť hlavný riadiaci systém, ktorý je momentálne možné prepnúť do režimu simulácie. Následne softvér nekomunikuje s fyzickými časťami vozidla, ale mokuje ich práve našou aplikáciou simulátora.

Počas vývoja bolo treba spracovať matematický model pohybu vozidla. Tento pohyb simuluje pohyb vozidla s Ackermannovým riadením zjednodušene na pohyb bicykla. Spomínaný model jazdí buď rovno alebo po kružniciach. V práci je tento pohyb matematicky objasnený a vizualizovaný.

Simulované vozidlo jazdí po virtuálnej mape. Táto mapa simuluje reálnu inteligentnú vozovku, ktorá je súčasťou Autonómneho transportného systému. Na tejto vozovke sa nachádzajú riadiace značky slúžiace na orientovanie sa vozidla. Rozpoznávanie týchto značiek je taktiež simulované. Výhodou

simulácie je aj možnosť skúšania viacerých typov vozidla, čo umožní rýchlejšie nastavenie jednotlivých parametrov algoritmov a tak rýchlejšiu adaptáciu celého systému na nový typ vozidla. Celá simulácia je vizualizovaná. Jednotlivý priebeh je možné si pustiť viac krát, čo používateľom umožňuje lepšiu predstavu o simulovanej dopravnej situácii. Celý proces simulácie je možné uložiť a následne spustiť. Jednotlivé údaje sú taktiež zaznamenávané do konzoly a jeho možné exportovať do textového súboru. To umožní prípadnú dodatočnú analýzu jednotlivých spustení.

Práca obsahuje stručný popis funkcií a manuál, vďaka ktorým je aplikáciu jednoduché používať.

Simulátor bol zhotovený podľa momentálnych potrieb tímu ATS a aj vďaka nemu je projekt zaujímavou alternatívou momentálneho trendu vývoja Automatických transportných systémov.

Literatúra

[1] *Transport reviews : a transnational transdisciplinary journal.*, chapter Transport reviews : a transnational transdisciplinary journal. Taylor & Francis, London, 1981.

[2] Gocator, Apr 2018. URL: http://arisens-ingenet.com/en/gocator/.

[3] Ackermann steering geometry, May 2020. URL: https://en.wikipedia.org/wiki/Ackermann_steering_geometry#/media/File:Ackermann_simple_design.svg.

[4] Autonomous driving performance in carla simulator using reinforcement learning. *Journal of Xidian University*, 14(4), 2020. doi:10.37896/jxu14.4/275.

[5] Jas documentation, 2ooo. URL: http://krum.rz.uni-mannheim.de/jas/.

[6] James C. Alexander and J. H. Maddocks. On the maneuvering of vehicles. *SIAM J. Appl. Math.*, 48(1):38–51, February 1988. URL: http://dx.doi.org/10.1137/0148002, doi:10.1137/0148002.

[7] J. Banks, J. S. Carson, B. L. Nelson, and D. Nicol. *Discrete-Event System Simulation.* Prentice Hall, 5 edition, 2010.

[8] Paul Bourke. Points, lines, and planes, Oct 1988. URL: `http://paulbourke.net/geometry/pointlineplane`.

[9] A. Chovanec. *Modelovanie a simulácia diskrétnych stochastickỳch procesov*. Trenčianska univerzita Alexandra Dubčeka, 2004. URL: `https://books.google.sk/books?id=OASDAAAACAAJ`.

[10] José Fernando Sabando Cárdenas, Jong Gyu Shin, and Sang Ho Kim. A few critical human factors for developing sustainable autonomous driving technology. *Sustainability*, 12(7):3030, Sep 2020. `doi:10.3390/su12073030`.

[11] José Fernando Sabando Cárdenas, Jong Gyu Shin, and Sang Ho Kim. A few critical human factors for developing sustainable autonomous driving technology. *Sustainability*, 12(7):3030, Sep 2020. `doi:10.3390/su12073030`.

[12] Guillaume Dubois and Martin Lundstedt. *Modeling and simulation: challenges and best practices for industry*. CRC Press in an imprint of the Taylor Francis Group, an informa business, 2018.

[13] Kamil DUREC. Informační technologie pro modelování a simulace [online]. Diplomová práce, Masarykova univerzita, Fakulta informatiky, Brno, 2013 [cit. 2019-12-12]. URL: `DostupnézWWW<https://is.muni.cz/th/e92u2/>`.

[14] Matthew England. Computer algebra software, 2008. URL: `http://www.sigsam.org/Resources/Software.html`.

[15] G.E. Prince and S.P. Dubois. Mathematical models for motion of the rear ends of vehicles. *Mathematical and Computer Modelling*,

49(9):2049 – 2060, 2009. URL: http://www.sciencedirect.com/science/article/pii/S0895717708003804, doi:https://doi.org/10.1016/j.mcm.2008.10.005.

[16] Jan Sajdl. Ackermannova podmínka, Mar 2016. URL: https://www.autolexicon.net/cs/articles/ackermannova-podminka/.

[17] University of Vienna X staff. Artificial intelligence speeds up photodynamics simulations, Sep 2019. URL: https://phys.org/news/2019-09-artificial-intelligence-photodynamics-simulations.html.

[18] Patricia M. Stohr-Hunt. An analysis of frequency of hands-on experience and science achievement. *Journal of Research in Science Teaching*, 33(1):101–109, January 1996. doi:10.1002/(SICI)1098-2736(199601)33:1<101::AID-TEA6>3.0.CO;2-Z.

[19] Bc. Katarína Šimnová. Riadenie vozidla po dráhe osadenej 3d značkami. Diplomová práce, FMFI UK,Bratislava, 2018. URL: DostupnézWWW<https://www.itspy.cz/wp-content/uploads/2018/11/IT_SPY_2018_Diplomova_prace_10.pdf/>.

Zoznam obrázkov