VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Autonomní řízení monopostu Formula Student v simulátoru

Autonomous Driving of a Formula Student in a Simulator

Bc. Michael Ceplý

Diplomová práce

Vedoucí práce: prof. Ing. Jan Platoš, Ph.D.

Ostrava, 2026

VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Č.j.VSB/25/139663
vsboee25040AB0

Zadání diplomové práce

Student: **Bc. Michael Ceplý**

Studijní program: N0613A140034 Informatika

Téma: Autonomní řízení monopostu Formula Student v simulátoru
Autonomous Driving of a Formula Student in a Simulator

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je ve zvoleném simulátoru pro autonomní řízení vytvořit agenta, který bude řídit vozidlo monopostu Formula Student v rámci prostředí a plnit zadané úkoly. Důraz je kladen na jednoduchou replikovatelnost postupů a co nejjednodušší integraci řídících algoritmů a využití dostupných senzorů.

Práce bude obsahovat:
1. Popis zvoleného simulátoru.
2. Detailní popis vybraných metod řízení.
3. Implementaci vybraných úkolů.
4. Ověření fungování implementovaných metod na vybraných úkolech a porovnání s existujícími přístupy.

Seznam doporučené odborné literatury:

[1] HERRMANN, Andreas, Walter BRENNER a Rupert STADLER. Autonomous Driving: How the Driverless Revolution will Change the World. Emerald Publishing Limited, 2018. ISBN 978-1787148345.
[2] VENTURI, Luca a Krishtof KORDA. Hands-On Vision and Behavior for Self-Driving Cars: Explore visual perception, lane detection, and object classification with Python 3 and OpenCV 4. Packt Publishing, 2020. ISBN 978-1800203587.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **prof. Ing. Jan Platoš, Ph.D.**

Datum zadání: 01.09.2025

Datum odevzdání: 30.04.2026

Garant studijního programu: prof. RNDr. Václav Snášel, CSc.

V IS EDISON zadáno: 10.10.2025 13:01:51

17. listopadu 2172/15
708 00 Ostrava-Poruba
Česká republika

spojovatelka: +420 597 321 111
epodatelna: epodatelna@vsb.cz
ID datové schránky: d3kj88v

IČ: 61989100
DIČ: CZ61989100

email: studijni.fei@vsb.cz
www.fei.vsb.cz

Abstrakt

Tato diplomová práce se zabývá návrhem a implementací autonomního systému řízení monopostu pro soutěž Formula Student Driverless v simulátoru CARLA. Práce nejprve popisuje pravidla soutěže, porovnává dostupné simulátory pro autonomní řízení a analyzuje vlastnosti senzorů relevantních pro detekci kuželů ohraničujících závodní trať. Implementační část pokrývá celý řetězec autonomní pipeline od detekce kuželů v mračnu bodů z LiDARu, přes klasifikaci jejich barvy konvoluční neuronovou sítí, mapování trati metodou Graph SLAM, až po řízení vozidla regulátory Pure Pursuit a Stanley. Systém pracuje ve dvou režimech odpovídajících disciplínám DV Cupu – reaktivním pro první průjezd neznámou tratí a závodním s využitím vytvořené mapy. Funkčnost obou režimů je ověřena na třech testovacích tratích a výsledky jsou porovnány z hlediska času průjezdu i přesnosti sledování trajektorie. Součástí práce je také zhodnocení dosažených výsledků a porovnání s existujícími přístupy v oblasti Formula Student Driverless.

Klíčová slova

autonomní řízení, Formula Student Driverless, simulátor CARLA, LiDAR, konvoluční neuronové sítě, fúze senzorů, Graph SLAM, Pure Pursuit, Stanley, detekce kuželů

Abstract

This master's thesis describes the design and implementation of an autonomous driving system for a Formula Student Driverless race car within the CARLA simulator. The thesis first describes the competition rules, compares available simulators for autonomous driving and analyses the properties of sensors relevant for the detection of cones delimiting the race track. The implementation part covers the complete autonomous pipeline from cone detection in LiDAR point clouds, through colour classification by a convolutional neural network, track mapping using Graph SLAM, to vehicle control by the Pure Pursuit and Stanley controllers. The system operates in two modes corresponding to the DV Cup disciplines – a reactive mode for the first lap of an unknown track and a racing mode that exploits the previously built map. The functionality of both modes is verified on three test tracks, and the results are compared in terms of lap time and trajectory tracking accuracy. The work also includes an evaluation of the achieved results and a comparison with existing approaches in Formula Student Driverless.

Keywords

autonomous driving, Formula Student Driverless, CARLA simulator, LiDAR, convolutional neural networks, sensor fusion, Graph SLAM, Pure Pursuit, Stanley, cone detection

Poděkování

Rád bych na tomto místě vyjádřil upřímné poděkování vedoucímu mé diplomové práce prof. Ing. Janu Platošovi, Ph.D., za trpělivost, cenné rady a čas, který mi v průběhu zpracování ochotně věnoval. Poděkování patří také celému týmu Formula TU Ostrava za jedinečnou příležitost podílet se na tak inspirativním projektu a za zkušenosti, které mi přinesl. Největší poděkování však patří mé rodině za zázemí a trpělivou podporu po celou dobu mého dlouhého studia.

Obsah

Seznam použitých zkratek a symbolů

CARLA	–	Car Learning to Act
CNN	–	Convolutional Neural Network
CV	–	Combustion Vehicle
DBSCAN	–	Density-Based Spatial Clustering of Applications with Noise
DV	–	Driverless Vehicle
EKF	–	Extended Kalman Filter
EV	–	Electric Vehicle
FOV	–	Field of View
FS	–	Formula Student
FSDS	–	Formula Student Driverless Simulator
GNSS	–	Global Navigation Satellite System
GTSAM	–	Georgia Tech Smoothing and Mapping
IMU	–	Inertial Measurement Unit
iSAM2	–	Incremental Smoothing and Mapping 2
LiDAR	–	Light Detection and Ranging
MPC	–	Model Predictive Control
PID	–	Proporcionálně-integračně-derivační
RANSAC	–	Random Sample Consensus
ROS	–	Robot Operating System
RTK	–	Real-Time Kinematic
SAE	–	Society of Automotive Engineers
SLAM	–	Simultaneous Localization and Mapping

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Autonomní řízení je v posledních letech jedním z nejintenzivněji rozvíjených směrů v oblasti aplikované robotiky. Cílem této práce je navrhnout a implementovat funkční autonomní systém pro monopost Formula Student v simulátoru CARLA a ověřit jeho chování v úlohách odpovídajících disciplínám Driverless Cupu. Pipeline musí pokrýt celý řetězec od zpracování senzorických dat, přes detekci a klasifikaci kuželů, lokalizaci a mapování trati, až po plánování trajektorie a řízení vozidla. Vzhledem k tomu, že vývoj autonomního systému přímo na reálném monopostu je pomalý a rizikový, je vhodné nejprve celý systém ověřit v simulačním prostředí.

Práce je rozdělena do několika samostatných kapitol. První část seznamuje čtenáře s pravidly soutěže Formula Student a Driverless Cupu, porovnává dostupné simulátory autonomní jízdy a popisuje senzory používané v autonomních vozidlech. Následující kapitola se věnuje teoretickým základům metod použitých v pipeline – konvolučním neuronovým sítím, algoritmům RANSAC a DBSCAN pro zpracování LiDARových dat, projekční matici pro fúzi senzorů, algoritmům SLAM a regulátorům pro sledování trajektorie. Druhá část práce popisuje konkrétní implementaci celé pipeline v simulátoru CARLA, výběr a konfiguraci senzorů a experimentální vyhodnocení dvou režimů řízení na třech testovacích tratích. Závěr práce shrnuje dosažené výsledky a nastiňuje směry dalšího vývoje.

Kapitola 2

Analýza problému

Autonomní řízení monopostu v prostředí Formula Student klade specifické nároky na volbu simulátoru i senzorů. Než se pustíme do popisu konkrétních metod, je potřeba pochopit pravidla soutěže, dostupné nástroje pro simulaci a vlastnosti senzorů, se kterými systém pracuje.

Obrázek 2.1: Monopost týmu Formula TU Ostrava na FS Austria 2025

2.1 Formula Student

Soutěž Formula Student je mezinárodním kláním studentů technických vysokých škol a univerzit. Jejich úkolem je navrhnout a postavit vlastní monopost podle předem určených pravidel. Každoročně vycházejí dvě sady pravidel - FS Rules pro evropské soutěže a FSAE Rules pro zbytek světa. Každý závod se skládá ze tří statických disciplín a čtyř dynamických [1]. Statické disciplíny se skládají z Business Plan Presentation, ve které se tým snaží prodat monopost nebo jeho část fiktivním investorům, Cost and Manufacturing, kde se hodnotí kvalita BOM celého monopostu a výpočet

finanční a CO_2e náročnosti vybrané součásti vozidla, a Engineering Design, kde tým představuje a obhajuje technická řešení monopostu před panelem odborníků z průmyslu. Dynamické disciplíny testují výkon vozu na trati. Skid Pad měří maximální boční přetížení při průjezdu kružnic konstantního poloměru ve tvaru osmičky. Acceleration hodnotí podélné zrychlení na přímém úseku o délce 75 m. Autocross je pak jízdou na čas po úzké trati o délce přibližně 1 km, která kombinuje krátké rovinky, zatáčky a slalomy. Královskou disciplínou je pak Endurance, který se jede na stejné trati jako Autocross. Úkolem je zde ujet přibližně 22 km, přičemž v polovině dochází k výměně pilota. Současně se také měří spotřeba paliva či elektrické energie, která je hodnocena v rámci Efficiency. Bodování jednotlivých disciplín se nachází v tabulce 2.1.

Disciplína	**CV & EV**	**DV Cup**
Business Plan Presentation	75	-
Cost and Manufacturing	100	-
Engineering Design	150	150
Skid Pad	75	75
Acceleration	75	75
Autocross	100	100
Endurance	325	-
Trackdrive	-	200
Efficiency	100	-
Celkem	**1000**	**600**

Tabulka 2.1: Bodování disciplín - Formula Student Czech Republic 2026 [2]

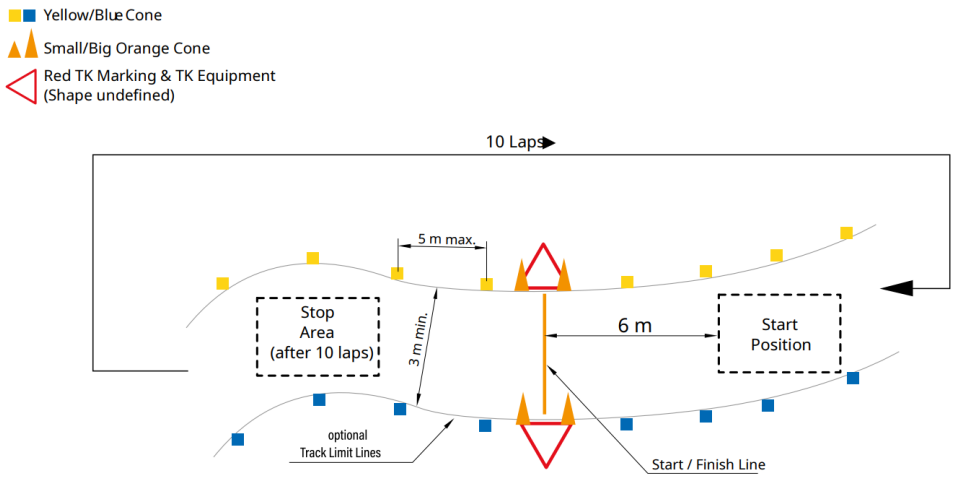
Kořeny soutěže sahají do roku 1981, kdy na University of Texas at Austin proběhl první ročník Formula SAE pod záštitou Society of Automotive Engineers [3]. Původně se jednalo o malou regionální akci se čtyřmi vozy, která však rychle rostla a v roce 1998 se rozšířila do Evropy první soutěží ve Velké Británii, organizovanou Institution of Mechanical Engineers [4]. V roce 2006 uspořádala organizace Formula Student Germany první ročník na evropské pevnině, a to na okruhu Hockenheimring, zpočátku pouze pro vozy se spalovacím motorem (CV). Elektrická třída (EV) byla zavedena v roce 2010 a v roce 2017 přibyla kategorie autonomních vozů (DV) - Formula Student Driverless [1]. V roce 2026 se soutěže účastní více než 600 týmů z celého světa, které poměřují své síly na 29 závodech [5].

2.1.1 Formula Student Driverless Cup

Driverless Cup (DV Cup) je samostatnou kategorií na vybraných FS závodech, kde týmy soutěží s autonomními monoposty bez zásahu řidiče. Kategorie byla poprvé představena v roce 2017 na Formula Student Germany [1]. Autonomní vůz musí splňovat veškeré technické požadavky hlavní

třídy EV nebo CV a navíc být vybaven senzorikou a výpočetní jednotkou schopnou řídit vůz po trati ohraničené barevnými kužely bez jakéhokoli lidského vstupu [6].

DV Cup se skládá z jedné statické a čtyř dynamických disciplín, jejichž bodování je v tabulce 2.1. Statickou disciplínou je Engineering Design, kde tým obhajuje technická řešení autonomního systému - především volbu senzorů, architekturu softwaru a strategii plánování trajektorie [2]. Dynamické disciplíny zahrnují DV Skid Pad a DV Acceleration, které testují základní schopnost vozu autonomně projet osmičku, respektive zrychlit na přímém úseku. DV Autocross ověřuje schopnost vozu projet jedním kolem neznámou trať vyznačenou kužely. Hlavní disciplínou DV Cupu je Trackdrive, což je obdoba Endurance pro autonomní vozy - vůz musí bez řidiče absolvovat 10 kol na trati podobné Autocrossu a prokázat tak spolehlivost a konzistenci autonomního systému [6]. Ukázka trati pro DV Autocross a Trackdrive je na obrázku 2.2.



Obrázek 2.2: Ukázka trati pro DV Autocross a Trackdrive [6]

Trať je pro všechny DV disciplíny vyznačena výhradně barevnými kužely [6], které lze vidět na obrázku 2.3. Levou hranici trati vymezují modré kužely, pravou hranici pak kužely žluté, přičemž jsou rozmístěny v intervalech přibližně 5 m. Oranžové kužely označují speciální oblasti - startovní a cílovou zónu. Používají se dva typy kuželů: malé kužely o výšce 325 mm a velké kužely o výšce 505 mm. Šířka trati je minimálně 3 m. Autonomní systém vozu musí být schopen kužely spolehlivě detekovat, klasifikovat podle barvy a na základě jejich polohy v reálném čase plánovat trajektorii.

2.1.2 Formula TU Ostrava

Formula TU Ostrava [7] je poměrně malý studentský tým působící na Vysoké škole báňské - Technické univerzitě Ostrava, který soutěží v kategorii vozidel se spalovacími motory. Je pod záštitou Institutu dopravy Fakulty strojní a je tvořen studenty téměř všech fakult VŠB-TUO. Tým byl založen v roce 2013 a do sezóny 2025 navrhl a postavil celkem 10 monopostů. Monopost ze sezóny

big orange cone	small orange cone	small yellow cone	small blue cone
two white stripes	single white stripe	single black stripe	single white stripe
WEMAS 307.610500.00.00	WEMAS 400.000013.00.00	WEMAS 400.000013.01.10	WEMAS 400.000043.00.00
285 × 285 × 505 mm		228 × 228 × 325 mm	
1.05 kg		0.45 kg	

Obrázek 2.3: Kužely používané pro DV Cup [6]

2025 se nachází na obrázku 2.1. Tým za svou historii dosáhl mnoha pódiových umístění, mezi která patří vítězství v disciplíně Cost and Manufacturing na FS Austria 2025 a celkově třetí místo na FS Czech 2025.

2.2 Simulátory pro autonomní řízení

Vývoj autonomního systému přímo na reálném voze je pomalý a riskantní, protože každý test na trati s kužely nese riziko havárie a přístup k trati je omezený. Simulátory umožňují testovat algoritmy percepce, plánování a řízení bez těchto omezení, s plnou opakovatelností a možností systematicky měnit podmínky.

2.2.1 CARLA

CARLA (Car Learning to Act) je open-source simulátor zaměřený na výzkum autonomní jízdy, představený v roce 2017 [8]. Původní verze byla postavena na Unreal Engine 4, aktuálně je k dispozici i verze využívající Unreal Engine 5 s vylepšeným vykreslováním světla a fyziky. Simulátor poskytuje konfigurovatelnou sadu senzorů: RGB kamery, hloubkové kamery, sémantickou segmentaci, LiDAR, radar, GNSS a IMU. Na generovaná senzorická data lze navíc aplikovat umělý šum, což je podstatné pro testování robustnosti filtračních algoritmů.

CARLA pracuje v architektuře server-klient, kde server provádí fyzikální simulaci a vykreslování scény, zatímco klient komunikuje skrze Python API nebo rozhraní pro ROS. Přes Python API lze vyčítat senzorická data, zasílat řídicí povely a ovládat parametry prostředí, včetně počasí, denní

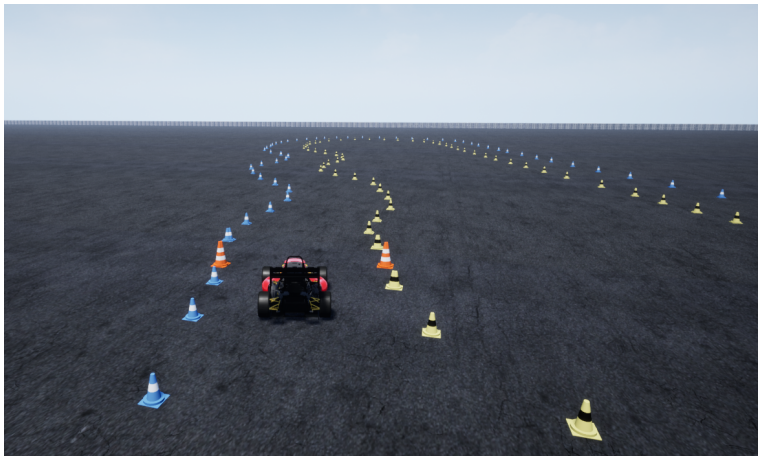
doby či koeficientu tření vozovky. Fyzikální model vozidla je konfigurovatelný, což umožňuje do jisté míry aproximovat chování závodního monopostu. Mapy lze upravovat v Unreal Editoru, kam je možné importovat vlastní 3D modely - například dopravní kužely - a vytvořit tak testovací tratě odpovídající prostředí DV Cupu. Simulátor je primárně navržen pro městský provoz, takže takové úpravy vyžadují určitou míru manuální práce.

2.2.2 Formula Student Driverless Simulator

Formula Student Driverless Simulator (FSDS) [9] vznikl pro potřeby virtuálního závodu FSOnline 2020, který se konal v důsledku zrušení reálných závodů během pandemie Covid-19. Na jeho vývoji se podíleli zástupci týmů FS Team Delft a MIT Driverless společně s organizátory závodu FS East. Simulátor je postaven nad Unreal Engine 4 a k simulaci senzorů a dynamiky vozidla využívá open-source plugin AirSim od společnosti Microsoft [10].

FSDS je na rozdíl od obecných simulátorů přímo přizpůsoben prostředí FS závodů. Obsahuje 3D modely kuželů odpovídající soutěžní specifikaci (malé a velké, modré, žluté i oranžové), předdefinovaný model monopostu s konfigurovatelnou sadou senzorů a několik komunitou vytvořených map - mimo jiné kopie tratí z reálných soutěží, jak je vidět na obrázku 2.4. Autonomní systém se připojuje prostřednictvím ROS bridge nebo Python API. Simulátor publikuje data ze senzorů (kamery, LiDAR, IMU, GNSS, volitelně i ground-truth polohy kuželů pro účely ladění) a přijímá řídicí příkazy ve formě požadovaného natočení volantu, úrovně plynu a brzdy. Součástí FSDS je i komponenta Operator - webové rozhraní pro řízení průběhu simulovaného závodu včetně rozlišení režimů Autocross a Trackdrive.

FSDS je fakticky ukončený projekt. Společnost Microsoft v roce 2022 zastavila vývoj platformy AirSim, na které je simulátor postaven, a přešla ke komerčnímu produktu Project AirSim. FSDS tak nelze snadno aktualizovat na novější verze Unreal Engine a postupně ztrácí kompatibilitu s novějšími distribucemi frameworku ROS 2 [10].



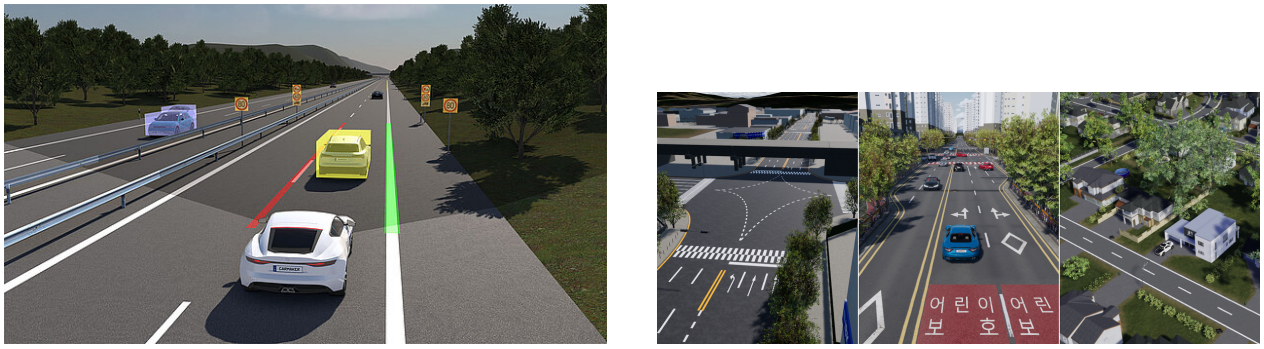
Obrázek 2.4: Ukázka Formula Student Driverless Simulator [9]

2.2.3 IPG Automotive CarMaker

CarMaker od společnosti IPG Automotive [11] je komerční nástroj používaný v automobilovém průmyslu pro virtuální testování dynamiky vozidel a asistenčních systémů. Jeho hlavní předností je detailní fyzikální model vozidla (multibody simulace) zahrnující kinematiku podvozku, pneumatiky a aerodynamiku. Pro autonomní systémy nabízí simulaci kamer, LiDARu a radaru. Simulátor je tradičně silně provázán s prostředím MATLAB/Simulink, podporuje však i rozhraní v C/C++ a Python API. Oproti open-source alternativám je nastavení vlastních scénářů pro testování počítačového vidění v tomto prostředí méně flexibilní. Ukázka prostředí CarMaker se nachází na obrázku 2.5 vlevo.

2.2.4 dSPACE AURELION

AURELION od společnosti dSPACE [12] je komerční řešení pro senzorově realistickou simulaci zaměřené na vývoj a validaci systémů autonomního řízení a asistenčních funkcí (ADAS). Simulátor je postaven na Unreal Engine a generuje fotorealistické obrazy v reálném čase pro simulaci kamer, LiDARu a radaru, včetně fyzikálně věrného modelování odrazů pomocí ray-tracingu. AURELION je součástí širšího ekosystému dSPACE, zahrnujícího modely dynamiky vozidla ASM, platformu VEOS pro SIL testování a cloudový validační framework SIMPHERA. Ukázka prostředí AURELION se nachází na obrázku 2.5 vpravo.



Obrázek 2.5: Ukázka IPG CarMaker [11] (vlevo) a dSPACE AURELION [12] (vpravo)

2.2.5 Výběr simulátoru

Volba simulátoru závisela na několika faktorech: musí zvládnout prostředí FS s kužely, běžet na dostupném hardwaru, umožňovat snadné úpravy scény a nabízet Python API. Porovnání kandidátů je v tabulce 2.2.

FSDS nabízí hotové prostředí pro DV Cup, ale jedná se o ukončený projekt se zastarávající kódovou základnou. dSPACE AURELION poskytuje vysokou kvalitu senzorové simulace certifikovanou pro průmyslové nasazení, avšak jedná se o komerční produkt zaměřený na validaci ADAS v silnič-

Simulátor	Licence	Stav vývoje	Hlavní rozhraní	HW náročnost
CARLA	Open-source	Aktivní	Python, ROS	Vyšší
FSDS	Open-source	Ukončen	Python, ROS	Střední
IPG CarMaker	Komerční	Aktivní	MATLAB, Python	Střední
dSPACE AURELION	Komerční	Aktivní	MATLAB, Python	Vyšší

Tabulka 2.2: Porovnání vybraných simulačních nástrojů pro autonomní řízení

ním provozu, nikoliv na závodní prostředí FS. IPG CarMaker vyniká přesností dynamického modelu vozidla, ale jeho uzavřenější architektura ztěžuje rychlé prototypování percepčních algoritmů.

Pro praktickou část byl zvolen simulátor CARLA. Má aktivní vývoj, dostatečnou vizuální věrnost a širokou komunitu. Klíčovým faktorem bylo Python API, přes které lze číst senzorická data i řídit vůz bez závislosti na ROS. Prostředí bylo přizpůsobeno potřebám DV Cupu importem vlastních modelů kuželů a tratí do Unreal Editoru.

2.3 Senzory autonomních vozidel

Autonomní formule musí současně detekovat a klasifikovat kužely podle barvy a lokalizovat se na trati [13]. K tomu slouží kombinace několika senzorů, protože žádný z nich sám o sobě neposkytuje vše potřebné [14]. Volba senzorů je navíc omezena prostorem na monopostu a výpočetním výkonem palubní jednotky.

2.3.1 Kamery

Kamera je elektronické zařízení snímající okolní scénu do dvourozměrného obrazu [14]. Světelné paprsky procházejí objektivem a dopadají na snímací čip, nejčastěji typu CMOS, jehož fotosenzory generují elektrické signály odpovídající intenzitě dopadajícího světla. Barevná informace se získává pomocí Bayerova filtru - mozaiky mikroskopických barevných filtrů umístěných nad jednotlivými pixely čipu. V kontextu autonomní formule patří kamera mezi nejdůležitější senzory, protože jako jediná poskytuje dostatečně kvalitní barevnou informaci potřebnou pro rozlišení modrých a žlutých kuželů vymezujících hranice trati. Mezi její výhody patří nízká cena, malé rozměry a relativně velký dosah. Nevýhodou je citlivost na světelné podmínky (protisvětlo, přechody mezi sluncem a stínem, odlesky na mokré vozovce) a absence přímé informace o hloubce.

Při výběru kamery pro monopost jsou klíčové rozlišení snímače (v praxi FS týmů obvykle 1 až 2 megapixely), zorné pole a snímková frekvence (alespoň 30 snímků za sekundu). Některé týmy používají dvě kamery - jednu s širším zorným polem pro blízké okolí a druhou s užším polem pro vzdálenější úsek trati.

2.3.2 LiDAR

LiDAR (Light Detection and Ranging) je aktivní senzor, který vysílá laserové paprsky, a na základě doby jejich odrazu od okolních objektů měří vzdálenost [15]. Základní princip spočívá v měření doby letu (Time of Flight) laserového pulzu – senzor vyšle krátký infračervený laserový pulz, ten se odrazí od objektu a vrátí se zpět do detektoru. Ze známé rychlosti světla c a změřeného časového intervalu t se vzdálenost d vypočítá podle rovnice 2.1. Výsledkem měření je trojrozměrné mračno bodů (point cloud), kde každý bod nese informaci o své poloze v prostoru a o intenzitě odrazu.

$$d = \frac{c \cdot t}{2} \tag{2.1}$$

Z hlediska konstrukce se v praxi vyskytují dva hlavní typy LiDARů relevantní pro autonomní formule [16]. Mechanicky rotující LiDARy obsahují soustavu laserových vysílačů a detektorů na rotujícím mechanismu, který zajišťuje 360stupňové pokrytí okolí. Senzor typicky obsahuje několik desítek laserových kanálů uspořádaných vertikálně, přičemž každý kanál snímá pod jiným vertikálním úhlem. Kompletní otočka mechanismu trvá řádově desítky milisekund, čímž vznikne jedno celé mračno bodů. Příkladem jsou senzory řady Velodyne nebo Ouster. Nevýhodou je mechanická složitost rotujících částí, vyšší hmotnost a citlivost na vibrace. Druhým typem jsou solid-state LiDARy, které nemají žádné pohyblivé části – paprsek je rozptylován elektronicky, například pomocí MEMS zrcátek. Díky tomu jsou kompaktnější, lehčí a odolnější vůči vibracím. Mají ovšem omezené zorné pole a pro pokrytí celého okolí monopostu je nutné použít více kusů.

V porovnání s kamerami je LiDAR výrazně přesnější v měření vzdáleností a jeho výkon téměř vůbec nezávisí na okolním osvětlení [14], což je podstatná výhoda při proměnlivých světelných podmínkách na trati. Tvar dopravního kužele se v mračnu bodů projevuje jako malý shluk bodů s charakteristickým profilem. Na blízkou vzdálenost (do přibližně 10 metrů) je tento tvar poměrně dobře rozpoznatelný. S rostoucí vzdáleností ovšem počet bodů na kuželu rychle klesá a detekce se stává méně spolehlivou.

Při výběru LiDARu pro monopost jsou rozhodující zejména počet kanálů (vertikální rozlišení), dosah, frekvence skenování a úhlové rozlišení. LiDARy s 16 kanály poskytují řidší mračno, ve kterém je kužel na vzdálenost 15 a více metrů reprezentován jen jednotkami bodů, zatímco senzory s 64 nebo 128 kanály nabízejí výrazně hustší pokrytí za odpovídající cenu. Dosah senzoru pro potřeby FS nemusí být extrémně velký - kužely je potřeba detekovat na vzdálenost maximálně 20 až 30 metrů. Frekvence skenování se pohybuje typicky mezi 10 a 20 Hz.

Hlavní slabinou LiDARu v kontextu FS je neschopnost rozeznat barvy objektů. Senzor sice poskytuje informaci o intenzitě odrazu, avšak rozdíl mezi modrými a žlutými kužely je v praxi příliš malý a příliš závislý na podmínkách na to, aby sloužil jako spolehlivý klasifikátor barvy. Proto se data z LiDARu téměř vždy fúzují s barevnou informací z kamery [13]. Mezi další nevýhody patří vyšší pořizovací cena a větší rozměry oproti kamerám.

2.3.3 IMU a GNSS

Pro určení polohy, orientace a dynamiky samotného vozidla se využívá inerciální měřicí jednotka (IMU) a globální navigační satelitní systém (GNSS). IMU typicky obsahuje tříosý akcelerometr a tříosý gyroskop s velmi vysokou frekvencí výstupu (řádově stovky Hz). Integrací zrychlení a úhlových rychlostí lze průběžně počítat polohu a orientaci monopostu, tento výpočet ovšem postupně akumuluje chybu (takzvaný drift), a proto se nemůže používat samostatně pro dlouhodobou lokalizaci.

GNSS, u autonomních formulí takřka výhradně vylepšený o technologii RTK (Real-Time Kinematic) pro dosažení centimetrové přesnosti, poskytuje absolutní pozici na trati. Samotný GNSS však trpí nižší obnovovací frekvencí (typicky 5 až 20 Hz) a možnými výpadky signálu. Proto se údaje z IMU a GNSS vzájemně doplňují - GNSS koriguje narůstající drift IMU a IMU poskytuje spojitý odhad polohy mezi jednotlivými GNSS měřeními. K fúzi dat z obou senzorů se přidávají ještě údaje ze snímačů otáček kol a celý odhad stavu se provádí pomocí filtračních algoritmů, například rozšířeného Kalmanova filtru.

Kapitola 3

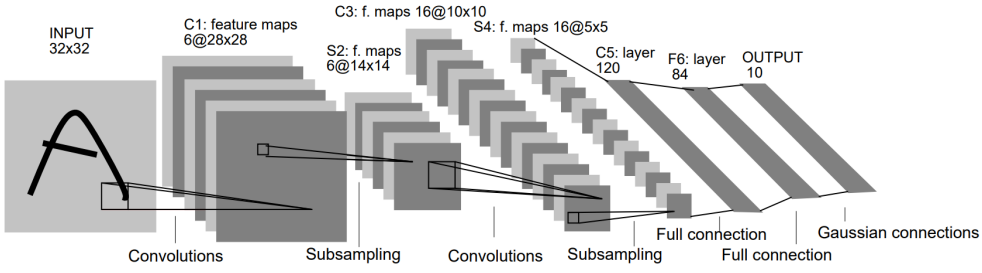
Metody

Autonomní pipeline se skládá z několika na sebe navazujících kroků: detekce kuželů, klasifikace jejich barvy, lokalizace vozidla, plánování trajektorie a samotné řízení. Každý krok využívá jinou metodu — od konvolučních sítí přes SLAM až po geometrické regulátory. Tato kapitola popisuje teoretický základ těchto metod; konkrétní implementace a parametry jsou pak v kapitole 4.

3.1 Zpracování obrazových dat pomocí CNN

Obrazová data z kamery slouží v pipeline této práce výhradně k určení barvy kužele — rozlišení modré (levá hranice), žluté (pravé hranice) a oranžové (startovní/cílová zóna). Samotnou detekci zajišťuje LiDAR. Pro klasifikaci se používá konvoluční neuronová síť (CNN) [17, 18].

CNN je třída hlubokých sítí navržená pro data s mřížkovou strukturou, typicky dvourozměrné obrazy. Oproti plně propojeným sítím využívá lokální propojení a sdílení vah, čímž výrazně snižuje počet parametrů a přirozeně zachycuje prostorové vztahy [18]. Typická architektura střídá konvoluční vrstvy extrahující příznaky, pooling vrstvy zmenšující prostorové rozměry a plně propojené vrstvy provádějící výslednou klasifikaci. Schéma klasické architektury LeNet-5 je na obrázku 3.1.



Obrázek 3.1: Architektura konvoluční neuronové sítě LeNet-5 [17]

Konvoluční vrstva obsahuje sadu naučitelných filtrů o malých rozměrech (typicky 3×3), které se posunují přes vstup a v každé pozici počítají skalární součin s odpovídajícím výřezem:

$$(I * K)(i, j) = \sum_m \sum_n I(i + m, j + n) \cdot K(m, n) \tag{3.1}$$

Na výstup se aplikuje nelineární aktivace, nejčastěji ReLU $f(x) = \max(0, x)$ [19], která je výpočetně efektivní a zmírňuje problém mizejícího gradientu. Pooling vrstvy (typicky max pooling 2×2) redukují rozměry map příznaků a zavádějí invarianci vůči malým posunům. Po sérii těchto bloků se příznaky převedou do jednorozměrného vektoru a předají plně propojeným vrstvám, na jejichž výstupu funkce softmax [18] produkuje pravděpodobnosti příslušnosti ke třídám. Pro klasifikaci kuželů je počet tříd $C = 4$ (modrý, žlutý, oranžový, nekužel).

Parametry sítě se optimalizují zpětnou propagací s křížovou entropií jako ztrátovou funkcí:

$$L = -\sum_{i=1}^{C} y_i \cdot \log(\hat{y}_i) \tag{3.2}$$

kde y_i je skutečná třída (one-hot) a $\hat{y}_i$ predikovaná pravděpodobnost. Jako optimalizátor se běžně používá Adam [20], který adaptivně upravuje krokovou velikost pro každý parametr. Konkrétní architektura a trénování klasifikátoru pro tuto práci jsou popsány v kapitole o implementaci.

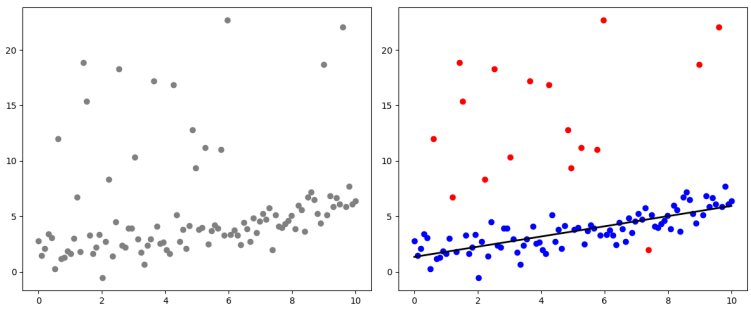
3.2 Zpracování dat z LiDARu

Data z LiDARu mají formu trojrozměrného mračna bodů, kde každý bod nese souřadnice v prostoru a hodnotu intenzity odrazu. Před samotnou detekcí kuželů je nutné toto mračno předzpracovat. Prvním krokem je typicky odstranění bodů patřících rovině vozovky, aby v datech zůstaly pouze body náležející objektům vyčnívajícím nad povrch. K tomu se používá algoritmus RANSAC. Druhým krokem je seskupení zbylých bodů do shluků, kde každý shluk ideálně odpovídá jednomu kuželu. Pro tento účel se používá algoritmus DBSCAN.

3.2.1 RANSAC

RANSAC (Random Sample Consensus) [21] je iterativní algoritmus pro odhad parametrů matematického modelu z dat obsahujících velké množství odlehlých hodnot (outlierů). Na rozdíl od klasických metod, jako je metoda nejmenších čtverců, které se snaží využít co nejvíce dat najednou a jsou tak citlivé na přítomnost outlierů, RANSAC postupuje opačně - v každé iteraci náhodně vybere minimální podmnožinu bodů potřebnou k určení modelu, z této podmnožiny model vypočítá a následně ověří, kolik zbývajících bodů s tímto modelem souhlasí (takzvaný consensus set). Body, jejichž vzdálenost od modelu nepřekročí stanovenou prahovou hodnotu, se považují za inliery.

Tento postup se opakuje po předem daný počet iterací a jako výsledek se vybere model s největším počtem inlierů. Výsledek je možné vidět na obrázku 3.2.



Obrázek 3.2: Vizualizace vstupu a výstupu algoritmu RANSAC na syntetických 2D datech

V kontextu zpracování LiDARových dat se RANSAC nejčastěji používá pro detekci a odstranění roviny vozovky z mračna bodů. Rovinu v prostoru lze popsat třemi parametry, proto algoritmus v každé iteraci náhodně vybere tři body, proloží jimi rovinu a spočítá, kolik bodů z celého mračna leží v dostatečné blízkosti této roviny. Po skončení všech iterací se rovina s největším počtem inlierů prohlásí za rovinu vozovky a její body se z mračna odstraní. Zbylé body pak reprezentují objekty vyčnívající nad vozovku — tedy kužely, svodidla, ostatní vozidla a podobně.

Počet iterací k potřebných k tomu, aby algoritmus s pravděpodobností p nalezl alespoň jednu podmnožinu tvořenou výhradně inliery, závisí na podílu inlierů w v datech a na velikosti podmnožiny n podle vztahu:

$$k = \frac{\log(1-p)}{\log(1-w^n)} \tag{3.3}$$

Pro detekci roviny je $n = 3$. Pokud například inliery tvoří 60% dat a požadovaná pravděpodobnost je 99%, potřebný počet iterací vychází přibližně na 20. V praxi se často volí vyšší počet iterací pro zajištění robustnosti, neboť podíl inlierů nemusí být předem přesně znám.

RANSAC je díky své robustnosti vůči outlierům vhodný právě pro LiDARová data, kde body vozovky mnohonásobně převyšují body kuželů. Algoritmus je však nedeterministický — výsledek závisí na náhodném výběru podmnožin — a vyžaduje vhodně zvolený práh: příliš malý práh vynechá body patřící rovině, příliš velký naopak zahrne i nízké objekty.

3.2.2 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) je shlukovací algoritmus založený na hustotě bodů v prostoru, navržený Esterem a kol. v roce 1996 [22]. Na rozdíl od algoritmů, jako je k-means, nevyžaduje předem znát počet shluků a dokáže najít shluky libovolného tvaru, což je při zpracování LiDARových dat podstatná vlastnost, protože kužely ani jiné objekty na trati nemají v mračnu bodů pravidelný tvar.

Algoritmus pracuje se dvěma parametry: poloměrem okolí ε (epsilon) a minimálním počtem bodů $MinPts$. Pro každý bod v mračnu algoritmus prozkoumá jeho ε-okolí, tedy množinu všech bodů vzdálených nejvýše ε. Pokud toto okolí obsahuje alespoň $MinPts$ bodů, je daný bod označen jako jádrový bod (core point). Body, které leží v ε-okolí jádrového bodu, ale samy nesplňují podmínku pro jádrový bod, se označují jako hraniční body (border points). Body, které neleží v ε-okolí žádného jádrového bodu, jsou klasifikovány jako šum (noise).

Shluky se tvoří tak, že algoritmus začne u libovolného dosud nezpracovaného jádrového bodu a rekurzivně přidává všechny body dosažitelné přes hustotní spojení - tedy body, ke kterým vede řetězec jádrových bodů, kde každý následující leží v ε-okolí předchozího. Tímto způsobem se shluk rozrůstá do všech směrů, kde je dostatečná hustota bodů, a přirozeně se zastaví na hranicích, kde hustota klesne pod stanovený práh.

V kontextu zpracování LiDARových dat po odstranění roviny vozovky pomocí RANSAC se DBSCAN aplikuje na zbývající body. Každý nalezený shluk potenciálně odpovídá jednomu objektu na trati - kuželu, bariéře nebo jinému předmětu. Shluky se následně filtrují podle velikosti a tvaru, aby se identifikovaly ty, které odpovídají dopravním kuželům. Body klasifikované jako šum se zahazují.

Volba parametrů ε a $MinPts$ závisí na hustotě mračna bodů, která je dána typem LiDARu a vzdáleností objektu. Pro bližší kužely, na které dopadá více bodů, je vhodné menší ε a vyšší $MinPts$, zatímco pro vzdálenější kužely s řidším pokrytím by byly potřeba opačné hodnoty. V praxi se proto volí kompromisní nastavení, případně se parametry adaptivně upravují podle vzdálenosti od senzoru.

DBSCAN je pro zpracování LiDARových dat vhodný ze dvou důvodů: nevyžaduje předem znát počet shluků (kuželů) a přirozeně odděluje šum. Citlivou stránkou je volba parametrů — nevhodné ε může dva blízké kužely sloučit do jednoho shluku nebo naopak rozdělit jeden kužel na více shluků. Kompromisní nastavení a jeho vliv na detekci jsou diskutovány v kapitole o implementaci.

3.3 Fúze senzorů

Kamera umí rozlišit barvy, ale neví, jak daleko kužel je. LiDAR přesně změří vzdálenost, ale nerozezná modrou od žluté. Pro spolehlivou detekci s klasifikací je proto nutné data z obou senzorů sloučit — fúzovat [14].

V kontextu Formula Student Driverless se nejčastěji používá takzvaná raná fúze (early fusion), při které se 3D body z LiDARu promítají do 2D obrazu kamery. Tím se každému bodu z mračna přiřadí odpovídající pixel v obraze a s ním i barevná informace. Kužel detekovaný v obraze kamerou (například pomocí CNN) tak získá přesnou prostorovou polohu z LiDARu, a naopak shluk bodů z LiDARu získá barevnou klasifikaci z kamery [13].

Alternativou je pozdní fúze (late fusion), při které oba senzory detekují objekty nezávisle a výsledné detekce se poté párují na základě prostorové blízkosti. Tento přístup je jednodušší na implementaci, ale vyžaduje, aby oba senzory byly schopny objekt spolehlivě detekovat samostatně, což u LiDARu s omezeným počtem kanálů na větší vzdálenosti nemusí být splněno.

3.3.1 Projekční matice

Klíčovým matematickým nástrojem pro ranou fúzi dat z LiDARu a kamery je projekční matice, která umožňuje převod 3D bodu z prostoru do 2D souřadnic v obraze kamery [23]. Tento převod se skládá ze dvou kroků - nejprve se bod transformuje ze souřadnicového systému LiDARu do souřadnicového systému kamery a poté se promítne na obrazovou rovinu.

Bod $P_L = [X_L, Y_L, Z_L]^T$ v souřadnicovém systému LiDARu se převede do souřadnicového systému kamery pomocí vnějších parametrů (extrinsic parameters) - rotační matice R o rozměru 3×3 a translačního vektoru t o rozměru 3×1, které popisují vzájemnou polohu a orientaci obou senzorů:

$$P_C = R \cdot P_L + t \tag{3.4}$$

Bod $P_C = [X_C, Y_C, Z_C]^T$ v souřadnicovém systému kamery se následně promítne do obrazových souřadnic $[u, v]^T$ pomocí vnitřní matice kamery K:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

kde f_x a f_y jsou ohniskové vzdálenosti vyjádřené v pixelech a c_x, c_y jsou souřadnice optického středu. Výsledná projekce bodu do obrazu se zapíše jako:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \cdot [R|t] \cdot \begin{bmatrix} X_L \\ Y_L \\ Z_L \\ 1 \end{bmatrix} \tag{3.6}$$

kde s je škálovací faktor odpovídající hloubce bodu Z_C a $[R|t]$ je matice o rozměru 3×4 vzniklá spojením rotace a translace. Součin $K \cdot [R|t]$ se označuje jako projekční matice P o rozměru 3×4, která celý převod z 3D do 2D popisuje jedinou maticí.

V praxi se tento postup používá tak, že pro každý centroid z mračna LiDARu se vypočítají jeho obrazové souřadnice $[u, v]$ a ověří se, zda se jedná o kužel. Pokud ano, přiřadí se tomuto kuželu prostorová poloha z LiDARu a barevná klasifikace z kamery. Body, které leží za kamerou ($Z_C \leq 0$) nebo mimo obraz, se z fúze vyřadí.

3.3.2 Kalmanův filtr

Kalmanův filtr [24] rekurzivně odhaduje stav dynamického systému z řady zašuměných měření. Algoritmus střídá dva kroky: predikci stavu na základě modelu systému a korekci na základě nového měření. Váhu obou zdrojů informací automaticky určuje Kalmanův zisk, odvozený z kovariancí predikce a měření.

Stav systému v čase k je reprezentován vektorem x_k a jeho kovarianční maticí P_k. Predikční krok využívá stavový model:

$$\hat{x}_{k|k-1} = F \cdot \hat{x}_{k-1|k-1} \tag{3.7}$$

$$P_{k|k-1} = F \cdot P_{k-1|k-1} \cdot F^T + Q \tag{3.8}$$

kde F je stavová přechodová matice, popisující dynamiku systému, a Q je kovarianční matice procesního šumu, která modeluje nepřesnost stavového modelu.

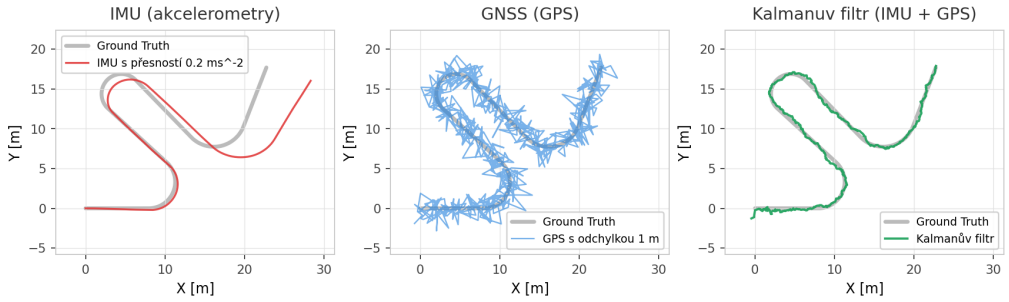
Korekční krok využívá měření z_k:

$$K_k = P_{k|k-1} \cdot H^T \cdot (H \cdot P_{k|k-1} \cdot H^T + R)^{-1} \tag{3.9}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \cdot (z_k - H \cdot \hat{x}_{k|k-1}) \tag{3.10}$$

$$P_{k|k} = (I - K_k \cdot H) \cdot P_{k|k-1} \tag{3.11}$$

kde H je měřicí matice popisující vztah mezi stavem a měřením, R je kovarianční matice měřicího šumu a K_k je Kalmanův zisk. Výraz $z_k - H \cdot \hat{x}_{k|k-1}$ se nazývá inovace a představuje rozdíl mezi skutečným měřením a predikcí měření.



Obrázek 3.3: Vizualizace dat z IMU a GNSS se šumem a odhad pozice pomocí Kalmanova filtru

Lineární Kalmanův filtr předpokládá lineární stavový model a gaussovský šum. Vztahy v autonomním vozidle jsou však často nelineární — například převod polárních souřadnic LiDARu do kartézských souřadnic mapy. Rozšířený Kalmanův filtr (EKF) [25] řeší tento problém linearizací nelineárních funkcí pomocí Jacobiánu v okolí aktuálního odhadu. V autonomních monopostech se EKF typicky používá pro fúzi dat z IMU, GNSS a odometrie kol.

3.4 SLAM

SLAM (Simultaneous Localization and Mapping) je problém, při kterém se mobilní robot (nebo v případě FS autonomní monopost) snaží současně vytvářet mapu svého okolí a zároveň se v této mapě lokalizovat [26]. V kontextu Formula Student Driverless je SLAM obzvlášť relevantní, protože vůz při prvním průjezdu tratí (DV Autocross) nezná rozmístění kuželů a musí si mapu trati postupně budovat za jízdy. Při opakovaných průjezdech (Trackdrive) pak může využít vytvořenou mapu k přesnějšímu plánování trajektorie a vyšší rychlosti [13].

Formálně lze SLAM popsat jako odhad sdružené pravděpodobnostní distribuce polohy vozidla x_t a mapy m (v prostředí FS tvořeném polohami kuželů) podmíněné všemi dosavadními měřeními $z_{1:t}$ a řídicími vstupy $u_{1:t}$:

$$p(x_t, m \mid z_{1:t}, u_{1:t}) \tag{3.12}$$

Hlavní obtížnost SLAM spočívá v tom, že lokalizace a mapování jsou vzájemně závislé - pro přesnou lokalizaci je potřeba znát mapu, ale pro vytvoření mapy je potřeba znát polohu vozidla. Různé přístupy k řešení tohoto problému se liší způsobem, jakým reprezentují a aktualizují sdruženou distribuci. Následující podsekce popisují tři nejrozšířenější varianty: EKF SLAM, FastSLAM a Graph SLAM.

3.4.1 EKF SLAM

EKF SLAM je historicky nejstarší a nejznámější přístup k řešení problému SLAM, založený na rozšířeném Kalmanově filtru [26]. Celý stav systému - poloha a orientace vozidla spolu s polohami všech detekovaných kuželů - je reprezentován jedním stavovým vektorem x a příslušnou kovarianční maticí P, která zachycuje nejistotu odhadu a korelace mezi polohou vozidla a jednotlivými kužely.

Stavový vektor má tvar:

$$x = \begin{bmatrix} x_v & y_v & \theta_v & x_1 & y_1 & \cdots & x_N & y_N \end{bmatrix}^T \tag{3.13}$$

kde $[x_v, y_v, \theta_v]^T$ je poloha a orientace vozidla a $[x_i, y_i]^T$ jsou souřadnice i-tého kužele v mapě. Kovarianční matice P má rozměr $(3 + 2N) \times (3 + 2N)$, kde N je počet kuželů v mapě.

Algoritmus v každém časovém kroku provádí dva kroky známé z klasického Kalmanova filtru: predikci (aktualizace odhadu polohy vozidla na základě řídicích vstupů) a korekci (zpřesnění celého stavu na základě nového měření kužele). Klíčovou vlastností EKF SLAM je, že korekce jednoho měření aktualizuje korelace mezi všemi kužely v mapě, což v principu vede ke konzistentnímu odhadu.

Hlavní nevýhodou EKF SLAM je kvadratická výpočetní složitost $O(N^2)$ vzhledem k počtu kuželů, způsobená nutností aktualizovat celou kovarianční matici při každém měření. Pro typickou FS trať s řádově 100 až 200 kužely je to ještě zvládnutelné, ale algoritmus se špatně škáluje na větší

prostředí. Další nevýhodou je linearizace nelineárních funkcí pomocí Jacobiánů, která může vést k nekonzistenci odhadu, pokud jsou nelinearity výrazné.

3.4.2 FastSLAM

FastSLAM je algoritmus navržený Montemerlem a kol., který překonává výpočetní omezení EKF SLAM rozložením problému na menší nezávislé části pomocí Rao-Blackwellizovaného částicového filtru [27]. Klíčové pozorování, na kterém FastSLAM staví, je, že pokud je známa trajektorie vozidla, jsou polohy jednotlivých kuželů na sobě vzájemně nezávislé. Tato podmíněná nezávislost umožňuje faktorizaci sdružené distribuce:

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) = p(x_{1:t} \mid z_{1:t}, u_{1:t}) \prod_{i=1}^{N} p(m_i \mid x_{1:t}, z_{1:t}) \tag{3.14}$$

Trajektorie vozidla $x_{1:t}$ se odhaduje pomocí částicového filtru, kde každá částice (particle) reprezentuje jednu hypotézu o trajektorii. Ke každé částici je připojena sada N nezávislých Kalmanových filtrů o rozměru 2×2, z nichž každý sleduje polohu jednoho kužele podmíněnou na trajektorii dané částice.

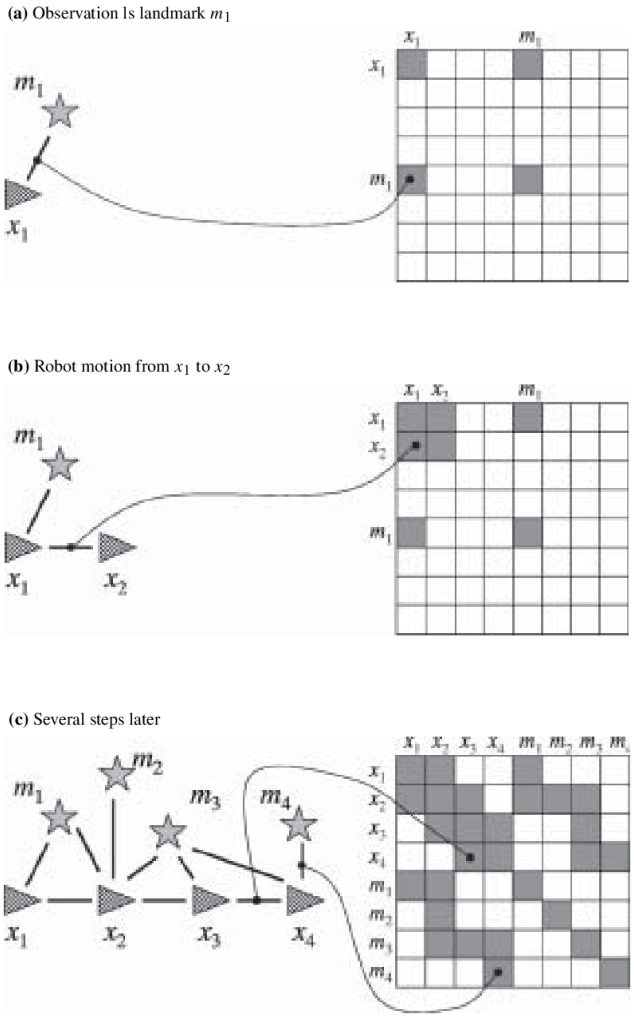
Díky této faktorizaci má FastSLAM výpočetní složitost $O(M \log N)$, kde M je počet částic a N počet kuželů, což je výrazné zlepšení oproti $O(N^2)$ u EKF SLAM. Vylepšená verze FastSLAM 2.0 [28] navíc zahrnuje aktuální měření do návrhu distribuce pro vzorkování částic, čímž snižuje problém degenerace částic a zvyšuje přesnost odhadu.

Nevýhodou FastSLAM je, že kvalita odhadu závisí na počtu částic – příliš málo částic vede k chybné reprezentaci distribuce, příliš mnoho částic zvyšuje výpočetní náročnost. V prostředí FS, kde se trať při opakovaných průjezdech nemění, se FastSLAM osvědčuje díky schopnosti udržovat více hypotéz o trajektorii současně, což pomáhá řešit nejednoznačnosti v přiřazení měření ke kuželům (data association).

3.4.3 Graph SLAM

Graph SLAM formuluje problém SLAM jako optimalizační úlohu nad grafem [29]. Na rozdíl od EKF SLAM a FastSLAM, které odhadují stav inkrementálně v každém časovém kroku (filtrační přístup), Graph SLAM shromáždí všechna měření a omezení a poté hledá konfiguraci, která je s nimi globálně nejkonzistentnější (vyhlazovací přístup).

Graf se skládá z uzlů a hran. Uzly odpovídají odhadovaným veličinám – polohám vozidla v jednotlivých časových krocích a polohám kuželů v mapě. Hrany reprezentují omezení (constraints) mezi uzly: hrany z odometrie spojují po sobě jdoucí polohy vozidla, hrany z měření spojují polohu vozidla s pozorovaným kuželem. Každá hrana nese informaci o relativní poloze mezi spojenými uzly a o nejistotě tohoto měření. Příklad takového grafu je na obrázku 3.4.



Obrázek 3.4: Mapování pomocí Graph SLAM [29]

Hledání optimální konfigurace uzlů se formuluje jako minimalizace součtu kvadratických chyb přes všechny hrany:

$$x^* = \arg\min_x \sum_{(i,j)\in\mathcal{E}} e_{ij}(x_i, x_j)^T \Omega_{ij} e_{ij}(x_i, x_j) \tag{3.15}$$

kde e_{ij} je chybová funkce vyjadřující rozdíl mezi predikovaným a měřeným vztahem uzlů i a j, Ω_{ij} je informační matice (inverzní kovariance) daného omezení a $\mathcal{E}$ je množina všech hran v grafu. Tato nelineární optimalizační úloha se řeší iterativně, nejčastěji metodami Gauss-Newton nebo Levenberg-Marquardt.

Hlavní výhodou Graph SLAM je, že optimalizuje celou trajektorii a mapu najednou, čímž dosahuje vyšší konzistence odhadu než filtrační metody. Dokáže přirozeně zpracovat uzavření smyčky (loop closure) - situaci, kdy vozidlo projede místem, kde už bylo, a systém na základě opakovaného

pozorování stejných kuželů opraví akumulovaný drift v celé trajektorii. To je v prostředí FS zvlášť důležité, protože trať je uzavřená a vozidlo ji při Trackdrive objíždí opakovaně.

Nevýhodou je vyšší výpočetní náročnost optimalizačního kroku, zejména u velkých grafů. V praxi se však používají efektivní řídké (sparse) řešiče, které využívají faktu, že většina uzlů je spojena pouze s malým počtem sousedů. Pro rozsah typické FS trati (řádově stovky uzlů) je Graph SLAM výpočetně dobře zvládnutelný a představuje v současnosti jeden z nejpoužívanějších přístupů k mapování trati v soutěži Formula Student Driverless [30].

3.5 Algoritmy pro řízení vozidla

Úkolem řídicího algoritmu je na základě naplánované trajektorie generovat konkrétní příkazy pro ovládací prvky vozidla - natočení volantu, úroveň plynu a brzdy - tak, aby monopost co nejpřesněji sledoval požadovanou dráhu. Řídicí algoritmy se obvykle dělí na podélné řízení (kontrola rychlosti) a příčné řízení (kontrola směru jízdy). V prostředí Formula Student Driverless je příčné řízení obzvlášť kritické, protože tratě obsahují ostré zatáčky a úzké průjezdy mezi kužely, kde i malá odchylka od trajektorie může vést ke sražení kužele a penalizaci.

Klíčovými veličinami pro popis odchylky vozidla od požadované trajektorie jsou příčná odchylka (crosstrack error) - vzdálenost referenčního bodu vozidla od nejbližšího bodu na trajektorii - a odchylka směru (heading error) - rozdíl mezi aktuálním směrem jízdy a směrem trajektorie v daném bodě. Cílem každého řídicího algoritmu je obě tyto odchylky minimalizovat.

3.5.1 PID regulátor

PID (Proporcionálně-Integračně-Derivační) regulátor [31] je nejrozšířenějším typem zpětnovazebního regulátoru v průmyslových aplikacích. Akční zásah se skládá ze tří složek: proporcionální složka reaguje na aktuální velikost regulační odchylky, integrační složka kumuluje odchylku v čase a eliminuje tak ustálenou odchylku, a derivační složka reaguje na rychlost změny odchylky a tlumí kmitání. Výstup regulátoru je dán vztahem:

$$u(t) = K_p \cdot e(t) + K_i \int_0^t e(\tau)d\tau + K_d \cdot \frac{de(t)}{dt} \tag{3.16}$$

kde $e(t)$ je regulační odchylka (v případě příčného řízení typicky příčná odchylka), K_p, K_i a K_d jsou zesílení jednotlivých složek.

Pro řízení autonomní formule se PID regulátor nejčastěji používá pro podélné řízení - tedy regulaci rychlosti vozidla na požadovanou hodnotu ovládáním plynu a brzdy. Pro příčné řízení (natočení volantu) je PID regulátor použitelný na jednoduchých tratích, avšak na složitějších trasách s proměnlivým poloměrem zatáček má tendenci k opožděné reakci a kmitání, protože neuvažuje geometrii trajektorie dopředu. Proto se pro příčné řízení v praxi dává přednost geometrickým nebo prediktivním metodám.

3.5.2 Pure Pursuit

Pure Pursuit je geometrický algoritmus pro příčné řízení, který sleduje bod na referenční trajektorii umístěný v předem definované vzdálenosti před vozidlem - takzvaný lookahead bod [32]. Algoritmus vypočítá natočení volantu tak, aby vozidlo projelo kruhový oblouk z aktuální polohy zadní nápravy do tohoto bodu.

Natočení volantu δ se určí ze vztahu:

$$\delta = \arctan\left(\frac{2 \cdot L \cdot \sin(\alpha)}{l_d}\right) \tag{3.17}$$

kde L je rozvor náprav vozidla, α je úhel mezi podélnou osou vozidla a přímkou vedoucí z centra zadní nápravy k lookahead bodu a l_d je lookahead vzdálenost.

Volba lookahead vzdálenosti zásadně ovlivňuje chování regulátoru. Kratší lookahead vzdálenost vede k přesnějšímu sledování trajektorie, ale může způsobovat kmitání, zejména při vyšších rychlostech. Delší lookahead vzdálenost naopak průjezd vyhlazuje, ale za cenu většího řezání zatáček. V praxi se proto lookahead vzdálenost často nastavuje jako lineární funkce rychlosti vozidla - čím rychleji monopost jede, tím dále dopředu se dívá.

Tento algoritmus je jednoduchý na implementaci a výpočetně nenáročný. Neuvažuje však směrovou odchylku (heading error) a při vjezdu do ostré zatáčky reaguje se zpožděním.

3.5.3 Stanley

Stanley je geometrický řídicí algoritmus pojmenovaný podle autonomního vozidla Stanley, které v roce 2005 zvítězilo v soutěži DARPA Grand Challenge [33]. Na rozdíl od Pure Pursuit, který pracuje s referenčním bodem na zadní nápravě, Stanley používá jako referenční bod střed přední nápravy a ve svém řídicím zákoně kombinuje korekci směrové odchylky i příčné odchylky.

Natočení volantu δ se vypočítá jako:

$$\delta(t) = \psi(t) + \arctan\left(\frac{k \cdot e(t)}{v(t)}\right) \tag{3.18}$$

kde $\psi(t)$ je odchylka směru vozidla od směru trajektorie (heading error), $e(t)$ je příčná odchylka přední nápravy od trajektorie (crosstrack error), $v(t)$ je aktuální rychlost vozidla a k je zesílení regulátoru. První člen koriguje směrovou odchylku, druhý člen koriguje příčnou odchylku, přičemž vliv příčné korekce klesá s rostoucí rychlostí, což přirozeně stabilizuje řízení ve vyšších rychlostech.

Oproti Pure Pursuit reaguje Stanley na odchylku od trajektorie bezprostředněji, protože pracuje s aktuální chybou na přední nápravě, nikoli s lookahead bodem. Díky tomu má typicky menší příčnou odchylku v zatáčkách. Nevýhodou je větší citlivost na šum v měření polohy a směru, což se může projevit nervózním řízením, zejména na nerovném povrchu.

3.5.4 MPC

Model Predictive Control (MPC) je prediktivní řídicí metoda, která v každém časovém kroku řeší optimalizační úlohu na konečném horizontu [34]. Na rozdíl od geometrických metod, které reagují na aktuální nebo blízkou odchylku od trajektorie, MPC predikuje budoucí chování vozidla na několik kroků dopředu pomocí matematického modelu a hledá optimální sekvenci řídicích vstupů, která minimalizuje zvolenou účelovou funkci.

V každém časovém kroku MPC řeší následující optimalizační problém:

$$\min_{u_0,\ldots,u_{N-1}} \sum_{k=0}^{N-1} \left[(x_k - x_k^{ref})^T Q (x_k - x_k^{ref}) + u_k^T R u_k \right] \tag{3.19}$$

kde x_k je predikovaný stav vozidla v kroku k (typicky poloha, směr, rychlost), x_k^{ref} je požadovaný stav na referenční trajektorii, u_k je řídicí vstup (natočení volantu, plyn/brzda), N je délka predikčního horizontu a matice Q a R jsou váhové matice penalizující odchylku od trajektorie, respektive velikost řídicích zásahů. Predikce stavů se provádí pomocí diskretizovaného modelu dynamiky vozidla, typicky kinematického bicyklového modelu.

Z optimální sekvence řídicích vstupů $u_0, \ldots, u_{N-1}$ se aplikuje pouze první prvek u_0 a celý výpočet se v dalším kroku opakuje s aktualizovaným stavem - tento princip se nazývá ustupující horizont (receding horizon).

Hlavní výhodou MPC je schopnost přirozeně zahrnout omezení - maximální natočení volantu, maximální zrychlení, hranice trati - přímo do optimalizačního problému. MPC tak dokáže generovat řídicí příkazy, které současně sledují trajektorii, dodržují fyzikální limity vozidla a reagují na nadcházející tvar trati. Nevýhodou je výrazně vyšší výpočetní náročnost oproti geometrickým metodám, protože v každém kroku je nutné řešit optimalizační úlohu v reálném čase. Pro monopost jedoucí rychlostí 60 km/h to znamená nutnost vypočítat řešení řádově během desítek milisekund.

3.6 Rychlostní profil v zatáčkách

Konstantní rychlost po celé trati je z hlediska celkového času průjezdu neoptimální – na rovinkách by vozidlo mohlo jet výrazně rychleji a před zatáčkami musí naopak zpomalit, aby nepřekročilo mez přilnavosti pneumatik. Rychlostní profil přiřazuje každému bodu trajektorie maximální bezpečnou rychlost na základě lokální křivosti trati a dynamických omezení vozidla. Výpočet se skládá ze dvou kroků: odhad poloměru zatáčky z geometrie trajektorie a propagace rychlostních omezení podél trati [35].

3.6.1 Odhad poloměru zatáčky

Pro výpočet maximální rychlosti v daném bodě trajektorie je nutné znát poloměr zatáčky, tedy poloměr kružnice, která v daném bodě nejlépe aproximuje tvar trati. Tento poloměr se odhaduje z

diskretizované středové čáry pomocí tří po sobě jdoucích bodů P_{i-1}, P_i a P_{i+1}.

Ze tří bodů v rovině lze analyticky určit poloměr kružnice, která jimi prochází. Používá se vztah založený na obsahu trojúhelníku a délkách jeho stran:

$$R_i = \frac{|P_{i-1} - P_i| \cdot |P_i - P_{i+1}| \cdot |P_{i+1} - P_{i-1}|}{4 \cdot A} \tag{3.20}$$

kde A je obsah trojúhelníku tvořeného body P_{i-1}, P_i, P_{i+1}, vypočtený například pomocí vektorového součinu. Pokud jsou tři body kolineární, obsah trojúhelníku je nulový a poloměr diverguje k nekonečnu, což odpovídá přímému úseku trati.

Maximální rychlost v bodě P_i je pak omezena boční přilnavostí pneumatik. Za předpokladu jízdy po kružnicovém oblouku platí rovnováha mezi dostředivou silou a třecí silou:

$$v_{max,i} = \sqrt{\mu \cdot g \cdot R_i} \tag{3.21}$$

kde μ je koeficient přilnavosti pneumatik, g je gravitační zrychlení a R_i je poloměr zatáčky v bodě i. Tento vztah předpokládá zjednodušený model bez přenosu zatížení a aerodynamického přítlaku, což je pro účely simulace na nižších rychlostech dostačující.

Pro vyšší rychlosti, kde už nelze zanedbat aerodynamický přítlak, je nutné rozšířit model o přídavnou normálovou sílu generovanou aerodynamickými prvky vozidla [36]. Aerodynamický přítlak roste s druhou mocninou rychlosti a zvyšuje efektivní zatížení pneumatik, čímž umožňuje vyšší boční zrychlení. Rovnováha sil v zatáčce pak vede ke vztahu:

$$v_{max,i} = \sqrt{\frac{\mu\, g\, R_i}{1 - \mu\, k_a\, R_i}}, \qquad k_a = \frac{\rho\, C_L\, S}{2m} \tag{3.22}$$

kde ρ je hustota vzduchu, C_L je koeficient přítlaku, S je čelní plocha vozidla a m je hmotnost vozidla. Tento vztah vychází z rovnováhy mezi dostředivou silou mv^2/R_i a maximální boční třecí silou $\mu(mg + \frac{1}{2}\rho C_L S v^2)$, kde druhý člen v závorce reprezentuje aerodynamický přítlak. Z tohoto vztahu je patrné, že přítlak umožňuje dosažení vyšších rychlostí v zatáčkách, přičemž jeho vliv roste s rychlostí vozidla.

3.6.2 Forward a backward propagace

Samotné omezení rychlosti na základě křivosti nestačí – vozidlo musí před zatáčkou včas zpomalit a po jejím projetí může zrychlovat pouze v mezích podélné přilnavosti. Tato omezení se zohledňují dvouprůchodovým algoritmem forward-backward propagace.

V prvním průchodu (backward) se prochází trajektorie od konce směrem k začátku. Pro každý bod se ověřuje, zda je rychlost v následujícím bodě dosažitelná při maximálním brzdném zpomalení a_{brake}. Pokud nikoli, rychlost se sníží podle kinematického vztahu:

$$v_i = \min\left(v_{max,i},\ \sqrt{v_{i+1}^2 + 2 \cdot a_{brake} \cdot \Delta s}\right) \tag{3.23}$$

kde Δs je vzdálenost mezi body P_i a P_{i+1}. Tento průchod zajistí, že vozidlo začne brzdit dostatečně brzy před každou zatáčkou.

Ve druhém průchodu (forward) se prochází trajektorie od začátku ke konci a aplikuje se omezení na maximální zrychlení a_{accel}:

$$v_i = \min\left(v_i,\ \sqrt{v_{i-1}^2 + 2 \cdot a_{accel} \cdot \Delta s}\right) \tag{3.24}$$

Tento průchod zajistí, že rychlostní profil respektuje omezené zrychlení vozidla – po výjezdu z pomalé zatáčky vozidlo zrychluje postupně a nedosáhne maximální rychlosti okamžitě.

Výsledkem obou průchodů je rychlostní profil, který v každém bodě trajektorie respektuje tři omezení současně: maximální boční zrychlení dané křivostí, maximální podélné zrychlení a maximální brzdné zpomalení. Celková maximální rychlost je navíc shora ohraničena parametrem v_{top}, který reflektuje výkonové limity pohonné jednotky.

Kapitola 4

Implementace a experimenty

V této kapitole je popsána implementaci celé autonomní pipeline v simulátoru CARLA – od přípravy prostředí přes detekci kuželů a SLAM až po řízení vozidla. Systém pokrývá dva režimy odpovídající disciplínám DV Cupu: reaktivní řízení pro první průjezd neznámou tratí (DV Autocross) a závodní režim s využitím mapy (Trackdrive).

4.1 Příprava simulačního prostředí

Před testováním pipeline bylo potřeba připravit simulační prostředí – nainstalovat CARLA, importovat vlastní modely kuželů a monopostu a vytvořit nástroj pro generování tratí. Experimenty byly prováděny na stolním počítači s procesorem AMD Ryzen 5800X, grafickou kartou NVIDIA GeForce RTX 5070 Ti, 32 GB RAM a Ubuntu 20.04.

4.1.1 Instalace a konfigurace CARLA

Byla použita CARLA verze 0.9.16 na Unreal Engine 4 a Ubuntu 20.04. Místo předkompilovaného balíčku byl zvolen build ze zdrojového kódu, protože předkompilovaná verze neumožňuje import vlastních 3D modelů – obsahuje pouze zabalené assety bez přístupu do Unreal Editoru.

Proces sestavení CARLA ze zdrojového kódu zahrnuje instalaci závislostí (Unreal Engine 4, kompilátor, Python API), stažení repozitáře CARLA, kompilaci enginu a sestavení Python klienta. Po úspěšné kompilaci je simulátor přístupný prostřednictvím Unreal Editoru, ve kterém lze upravovat existující mapy, přidávat vlastní assety a konfigurovat parametry prostředí. Veškerá komunikace s autonomním systémem pak probíhá přes Python API simulátoru.

4.1.2 Import kuželů a tvorba tratě

Dopravní kužely odpovídající specifikaci Formula Student [6] byly namodelovány v open-source nástroji Blender. Celkem byly vytvořeny čtyři varianty kuželů: tři malé kužely o výšce 325 mm

v modré, žluté a oranžové barvě a jeden velký oranžový kužel vysoký 505 mm. Modely byly exportovány ve formátu FBX, který je kompatibilní s Unreal Engine, a importovány do CARLA prostřednictvím Unreal Editoru. Každému kuželu byl přiřazen odpovídající materiál s barvou a odrazivostí, přibližující reálné vlastnosti plastových kuželů používaných na závodech. Výsledek je na obrázku 4.2.

Základem testovací mapy je jedna z předpřipravených map městského prostředí. Z této mapy byly následně odstraněny všechny objekty s výjimkou původních zdrojů světla (DirectionalLight a SkyLight). Následně byla vytvořena plocha (Plane) o rozměrech 10×10 kilometrů. Té byla přiřazena předpřipravená textura asfaltu a fyzika pro zamezení propadávání vozidla skrze tuto plochu.

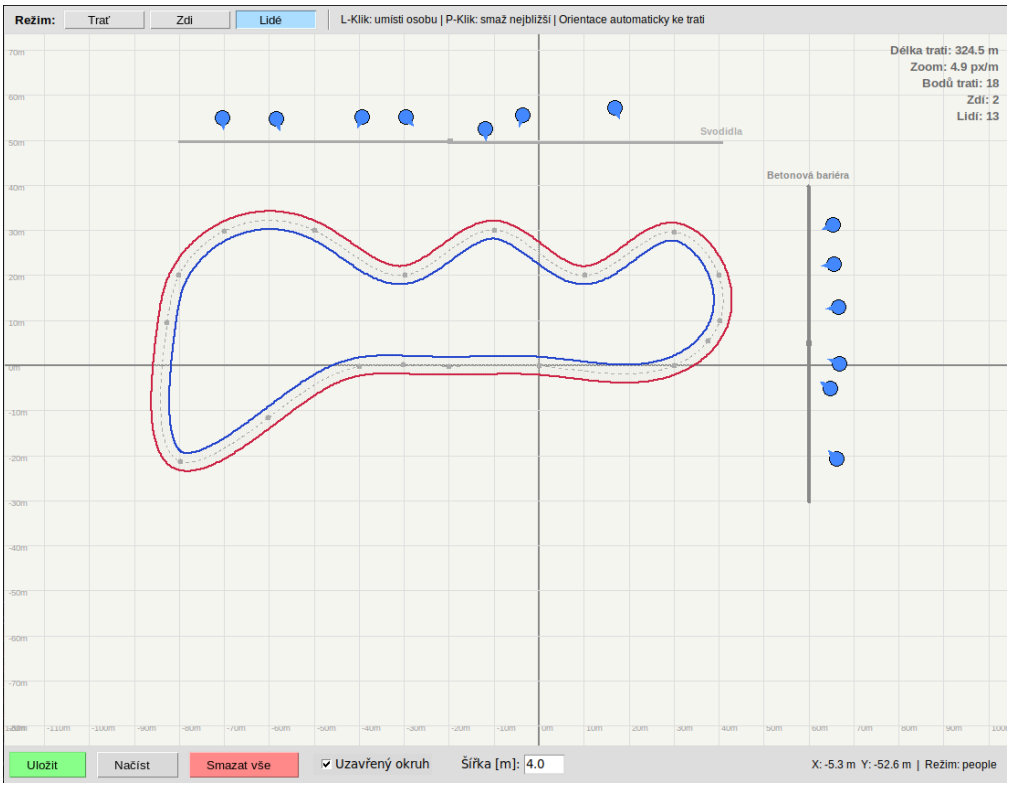
Pro tvorbu tratí byl vytvořen vlastní nástroj s grafickým rozhraním v knihovně Tkinter, který lze vidět na obrázku 4.1. Nástroj zobrazuje mřížku, ve které uživatel pomocí klikání definuje řídící body trati. Tyto body se proloží splajnovou křivkou, čímž vznikne hladká středová čára trati. Podél této čáry se automaticky vygenerují pozice kuželů v souladu s pravidly DV Cupu – modré kužely na levé straně, žluté na pravé, se vzdáleností přibližně 5 metrů mezi sousedními kužely na jedné straně a minimální šířkou trati 3 metry. Oranžové kužely označující startovní a cílovou zónu se umisťují na začátek a na konec trati. Nástroj dále umožňuje přidávat do okolí trati dva typy bariér – svodidla a betonové zábrany známé z dálnic a náměstí – a také postavy diváků podél trati. Tyto prvky slouží k testování robustnosti percepčního systému v přítomnosti objektů, které nejsou kužely a které by mohly být chybně detekovány.

Výstupem nástroje je konfigurační soubor s pozicemi všech kuželů a landmarků, který se následně načte pomocí Python API CARLA a objekty se programově rozmístí na zvolené mapě. Díky tomuto přístupu je možné rychle generovat různé konfigurace tratí bez nutnosti ručně umisťovat jednotlivé objekty do virtuální scény.

4.1.3 Konfigurace vozidla

Jako model monopostu byl použit 3D model MAD Formula Team MFT02 [37], vytvořený týmem MAD Formula Team z Universidad Carlos III de Madrid a volně dostupný na platformě OverTake.gg. Model byl původně vytvořen pro simulátor Assetto Corsa a vychází z CAD dat reálného monopostu sezóny 2022. Pro použití v CARLA byl model převeden do formátu FBX a importován do Unreal Editoru, kde byl přiřazen jako nový typ vozidla (vehicle blueprint). Model je na obrázku 4.2.

Fyzikální vlastnosti vozidla – hmotnost, rozvor, rozchod kol, polohu těžiště a parametry pneumatik – byly nakonfigurovány tak, aby se přibližně blížily monopostu Formula Student. Přesná kalibrace dynamiky nebyla cílem práce. Důležité bylo, aby se vůz choval kvalitativně podobně – nízké těžiště, krátký rozvor a dobrá akcelerace.



Obrázek 4.1: Náhled aplikace pro tvorbu vlastních tratí

4.2 Výběr a konfigurace senzorů

Konfigurace senzorů přímo ovlivňuje, jak daleko a jak spolehlivě systém vidí kužely. CARLA umožňuje senzory libovolně měnit – typ, rozlišení, zorné pole, frekvenci i umístění na vozidle. Byly porovnány tři LiDARy z hlediska detekce kuželů na různé vzdálenosti a otestováno několik konfigurací kamery.

4.2.1 Porovnání LiDARů

Tři LiDARy byly porovnány podle toho, kolik bodů dopadne na standardní malý kužel (výška 325 mm) na vzdálenostech 1 až 30 metrů. Jako metrika byl zvolen medián počtu bodů na kuželu – minimální práh pro spolehlivou detekci algoritmem DBSCAN jsou 3 body. Specifikace senzorů jsou v tabulce 4.1, výsledky na obrázku 4.3. Testovány byly LiDARy Velodyne VLP-16 [38], Ouster OS2-128 [39] a Valeo SCALA 2 [40].

Z grafu na obrázku 4.3 je patrné, že počet bodů na kuželu klesá s rostoucí vzdáleností u všech tří senzorů, avšak rychlost tohoto poklesu se výrazně liší. Velodyne VLP-16 se svými 16 kanály dosahuje maximální vzdálenosti detekce pouze přibližně 6 metrů, což je pro autonomní jízdu na FS trati nedostatečné – vůz jedoucí rychlostí 60 km/h potřebuje detekovat kužely alespoň na 15

Obrázek 4.2: Monopost MFT02 s vlastními kužely v simulátoru CARLA

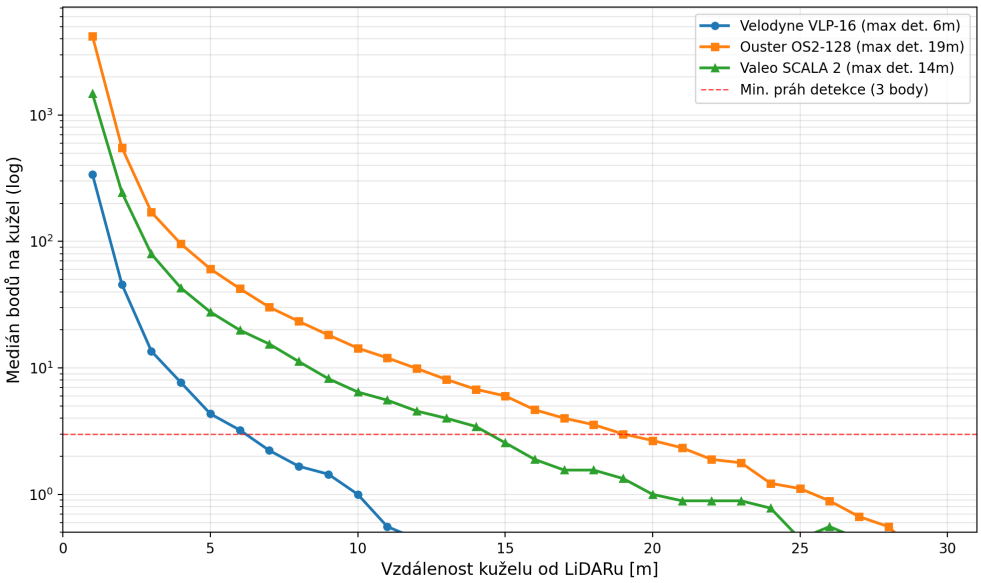
Parametr	**Velodyne VLP-16**	**Ouster OS2-128**	**Valeo SCALA 2**
Kanály	16	128	16
Bodů/s	300 000	2 621 440	705 400
Vert. FOV	-15,0/+15,0	-11,25/+11,25	-5,0/+5,0
Dosah	100 m	200 m	150 m
Frekvence	20 Hz	20 Hz	25 Hz

Tabulka 4.1: Specifikace porovnávaných LiDARů

metrů, aby měl dostatek času na naplánování trajektorie. Ouster OS2-128 se 128 kanály detekuje kužely až do vzdálenosti přibližně 19 metrů, čímž jako jediný ze tří senzorů splňuje tento požadavek. Valeo SCALA 2 navzdory pouze 16 kanálům dosahuje maximální detekce kolem 14 metrů díky vyšší frekvenci bodů na kanál a užšímu vertikálnímu zornému poli, které koncentruje paprsky do menšího úhlového rozsahu.

Tabulka 4.2 ukazuje, že dominantní fází je RANSAC segmentace roviny země, která tvoří 88–90% celkového času u všech tří senzorů. Čtení surových dat a FOV filtr jsou zanedbatelné — numpy operace nad polem float32 jsou řádově rychlejší než geometrické výpočty Open3D. DBSCAN clustering je překvapivě rychlý i u Ousteru, protože pracuje pouze nad body po odstranění země, jejichž počet je výrazně nižší.

Velodyne VLP-16 dosahuje celkového času 5,69 ms, Valeo SCALA 2 pak 8,21 ms — rozdíl



Obrázek 4.3: Porovnání LiDARů – medián počtu bodů na kuželu v závislosti na vzdálenosti

odpovídá vyššímu počtu bodů na snímek způsobenému škálováním z reálného HFOV 133° na 360° v simulátoru. Ouster OS2-128 s časem 19,01 ms zatěžuje pipeline nejvíce, přičemž prodloužení je způsobeno výhradně RANSAC segmentací nad hustším mrakem bodů (128 kanálů, 2,6 mil. bodů/s). Vzhledem k tomu, že LiDARy pracují na frekvenci 20–25 Hz (perioda 40–50 ms), všechny tři senzory splňují požadavek na real-time zpracování s dostatečnou rezervou pro ostatní fáze pipeline.

LiDAR	Čtení [ms]	Filtr [ms]	RANSAC [ms]	DBSCAN [ms]	Celkem [ms]
Velodyne VLP-16	0.02	0.20	5.13	0.23	**5.69**
Ouster OS2-128	0.03	1.42	17.16	0.25	**19.01**
Valeo SCALA 2	0.03	0.44	7.40	0.24	**8.21**

Tabulka 4.2: Mediánové časy zpracování LiDAR pipeline

Na základě tohoto srovnání byl pro implementaci zvolen LiDAR s konfigurací odpovídající Ouster OS2-128, protože jako jediný ze tří porovnávaných senzorů spolehlivě detekuje kužely na vzdálenost dostatečnou pro bezpečnou jízdu na typické FS trati.

4.2.2 Konfigurace kamery

Obdobně byly otestovány čtyři rozlišení kamery. Měřeny byly mediánové časy čtení snímku, projekce LiDAR bodů, ořezu výřezů a inference CNN. Výsledky jsou v tabulce 4.3.

Z dat je patrné, že rychlost čtení se liší podle rozlišení. U nižších rozlišeních (960×600 a 1280×720) jsou čtení snímku a projekce přibližně vyrovnané, zatímco u vyšších rozlišeních (1920×1080

a 2560×1440) se čtení snímku stává výrazně dominantní složkou — u rozlišení 2560×1440 tvoří přes 75 % celkového času. Projekce LiDAR bodů je naopak na rozlišení prakticky nezávislá a pohybuje se stabilně v rozsahu 5,8–7,0 ms napříč všemi konfiguracemi, což odpovídá očekávání: projekce závisí na počtu bodů mračna, nikoli na rozlišení obrazu. Crop a CNN inference jsou zanedbatelné — ořez je čistě numpy operace a CNN pracuje nad malými výřezy, nikoli nad celým snímkem.

LiDAR Ouster OS2-128 zvolený v předchozí sekci zpracovává snímky na frekvenci 20 Hz (perioda 50 ms). Aby kamera nebyla úzkým hrdlem pipeline, byl stanoven požadavek, že čas zpracování kamery nesmí přesáhnout čas zpracování LiDAR pipeline (19,01 ms). Rozlišení 1920×1080 se s hodnotami 19,36–20,71 ms pohybuje na hraně tohoto limitu. Rozlišení 2560×1440 s celkovými časy 28–30 ms tento limit jednoznačně překračuje. Rozlišení 960×600 (≈ 10,5 ms) a 1280×720 (≈ 13,6 ms) oba splňují požadavek s dostatečnou rezervou.

Bylo zvoleno rozlišení 1280×720 s FOV 120°. Oproti 960×600 má lepší prostorové rozlišení pro detekci kuželů a výkonově leží mezi testovanými konfiguracemi 1280×720 a 1920×1080, tedy bezpečně pod limitem LiDAR pipeline. FOV 120° odpovídá horizontálnímu záběru LiDARu, takže oba senzory vidí přibližně stejnou oblast.

Rozlišení [px]	Čtení [ms]	Projekce [ms]	Crop [ms]	ConeCNN [ms]	Celkem [ms]
960×600	3,59	5,95	0,03	0,97	**10,54**
1280×720	5,73	6,67	0,03	1,28	**13,71**
1920×1080	12,31	6,27	0,03	0,94	**19,55**
2560×1440	22,18	6,50	0,03	0,93	**29,63**

Tabulka 4.3: Mediánový čas jednotlivých fází pipeline kamery

4.2.3 Umístění senzorů na vozidle

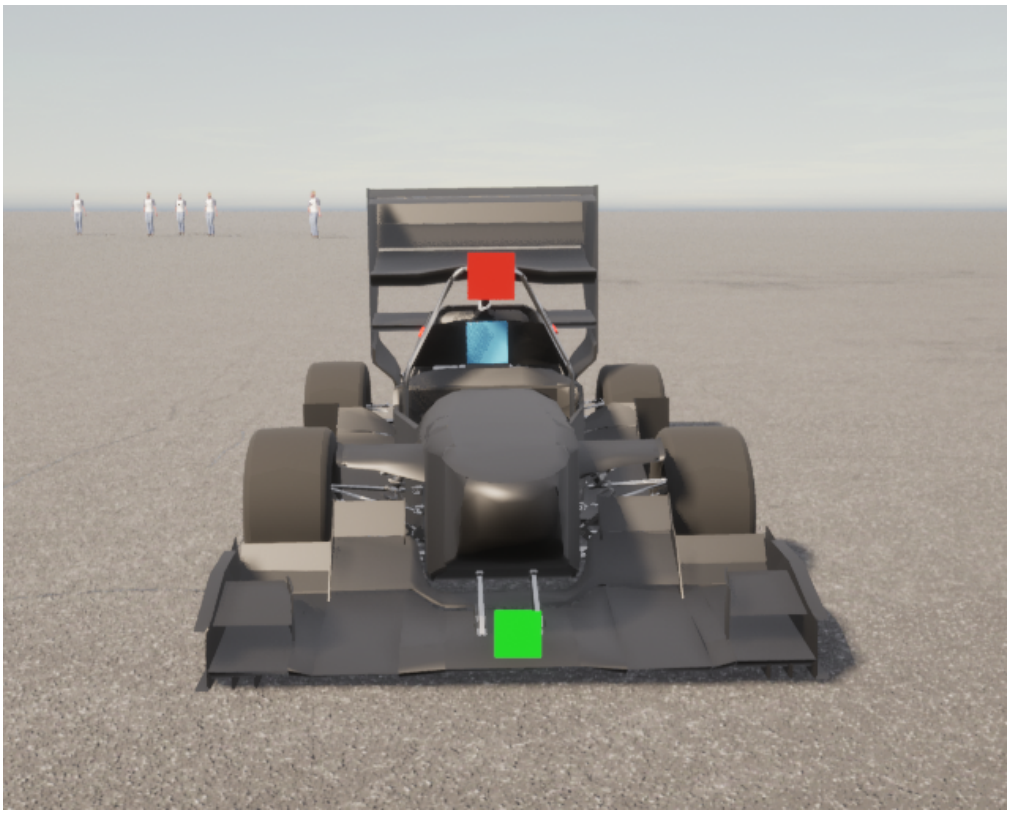
Rozmístění senzorů na vozidle vychází z konstrukčních omezení monopostu a požadavků na co nejlepší výhled do prostoru před vozem. Přehled umístění je znázorněn na obrázku 4.4.

LiDAR je umístěn na předním křídle (front wing), čímž získává nízký pozorovací úhel těsně nad vozovkou. Tato poloha maximalizuje počet paprsků dopadajících na spodní část kuželů v blízkém okolí vozu, avšak způsobuje, že střed záběru LiDARu je posunut před střed vozu přibližně o 1,55 metru.

Kamera je uchycena na mainhoopu, přibližně ve výši hlavy řidiče. Oproti LiDARu má tedy vyšší pozorovací bod a mírně odlišnou perspektivu — zejména v blízkosti vozu se úhly záběru obou senzorů liší. Pro zachování překryvu mezi obrazovými daty a bodovým mrakem bylo zvoleno FOV 120°, které kompenzuje tento perspektivní rozdíl a zajišťuje, že kužely detekované LiDARem leží s vysokou pravděpodobností i v záběru kamery.

IMU a GNSS jsou umístěny uvnitř karoserie v blízkosti těžiště vozu. Tato poloha je standardní praxí — měření IMU jsou nejpřesnější, pokud senzor leží co nejblíže těžišti, protože se minimalizují parazitní zrychlení způsobená rotací vozidla kolem těžiště.

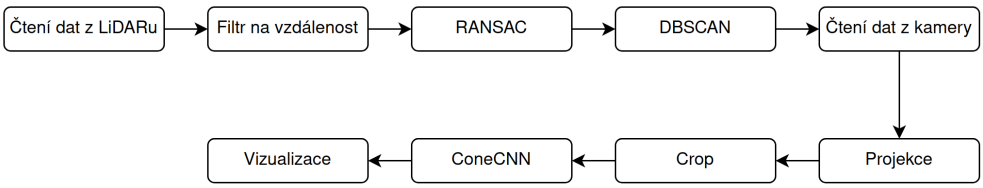
Všechny transformace mezi souřadnicovými systémy jednotlivých senzorů a rámcem vozidla jsou definovány staticky v konfiguraci simulátoru CARLA prostřednictvím Python API, kde jsou senzory připojeny k vozidlu s pevně zadanou polohou a orientací.



Obrázek 4.4: Umístění LiDARu (zelený čtverec) a kamery (červený čtverec) na monopostu

4.3 Detekce a klasifikace kuželů

Celý percepční systém stojí na třech krocích: LiDAR detekuje kandidáty na kužely v mračnu bodů, ty se promítnou do obrazu kamery a konvoluční síť klasifikuje jejich barvu. Na výstupu je seznam kuželů s 3D polohou a barevnou třídou (modrý, žlutý, oranžový, nekužel). Blokové schéma pipeline je na obrázku 4.5, vizualizace jednotlivých kroků na obrázku 4.6.



Obrázek 4.5: Blokové schéma pro detekci kuželů

4.3.1 LiDAR pipeline

Zpracování dat z LiDARu sleduje postup popsaný v kapitole o metodách. V každém skenovacím cyklu se přijme mračno bodů a zpracuje se ve dvou krocích. Prvním krokem je odstranění roviny vozovky pomocí algoritmu RANSAC. Algoritmus iterativně hledá rovinu s největším počtem inlierů, přičemž prahová vzdálenost pro zařazení bodu k rovině je nastavena na 0,05 m. Body klasifikované jako součást vozovky se z mračna odstraní.

Na zbývající body se aplikuje algoritmus DBSCAN s parametry $\varepsilon = 0{,}6$ m a $MinPts = 4$. Každý nalezený shluk představuje kandidáta na objekt na trati. Shluky se následně filtrují podle počtu bodů a prostorových rozměrů – příliš velké shluky (odpovídající bariérám nebo divákům) a příliš malé shluky (šum) se vyřazují. Pro každý zbývající shluk se vypočítá centroid jako průměr souřadnic všech bodů ve shluku. Tento centroid reprezentuje odhadovanou polohu kužele v prostoru.

4.3.2 Projekce centroidů do kamery

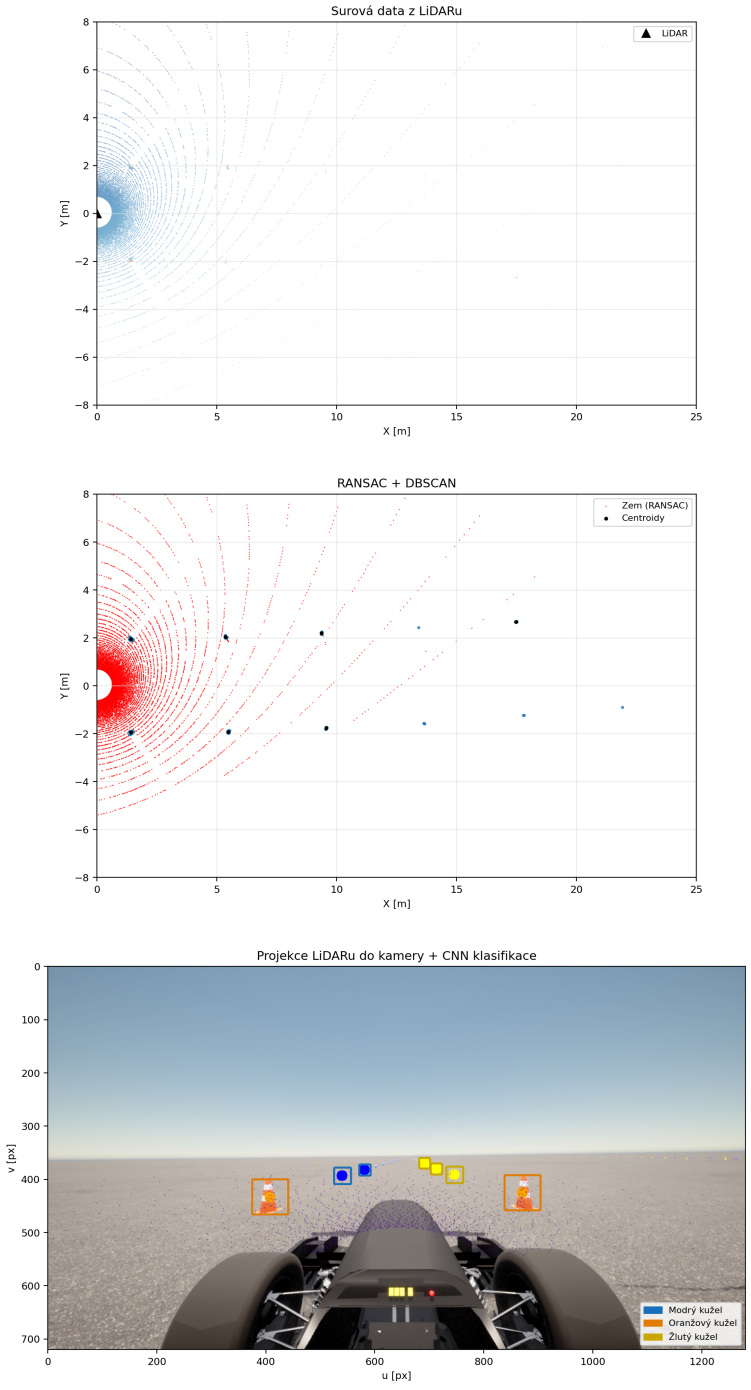
Centroid každého shluku se promítne do obrazu kamery projekční maticí (rovnice 3.6). Body za kamerou ($Z_C \leq 0$) nebo mimo rozlišení obrazu se vyřadí.

Kolem každého promítnutého bodu se v obraze vyřízne výřez (crop), jehož velikost se určuje automaticky na základě známé fyzické velikosti kužele a jeho vzdálenosti od kamery. Ze vzdálenosti a rozměrů kužele se dopočítá odpovídající počet pixelů v obraze a výřez se přeškáluje na vstupní rozlišení sítě 32×32 pixelů. Díky tomu je kužel ve výřezu vždy přibližně stejně velký, bez ohledu na vzdálenost od senzoru. Tyto výřezy slouží jako vstup pro klasifikační neuronovou síť.

4.3.3 ConeCNN

Pro klasifikaci barvy kužele z výřezu obrazu byla navržena lehká konvoluční neuronová síť nazvaná ConeCNN. Síť klasifikuje vstupní výřez do čtyř tříd: modrý kužel, žlutý kužel, oranžový kužel a nekužel (pozadí, bariéra nebo jiný objekt). Architektura sítě je shrnuta v tabulce 4.4.

Síť se skládá ze tří konvolučních bloků, z nichž každý obsahuje konvoluční vrstvu s jádrem 3×3, aktivační funkci ReLU a pooling vrstvu. Počet filtrů se postupně zdvojnásobuje z 32 na 64 a poté na 128. Po třetím bloku se místo klasického max poolingu používá adaptivní průměrový pooling (AdaptiveAvgPool), který výstup zredukuje na vektor o délce 128 bez ohledu na prostorové rozměry



Obrázek 4.6: Jednotlivé kroky detekční pipeline

vstupu. Výstupní lineární vrstva mapuje tento vektor na 4 třídy. Celá síť má pouze 93 764 parametrů a zabírá 0,85 MB.

Vrstva	Vstup	Výstup	Parametry	Kernel
Conv2d	$3 \times 32 \times 32$	$32 \times 32 \times 32$	896	3×3
ReLU	$32 \times 32 \times 32$	$32 \times 32 \times 32$	–	–
MaxPool2d	$32 \times 32 \times 32$	$32 \times 16 \times 16$	–	2×2
Conv2d	$32 \times 16 \times 16$	$64 \times 16 \times 16$	18 496	3×3
ReLU	$64 \times 16 \times 16$	$64 \times 16 \times 16$	–	–
MaxPool2d	$64 \times 16 \times 16$	$64 \times 8 \times 8$	–	2×2
Conv2d	$64 \times 8 \times 8$	$128 \times 8 \times 8$	73 856	3×3
ReLU	$128 \times 8 \times 8$	$128 \times 8 \times 8$	–	–
AdaptiveAvgPool	$128 \times 8 \times 8$	$128 \times 1 \times 1$	–	–
Flatten	$128 \times 1 \times 1$	128	–	–
Linear	128	4	516	–
Celkem parametrů			**93 764**	
Velikost modelu			0,85 MB	
Multiply-adds			10,38 M	

Tabulka 4.4: Architektura ConeCNN – přehled vrstev a parametrů

4.3.3.1 Dataset a trénování

Trénovací data byla získána manuálním řízením vozidla v simulátoru CARLA pomocí ovladače Xbox připojeného přes knihovnu PyGame. Během jízdy po různých konfiguracích tratí se v každém skenu LiDARu detekovaly shluky bodů a vypočítaly se jejich centroidy. Tyto centroidy se promítly do obrazu kamery pomocí projekční matice a kolem každého promítnutého bodu se vyřízl výřez. Velikost výřezu se určuje automaticky na základě známé fyzické velikosti kužele a jeho vzdálenosti od kamery a výřez se přeškáluje na vstupní rozlišení sítě 32×32 pixelů.

Získané výřezy byly ručně roztříděny do čtyř tříd. Celkem bylo získáno 6 674 výřezů. Dataset byl rozdělen stratifikovaným způsobem na trénovací a validační část v poměru 80:20. Rozložení tříd shrnuje tabulka 4.5.

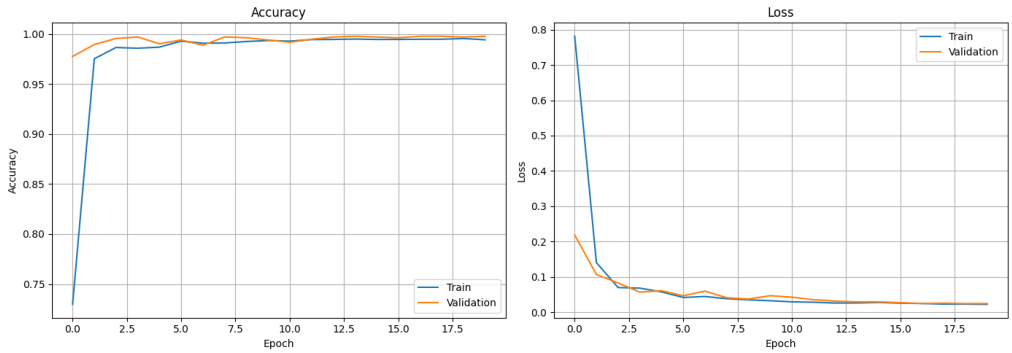
Třída	Trénovací	Validační	Celkem
Modrý kužel	1 374	343	1 717
Žlutý kužel	1 354	339	1 693
Oranžový kužel	1 475	369	1 844
Nekužel	1 136	284	1 420
Celkem	**5 339**	**1 335**	**6 674**

Tabulka 4.5: Rozložení tříd v datasetu pro trénování ConeCNN

Na trénovací data byla aplikována augmentace: náhodný ořez s paddingem 4 pixely, horizontální zrcadlení s pravděpodobností 50 % a mírné změny jasu, kontrastu a saturace (ColorJitter). Změna

odstínu (hue) byla záměrně omezena na ±0,01, aby se zabránilo záměně oranžové a žluté barvy. S pravděpodobností 10 % se navíc přidává gaussovský šum o směrodatné odchylce 0,01.

Síť byla trénována optimalizátorem Adam s počátečním learning rate 1×10^{-3} a regularizací weight decay 1×10^{-4} po dobu 20 epoch. Learning rate byl adaptivně snižován plánovačem ReduceLROnPlateau, který při stagnaci validační přesnosti po 3 epochách snížil krok na polovinu. Jako ztrátová funkce byla použita křížová entropie. Průběh trénování je zobrazen na obrázku 4.7.



Obrázek 4.7: Accuracy a loss během tréninku ConeCNN

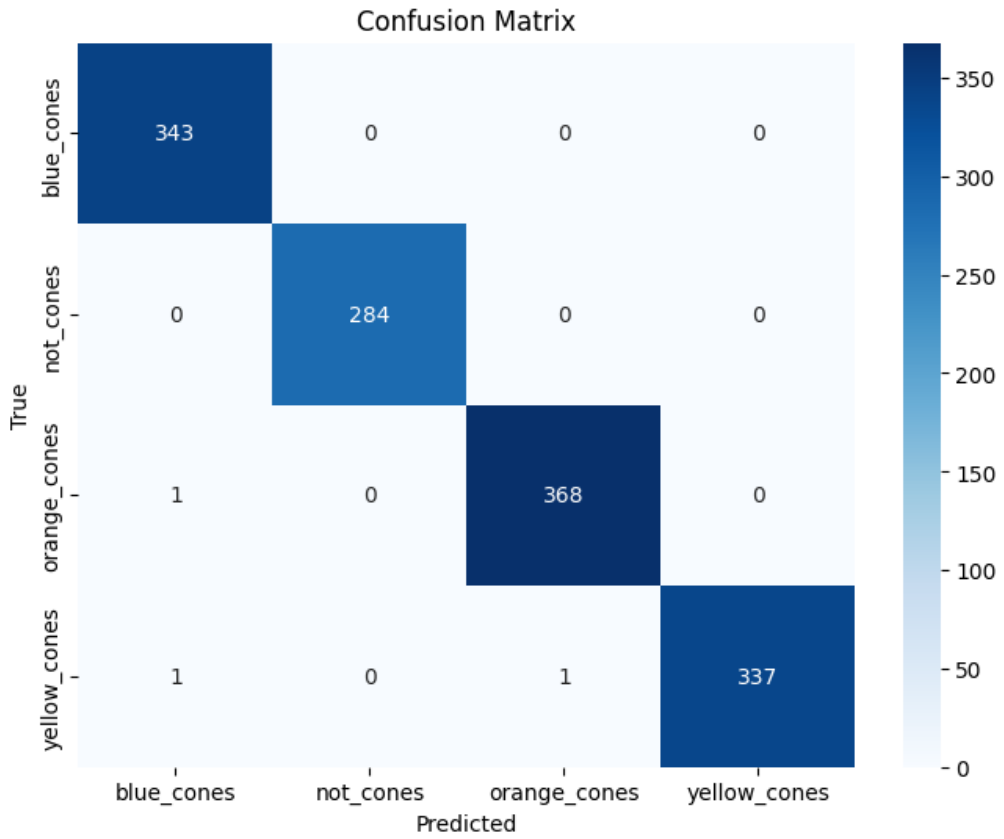
4.3.3.2 Výsledky klasifikace

Výsledky klasifikace na validační sadě jsou zobrazeny v matici záměn (confusion matrix) na obrázku 4.8. Z celkového počtu 1 335 testovacích vzorků byla síť schopna správně klasifikovat 1 332 vzorků, což odpovídá přesnosti 99,85%. Došlo pouze ke třem záměnám – jeden oranžový kužel byl klasifikován jako modrý, jeden žlutý kužel jako modrý a jeden žlutý kužel jako oranžový. Třída no_cone byla klasifikována zcela bezchybně, což je důležité pro eliminaci falešných detekcí z bariér a diváků podél trati.

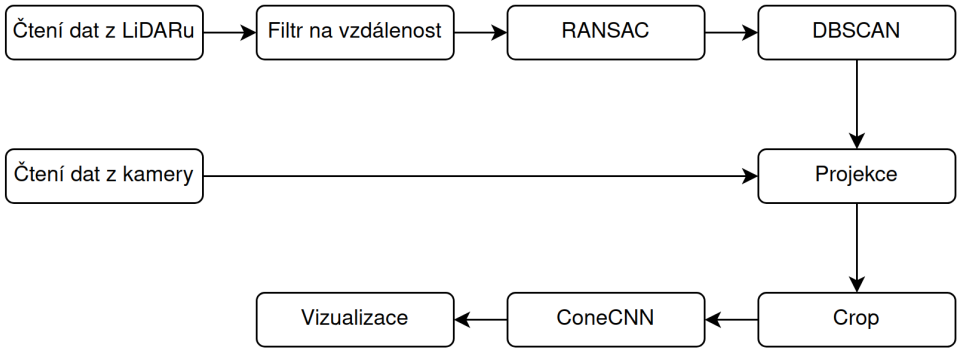
Nízká chybovost klasifikátoru potvrzuje, že přístup založený na projekci LiDARových centroidů do obrazu a následné klasifikaci malých výřezů pomocí lehké CNN je pro prostředí Formula Student dostatečně přesný. Díky malé velikosti sítě (0,85 MB) a nízkému počtu operací (10,38 M multiply-adds) je klasifikace výpočetně nenáročná a nebrzdí zbytek pipeline. Na stolním počítači s grafickou kartou NVIDIA RTX 5070 Ti dosahovala síť rychlosti klasifikace přibližně 3200 obrázků za sekundu, tudíž průměrný čas pro klasifikaci jednoho výřezu je 0,31 ms.

4.4 Optimalizace a paralelizace detekční pipeline

Původní pipeline zpracovávala data sekvenčně – kamera, pak LiDAR, pak klasifikace. Jednotlivé fáze přitom pracují s nezávislými daty na různých jednotkách (CPU pro LiDAR, GPU pro CNN), takže na sebe zbytečně čekaly. Pro real-time zpracování při 20 Hz bylo potřeba pipeline zrychlit jak na úrovni algoritmů, tak paralelizací čtení senzorů. Výsledné schéma je na obrázku 4.9.



Obrázek 4.8: Matice záměn pro ConeCNN na validační sadě



Obrázek 4.9: Paralelizované schéma systému pro detekci kuželů

4.4.1 Optimalizace LiDAR pipeline

Největší časovou úsporou v LiDAR pipeline bylo zavedení vzdálenostního předfiltru před algoritmem RANSAC. Původní implementace předávala algoritmu RANSAC celé mračno bodů po FOV filtru, včetně bodů vzdálených desítky metrů, které pro detekci kuželů nejsou relevantní. Přidáním filtru

na maximální vzdálenost 15 metrů se počet bodů vstupujících do RANSAC výrazně snížil, protože většina bodů v mračnu pochází z vozovky a okolních ploch za hranicí užitečného dosahu. Vzhledem k tomu, že segmentace roviny vozovky pomocí RANSAC tvoří přibližně 90 % celkového času LiDAR pipeline, i mírné snížení počtu vstupních bodů má znatelný dopad na celkový čas zpracování.

Druhou optimalizací bylo snížení počtu iterací algoritmu RANSAC z 500 na 150. Na závodní trati Formula Student je vozovka téměř dokonale rovinná bez výrazných nerovností, takže algoritmus konverguje k správné rovině již po několika desítkách iterací. Empirické testování potvrdilo, že snížení počtu iterací nezhoršilo kvalitu segmentace – počet chybně klasifikovaných bodů zůstal pod 0,1 %.

Třetí optimalizací bylo zavedení filtru na maximální velikost shluku v algoritmu DBSCAN. Shluky obsahující více než 100 bodů nemohou odpovídat dopravnímu kuželu – jedná se o bariéry nebo diváky. Jejich přeskočení šetří čas při výpočtu centroidů a snižuje počet výřezů předávaných klasifikační síti.

4.4.2 Dávková inference neuronové sítě

Původní implementace klasifikovala každý výřez zvlášť jedním voláním neuronové sítě. Režie spojená s přenosem dat mezi CPU a GPU a se spuštěním výpočetního jádra přitom dominovala nad samotným výpočtem, protože síť ConeCNN je velmi malá (93 764 parametrů). Optimalizovaná verze nejprve shromáždí všechny výřezy z daného snímku do jednoho dávkového tensoru a provede jedinou inferenci pro celou dávku. Při typickém počtu 5 až 15 viditelných kuželů v jednom snímku tato úprava snížila celkový čas klasifikace přibližně trojnásobně, protože se eliminuje opakovaná režie GPU volání.

4.4.3 Paralelní čtení senzorů

Čtení dat z kamery a LiDARu je v simulátoru CARLA blokující operace – volající vlákno čeká, dokud senzor nedodá nová data. V sekvenční implementaci se nejprve čte kamera a teprve po dokončení se čte LiDAR, přestože oba senzory generují data nezávisle na sobě.

Optimalizovaná implementace využívá fond vláken (`ThreadPoolExecutor`) se třemi pracovními vlákny. Čtení kamery a zpracování LiDARu (včetně RANSAC a DBSCAN) se spustí současně jako dvě nezávislé úlohy. Hlavní vlákno mezitím zpracovává data z IMU a odometrii, a poté si vyzvedne výsledky obou úloh. Díky tomu se čas čtení kamery (7ms) zcela překryje se zpracováním LiDARu (19 ms) a celková latence jednoho snímku se zkrátí o čas čtení kamery.

4.4.4 Výsledný čas zpracování

Tabulka 4.6 porovnává mediánové časy jednotlivých fází pipeline před optimalizací a po ní. Měření probíhalo v produkčním režimu s vypnutými debug výstupy. Celkový čas zpracování jednoho snímku klesl ze 39 ms na 14 ms, což odpovídá frekvenci přibližně 71 Hz. Jedná se o skoro trojnásobné

zrychlení oproti původní implementaci a poskytuje výraznou rezervu vůči frekvenci senzorů 20 Hz. Při zapnuté debug vizualizaci pro účely vývoje se mediánová latence prodlužuje přibližně o 22 ms.

Fáze pipeline	**Před [ms]**	**Po [ms]**
LiDAR (čtení + RANSAC + DBSCAN)	20	7
Kamera (čtení + dekódování)	7	
Projekce + crop	7	6
ConeCNN inference	5	1
Celkem	**39**	**14**

Tabulka 4.6: Porovnání mediánových časů pipeline před a po optimalizaci.

4.5 Reaktivní řízení

Při prvním průjezdu tratí (DV Autocross) vůz nezná rozmístění kuželů a nemá k dispozici mapu. Musí se orientovat výhradně podle kuželů, které právě vidí, a z nich průběžně generovat lokální trajektorii na pár metrů dopředu.

4.5.1 Generování lokální trajektorie

Z detekovaných kuželů se nejprve oddělí modré (levá hranice trati) a žluté (pravá hranice trati) podle klasifikace z ConeCNN. Pokud jsou v zorném poli dva oranžové kužely, mají prioritu – označují startovní nebo cílovou bránu a vozidlo je využije jako referenční body pro průjezd. Kužely na každé straně se seřadí podle vzdálenosti od vozidla a propojí se do linie tvořící levou a pravou hranici trati. Pro každou dvojici protilehlých bodů na levé a pravé hranici se vypočítá středový bod (midpoint). Výsledná sekvence středových bodů tvoří lokální trajektorii, kterou má vozidlo sledovat.

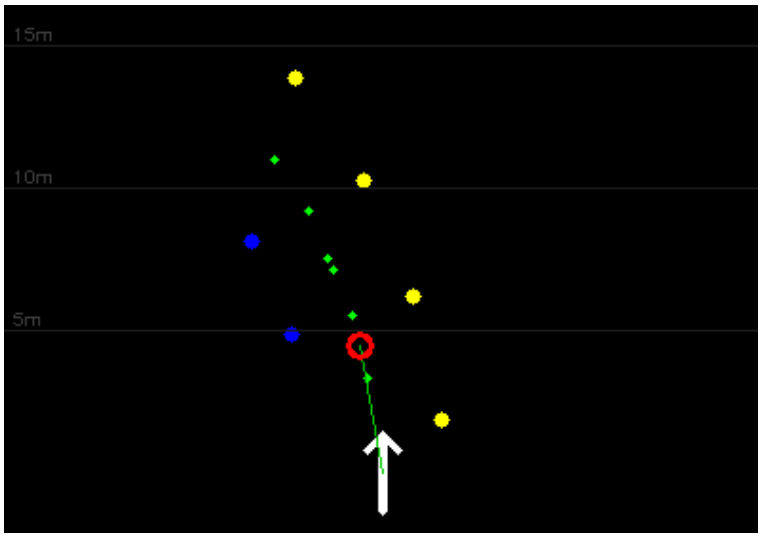
Tento přístup nevyžaduje žádnou znalost tvaru trati dopředu. Jeho omezením je, že kvalita trajektorie závisí na počtu a rozložení aktuálně viditelných kuželů – v ostrých zatáčkách, kde je viditelný úsek trati kratší, se trajektorie generuje pouze na několik metrů, což omezuje čas na reakci.

4.5.2 Pure Pursuit s fixním lookaheadem

Pro příčné řízení (natočení volantu) se používá algoritmus Pure Pursuit, jehož princip byl popsán v kapitole o metodách. Lookahead bod se hledá na vygenerované lokální trajektorii ve fixní vzdálenosti 4 m od zadní nápravy vozidla. Ukázka lookahead bodu v jednom detekčním snímku se nachází na obrázku 4.10. Natočení volantu se vypočítá podle rovnice 3.17.

Fixní lookahead vzdálenost je zvolena jako kompromis – dostatečně krátká na to, aby vozidlo sledovalo trajektorii v zatáčkách, a dostatečně dlouhá na to, aby nedocházelo ke kmitání na rovin-

kách. Pro závodní režim s vyšší rychlostí by byl vhodnější adaptivní lookahead závislý na rychlosti vozidla. Tato optimalizace je však ponechána na pokročilejší režim řízení.



Obrázek 4.10: Top-down na detekované kužely, středové body a lookahead bod pro reaktivní řízení

4.5.3 PID pro podélné řízení

Rychlost je regulována PID regulátorem na přibližně 18 km/h (minimální rychlost požadovaná pravidly FS Czech Republic [2]). Regulační odchylka je rozdíl požadované a aktuální rychlosti; kladný výstup jde na plyn, záporný na brzdu.

Pro reaktivní režim je konstantní rychlost dostačující – vůz projede trať bezpečně, i když pomalu. Proměnlivý rychlostní profil vyžaduje znalost celé trati, což je k dispozici až po dokončení SLAM mapy.

4.5.4 Zhodnocení reaktivního řízení

Reaktivní řízení bylo testováno na několika tratích. Vozidlo projede trať spolehlivě bez sražení kužele při 18 km/h, ale jede středem trati bez řezání zatáček. Porovnání se závodním režimem je v sekci o experimentech.

4.6 Lokalizace a mapování

Pro grafový SLAM (sekce 3.4) byla zvolena knihovnu GTSAM (Georgia Tech Smoothing and Mapping) [41] – open-source C++ knihovnu s Python rozhraním pro optimalizaci faktorových grafů. GTSAM je široce používaná v robotice i v FS Driverless týmech.

GTSAM pracuje s konceptem faktorového grafu [42], což je bipartitní graf skládající se z proměnných a faktorů. Proměnné reprezentují neznámé veličiny – v kontextu dvourozměrného SLAM jde o pozice vozidla typu Pose2 (x, y, theta) a souřadnice landmarků typu Point2 (x, y). Faktory kódují pravděpodobnostní omezení mezi proměnnými, přičemž každý faktor je parametrizován střední hodnotou měření a kovarianční maticí šumu. Knihovna nabízí předdefinované typy faktorů vhodné pro mobilní robotiku: PriorFactorPose2 pro ukotvení počáteční pozice, BetweenFactorPose2 pro odometrická omezení mezi po sobě jdoucími pozicemi a BearingRangeFactor2D pro měření úhlu a vzdálenosti k landmarku. Volitelně systém přijímá také data z GNSS senzoru. Pokud jsou k dispozici, přidává se pro aktuální pozici vozidla priórní faktor s kovarianční maticí 2,0 m v poloze a s prakticky neomezenou kovariancí v orientaci ($\sigma = 999$ rad), což efektivně vypíná omezení. Tento faktor slouží jako hrubá korekce absolutní pozice a zabraňuje dlouhodobému driftu odometrie v případech, kdy je počet pozorovaných landmarků dočasně nízký.

Standardní dávková optimalizace faktorového grafu vyžaduje přepočet celého systému při každém přidání nového faktoru, což je pro aplikace pracující v reálném čase nepraktické. Knihovna GTSAM proto implementuje algoritmus iSAM2 (incremental Smoothing and Mapping 2) [43], který při přidání nových faktorů přeoptimalizuje pouze tu část grafu, která je novými měřeními ovlivněna.

Systém pracuje ve dvou režimech. V režimu mapování vozidlo projíždí trať poprvé, buduje graf pozic a landmarků a průběžně optimalizuje jejich vzájemné polohy. Po uzavření smyčky – tedy opětovném pozorování obou oranžových kuželů startovní čáry – přepne systém do režimu lokalizace, ve kterém se mapa zmrazí a nová pozorování slouží výhradně k upřesnění aktuální polohy vozidla vůči existujícím landmarkům. Toto rozdělení zabraňuje neomezenému růstu grafu a zároveň eliminuje drift odometrie akumulovaný během prvního okruhu.

Faktorový graf implementovaného systému se skládá ze tří typů faktorů. Priórní faktor (PriorFactorPose2) ukotvuje počáteční pozici vozidla do počátku souřadného systému s kovarianční maticí o hodnotách 0,1 m v poloze a 0,05 rad v orientaci. Odometrické faktory (BetweenFactorPose2) spojují po sobě následující pozice vozidla a kódují relativní posunutí odhadnuté z IMU a rychlostního čidla. Měřicí faktory (BearingRangeFactor2D) propojují pozici vozidla s landmarkem pomocí úhlu a vzdálenosti odvozených z detekce kuželu v LiDARovém mračnu bodů. Šum odometrických faktorů je nastaven na 0,2 m v poloze a 0,1 rad v orientaci, šum měřicích faktorů pak na 0,1 m v obou osách.

Instance iSAM2 je vytvořena s výchozími parametry a aktualizována v každém snímku simulace. Nové odometrické a měřicí faktory se akumulují do dočasného grafu, který je v každém kroku předán optimalizátoru spolu s počátečními odhady nových proměnných. Po aktualizaci se dočasný graf vymaže a optimalizované hodnoty všech proměnných jsou dostupné pro další zpracování. Každých 50 snímků se navíc provádí dodatečná iterace optimalizátoru bez nových faktorů, která zajišťuje konvergenci v oblastech grafu, jež nebyly přímo ovlivněny posledními měřeními.

4.6.1 Odometrie a fúze GPS

Odometrie poskytuje relativní posunutí vozidla mezi dvěma po sobě jdoucími snímky a tvoří základ pro odometrické faktory ve faktorovém grafu. Implementace kombinuje data ze dvou senzorů: rychlost vozidla a kompasový kurz z IMU.

Přírůstek v podélném směru (dx) je vypočten jako součin aktuální rychlosti vozidla v metrech za sekundu a časového kroku simulace, který činí 1/30 sekundy. Příčný přírůstek (dy) je nastaven na nulu za předpokladu, že vozidlo se pohybuje bez bočního skluzu – zjednodušení, které je přijatelné při rychlostech do 30 km/h na trati s mírnými zatáčkami.

Změna orientace (dtheta) je odvozena jako rozdíl dvou po sobě jdoucích hodnot kompasu IMU. Protože kompas v simulátoru CARLA udává kurz ve směru hodinových ručiček, zatímco knihovna GTSAM pracuje s úhly rostoucími proti směru hodinových ručiček, je nutné výsledný rozdíl negovat. Zároveň je každý přírůstek orientace normalizován do intervalu $(-\pi, \pi)$ funkcí atan2, aby se zabránilo skokům při přechodu přes hranici 360 stupňů. Při prvním snímku po spuštění ještě neexistuje předchozí hodnota kurzu, proto je přírůstek orientace inicializován na nulu, čímž se zabrání nežádoucímu počátečnímu skoku v grafu.

Vedle odometrie systém volitelně přijímá data z GNSS senzoru, která slouží jako hrubá globální korekce pozice a zabraňují dlouhodobému driftu odometrie v situacích, kdy je počet pozorovaných kuželů dočasně nízký. Při prvním přijatém měření se aktuální zeměpisná poloha uloží jako referenční bod a všechna následná měření se vůči ní převádí do lokálního kartézského souřadného systému ekvirektangulární projekcí na sférickém modelu Země. Pro rozsah trati Formula Student (stovky metrů) je tato aproximace dostatečně přesná. Pro každé GPS měření se do faktorového grafu přidá priórní faktor *PriorFactorPose2* navázaný na aktuální pozici vozidla. Šum faktoru je nastaven na 2,0 m v obou osách polohy, což odpovídá typické přesnosti standardního GNSS bez RTK korekce. Kovariance v orientaci je záměrně zvolena prakticky neomezená ($\sigma = 999\,\text{rad}$), čímž se omezení v tomto směru efektivně vypíná – GPS poskytuje absolutní pozici, nikoli orientaci, a ta zůstává odhadována výhradně z odometrie a pozorování kuželů.

4.6.2 Datová asociace landmarků

Datová asociace odpovídá na otázku, zda pozorovaný kužel odpovídá již existujícímu landmarku v mapě, nebo zda se jedná o nový dosud nezmapovaný kužel. Spolehlivost asociace přímo ovlivňuje kvalitu výsledné mapy – chybné přiřazení vytváří falešné smyčky v grafu a deformuje optimalizovanou trajektorii.

Implementace využívá metodu nejbližšího souseda (nearest neighbour) s prahem 1,5 metru. Pozorovaný kužel je nejprve transformován ze souřadnic LiDARu do globálního souřadného systému pomocí aktuálně odhadované pozice vozidla. Poté se prohledají všechny existující landmarky stejné třídy (modrý, žlutý nebo oranžový) a vybere se ten s nejmenší euklidovskou vzdáleností. Pokud je

tato vzdálenost menší než práh, pozorování se přiřadí k existujícímu landmarku. V opačném případě se vytvoří nový landmark s unikátním identifikátorem.

Při transformaci souřadnic z rámce LiDARu do rámce vozidla je nutné zohlednit dvě korekce. LiDAR je umístěn 1,55 m před referenčním bodem vozidla, proto se k podélné souřadnici přičítá tento offset. Současně osa Y v simulátoru CARLA směřuje doprava, zatímco v knihovně GTSAM doleva, takže příčná souřadnice musí být negována.

Klasifikace kuželu (barva) se při asociaci využívá jako tvrdé omezení – modrý kužel nemůže být přiřazen ke žlutému landmarku a naopak. Toto omezení výrazně snižuje riziko chybné asociace, protože kužely jednotlivých barev lemují trať na protilehlých stranách a jejich vzájemná vzdálenost typicky přesahuje šířku trati.

Po uzavření smyčky přechází systém do režimu lokalizace, ve kterém je mapa zmrazena. V tomto režimu se nová pozorování přiřazují pouze k existujícím landmarkům. Pokud pozorovaný kužel nemá ve zmrazené mapě odpovídající protějšek pod prahem 1,5 metru, je jednoduše zahozen a do grafu se nepřidává žádný faktor.

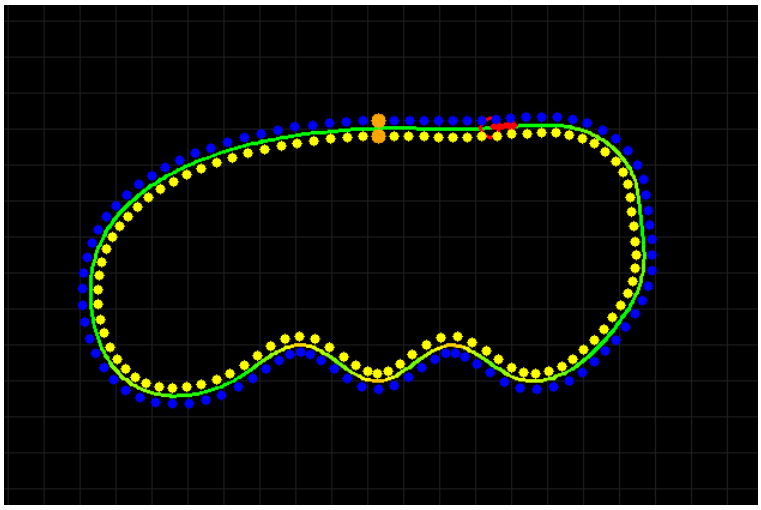
4.6.3 Detekce uzavření smyčky

Detekce uzavření smyčky (loop closure) je klíčovým krokem, který umožňuje eliminovat drift akumulovaný odometrií během celého okruhu. V kontextu Formula Student se úloha zjednodušuje díky tomu, že trať obsahuje právě dva oranžové kužely umístěné po stranách startovní čáry – žádné další oranžové kužely se na trati nevyskytují.

Registrace startovní čáry probíhá automaticky v prvních snímcích jízdy. Systém průběžně sleduje, zda mapa obsahuje alespoň dva landmarky oranžové třídy. Jakmile jsou nalezeny, jejich identifikátory se uzamknou jako referenční body startovní čáry a další oranžové kužely již nejsou vytvářeny.

Při návratu k startovní čáře – po ujetí dostatečné vzdálenosti definované prahem 40 metrů – se aktivuje speciální logika asociace pro oranžové kužely. Jakékoliv pozorování oranžového kuželu je v tomto stavu nuceně přiřazeno k nejbližšímu ze dvou uzamčených startovních kuželů bez ohledu na vzdálenostní práh. Toto nucené přiřazení je oprávněné právě díky unikátnosti oranžových kuželů na trati – pokud senzory po dokončení okruhu detekují oranžový kužel, musí se jednat o startovní čáru.

Uzavření smyčky je dokončeno ve chvíli, kdy jsou oba startovní kužely znovu pozorovány. Tato podmínka se akumuluje napříč snímky – není nutné vidět oba kužely současně v jednom záběru. Po splnění podmínky systém provede pět dodatečných iterací optimalizátoru iSAM2, které zajistí konvergenci grafu po vložení silně omezujících faktorů uzavření smyčky. Aktuální pozice vozidla i všechny landmarky se aktualizují na optimalizované hodnoty a systém přepne do režimu lokalizace, ve kterém se mapa již nemění a nové landmarky se nevytvářejí. Mapa po uzavření smyčky je na obrázku 4.11.



Obrázek 4.11: Graph SLAM mapa po uzavření smyčky

4.7 Plánování trajektorie

Jakmile SLAM dokončí mapu, máme k dispozici optimalizované pozice všech kuželů s barevnou klasifikací. Z nich je potřeba vygenerovat trajektorii pro závodní režim – spárovat protilehlé kužely, vypočítat středovou čáru a přiřadit jí rychlostní profil.

4.7.1 Párování kuželů

Párování kuželů (cone pairing) přiřazuje ke každému modrému kuželu na levé straně trati odpovídající žlutý kužel na pravé straně, čímž vznikají dvojice definující příčný průřez trati v daném místě. Implementovaný algoritmus využívá směrové párování (directional cone pairing), které zohledňuje nejen vzájemnou vzdálenost kuželů, ale také směr trati v daném úseku.

Algoritmus začíná u dvojice startovních oranžových kuželů, jejichž pozice jsou známé ze SLAM mapy. Z těchto dvou kuželů se odvodí počáteční směr trati jako vektor kolmý na spojnici obou kuželů, orientovaný ve směru jízdy. V každém kroku algoritmus hledá další pár kuželů tak, že prohledává okolí aktuální pozice ve směru trati. Z modrých kuželů se vybere ten, který leží nejblíže prodloužení aktuálního směru a zároveň se nachází na levé straně od středové čáry. Obdobně se z žlutých kuželů vybere nejbližší kužel na pravé straně. Nalezená dvojice se uloží a směr trati se aktualizuje podle vektoru spojujícího předchozí střed s novým středem.

Párování pokračuje, dokud se algoritmus nevrátí k počátečním oranžovým kuželům, nebo dokud nejsou vyčerpány všechny kužely. Směrové omezení zabraňuje zpětnému párování – algoritmus nemůže spárovat kužely, které leží za aktuální pozicí vzhledem ke směru jízdy. Maximální vzdálenost pro hledání dalšího páru je nastavena na 8 metrů, což odpovídá přibližně dvojnásobku typické

rozteče kuželů na trati FS. Pokud v tomto okruhu není nalezen vhodný kandidát na jedné straně, algoritmus přeskočí daný úsek a pokračuje dál.

Výstupem párování je uspořádaný seznam dvojic (modrý kužel, žlutý kužel), které definují koridor trati od startu zpět ke startu.

4.7.2 Generování středové čáry

Ze seznamu párů kuželů se vygeneruje středová čára trati. Pro každou dvojici se vypočítá středový bod jako aritmetický průměr souřadnic levého a pravého kužele. Výsledná sekvence středových bodů je poté proložena kubickým splajnem, který zajistí hladký průběh trajektorie bez ostrých zlomů.

Kubický splajn je po částech polynomiální křivka stupně tři, která prochází všemi zadanými body a má spojité první i druhé derivace. Díky tomu je křivost trajektorie spojitá, což je důležité pro plynulý průběh natočení volantu – nespojitost v křivosti by vyžadovala skokovou změnu natočení volantu, kterou aktuátor nedokáže realizovat.

Splajn je parametrizován délkou oblouku, přičemž vzorkování je zvoleno s krokem 0,5 metru. Výsledkem je hustě vzorkovaná středová čára s několika stovkami bodů, kde každý bod nese informaci o své pozici, směru tečny a křivosti. Pro uzavřenou trať je splajn periodický – poslední úsek plynule navazuje na první.

Středová čára sama o sobě není optimální závodní trajektorií, protože nevyužívá celou šířku trati pro řezání zatáček. Optimalizace závodní linie (racing line) ve smyslu minimalizace času na kolo je složitý optimalizační problém, který by vyžadoval řešení nelineárního programu s omezením na hranice trati. Implementace takové optimalizace přesahuje rozsah této práce – jako trajektorie pro závodní režim se proto používá středová čára doplněná o korekce z algoritmu cone repulsion, který je popsán v sekci o řízení.

4.7.3 Výpočet rychlostního profilu

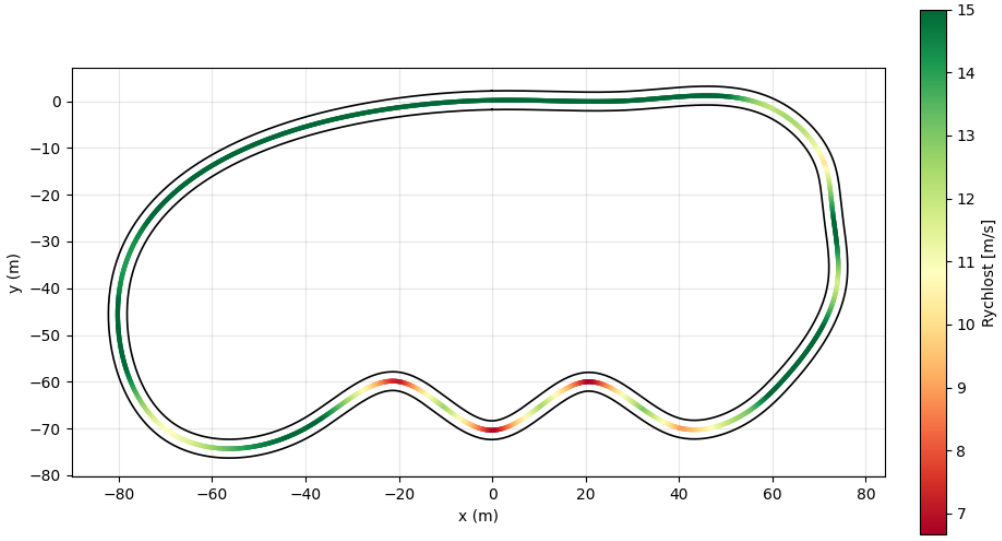
Pro každý bod středové čáry se vypočítá maximální bezpečná rychlost z lokální křivosti (rovnice 3.21). Poloměr zatáčky R_i se odhadne ze tří po sobě jdoucích bodů splajnu podle rovnice 3.20. Na rovných úsecích poloměr diverguje k vysokým hodnotám a rychlost omezuje jen $v_{top} = 15$ m/s (54 km/h).

Koeficient přilnavosti μ je nastaven na 1,0, což odpovídá suchému asfaltu a závodním pneumatikám. Maximální zrychlení a_{accel} je 5 m/s^2 a maximální brzdné zpomalení a_{brake} je 5 m/s^2. Tyto hodnoty jsou konzervativní odhady pro monopost Formula Student a byly zvoleny tak, aby vozidlo nepřekračovalo mez přilnavosti ani v nejostřejších zatáčkách.

Po výpočtu křivostních omezení se aplikuje dvouprůchodová forward-backward propagace podle rovnic 3.23 a 3.24. Backward průchod zajistí včasné brzdění před zatáčkami, forward průchod omezí zrychlení po výjezdu ze zatáček. Protože trať je uzavřená, oba průchody se provedou dvakrát –

jednou přes celý okruh a podruhé přes spojení konce a začátku – aby se rychlostní profil správně propagoval přes startovní čáru.

Výsledný rychlostní profil se uloží společně se středovou čárou a slouží jako vstup pro podélné řízení v závodním režimu. Příklad takového profilu je možné vidět na obrázku 4.12.



Obrázek 4.12: Rychlostní profil testovací trati B

4.8 Řízení vozidla

Závodní režim se aktivuje po dokončení SLAM mapy a výpočtu rychlostního profilu. Vozidlo už nejede naslepo podle aktuálně viditelných kuželů, ale sleduje předem vypočítanou trajektorii s proměnlivou rychlostí. Pro příčné řízení je použit Pure Pursuit s adaptivním lookaheadem nebo Stanley controller, pro podélné PID regulátor s cílovou rychlostí z rychlostního profilu. Jako doplněk slouží korekce cone repulsion.

4.8.1 PID pro podélné řízení

Podélné řízení v závodním režimu používá stejný PID regulátor jako reaktivní režim, avšak cílová rychlost již není konstantní. V každém snímku se z aktuální pozice vozidla na trajektorii odečte odpovídající hodnota rychlostního profilu a ta se použije jako setpoint regulátoru. Regulační odchylka je rozdíl mezi požadovanou a aktuální rychlostí vozidla. Kladný výstup regulátoru se mapuje na plyn, záporný na brzdu.

Konstanty regulátoru ($K_p = 0{,}5$, $K_i = 0{,}05$, $K_d = 0{,}01$) byly nastaveny experimentálně tak, aby vozidlo plynule zrychlovalo na rovinkách a včas brzdilo před zatáčkami bez výrazného překmitu.

Integrační složka je omezena anti-windup mechanismem, který zabraňuje akumulaci integrátoru při saturaci aktuátoru (plný plyn nebo plná brzda).

4.8.2 Pure Pursuit s adaptivním lookaheadem

V reaktivním režimu používá Pure Pursuit fixní lookahead vzdálenost 4 m, což je kompromis mezi přesností sledování v zatáčkách a stabilitou na rovinkách. V závodním režimu se lookahead vzdálenost l_d adaptivně mění v závislosti na aktuální rychlosti vozidla v [m/s]:

$$l_d = k_v \cdot v \tag{4.1}$$

kde $k_v = 0{,}7$ s je zesílení a v je aktuální rychlost v m/s. Výsledek se saturuje do rozsahu $\langle 4{,}0;\ 6{,}5 \rangle$ m. Při nižších rychlostech (pod 5,7 m/s) vozidlo sleduje bližší bod a přesněji kopíruje zatáčky, zatímco při vyšších rychlostech se dívá dále dopředu.

Lookahead bod se hledá na středové čáře jako bod ve vzdálenosti l_d od aktuální pozice vozidla měřené podél trajektorie. Natočení volantu se vypočítá podle rovnice 3.17 stejně jako v reaktivním režimu.

4.8.3 Stanley controller

Vedle Pure Pursuit je v závodním režimu implementován i Stanley controller podle rovnice 3.18. Stanley pracuje s referenčním bodem na přední nápravě a kombinuje korekci směrové odchylky (heading error) s korekcí příčné odchylky (crosstrack error). Zesílení regulátoru je nastaveno na $k = 0{,}4$.

V implementaci se příčná odchylka $e(t)$ počítá jako nejmenší vzdálenost středu přední nápravy od středové čáry se znaménkem – kladná hodnota znamená odchylku doleva, záporná doprava. Směrová odchylka $\psi(t)$ je rozdíl mezi aktuálním kurzem vozidla a směrem tečny k trajektorii v nejbližším bodě.

4.8.4 Cone repulsion

Cone repulsion je doplňková korekce trajektorie, která odpuzuje vozidlo od kuželů viditelných v aktuálním záběru senzorů. Účelem je zvýšit bezpečnostní odstup od hranic trati, zejména v situacích, kdy je středová čára ze SLAM mapy nepřesná nebo kdy vozidlo vlivem dynamiky sjede blíže k jedné straně.

Pro každý kužel detekovaný v aktuálním snímku se vypočítá odpudivá síla směřující od kužele k vozidlu. Velikost síly lineárně klesá se vzdáleností:

$$F_{repulse} = k_{rep} \cdot \left(1 - \frac{d}{d_{max}}\right) \tag{4.2}$$

kde d je vzdálenost kužele od referenčního bodu vozidla, $d_{max} = 0{,}5$ m je maximální vzdálenost působení odpudivé síly a $k_{rep} = 0{,}3$ je zesílení. Pro kužely vzdálenější než d_{max} je síla nulová. Výsledný laterální posun cílového bodu je omezen na $\pm 1{,}5$ m, aby korekce nenarušovala sledování trajektorie na rovných úsecích, kde jsou kužely dostatečně daleko.

4.9 Celkové zhodnocení

4.9.1 Testovací tratě

Pro vyhodnocení byly nástrojem pro tvorbu tratí vygenerovány tři tratě s odlišným charakterem. Trať A je rychlá s dlouhými rovinkami a mírnými oblouky (minimální poloměr 20,2 m), trať B je smíšená se středně ostrými zatáčkami a trať C je technická s krátkými rovinkami, ostrými zatáčkami a slalomem.Parametry tratí shrnuje tabulka 4.7 a jejich profily lze vidět na obrázku 4.13.

Parametr	**Trať A**	**Trať B**	**Trať C**
Délka [m]	464,4	404,8	631,4
Počet kuželů	228	192	306
Min. poloměr zatáčky [m]	20,2	7,2	6,7
Charakter	rychlá	smíšená	technická

Tabulka 4.7: Parametry testovacích tratí

4.9.2 Porovnání režimů řízení

Porovnávány byly tři konfigurace příčného řízení:

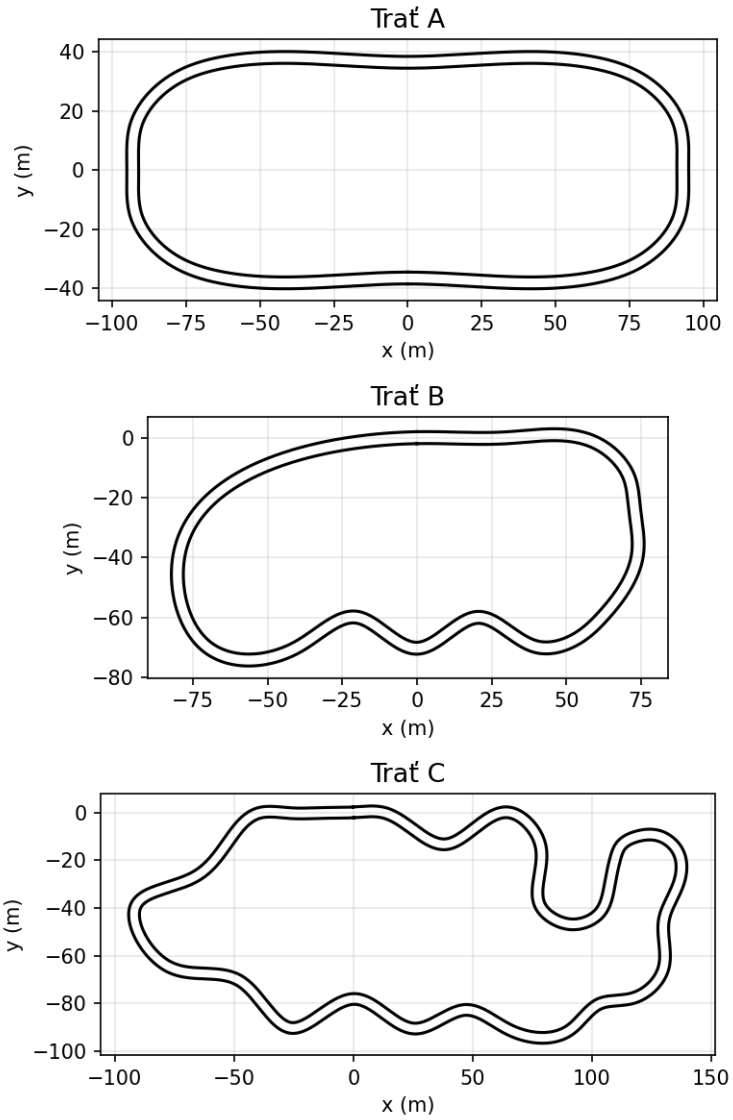
1. **Reaktivní** – Pure Pursuit s fixním lookaheadem 4 m, konstantní cílová rychlost 5 m/s, bez SLAM mapy. Odpovídá prvnímu průjezdu v disciplíně DV Autocross.

2. **Závodní (Pure Pursuit)** – Pure Pursuit s adaptivním lookaheadem, proměnlivý rychlostní profil ze SLAM mapy, cone repulsion.

3. **Závodní (Stanley)** – Stanley controller s $k = 0{,}4$, ostatní parametry shodné s konfigurací 2.

Pro každou konfiguraci bylo na každé trati odjeto 10 okruhů. U závodních režimů byl první okruh (mapovací) vyloučen z měření. Zaznamenávány byly celkový čas za 10 kol, čas nejlepšího okruhu, průměrná a maximální příčná odchylka od středové čáry a počet sražených kuželů. Výsledky shrnuje tabulka 4.8.

Reaktivní režim projíždí všechny tři tratě bezpečně a bez sražení kuželů, avšak s výrazně delšími časy. Konstantní rychlost 5 m/s a trajektorie generovaná pouze z aktuálně viditelných kuželů nedovoluje řezat zatáčky ani zrychlovat na rovinkách.



Obrázek 4.13: Profily tratí použitých pro testování

Závodní režim s Pure Pursuit a adaptivním lookaheadem dosahuje na všech tratích podstatně kratších časů. Na trati C (technická, přibližně 630 m) činí nejlepší čas okruhu 75,5 s oproti 126,7 s v reaktivním režimu, což odpovídá zlepšení o 40,4 %. Maximální rychlost na rovinkách dosahuje přibližně 15 m/s (54 km/h), v nejostřejších zatáčkách vozidlo zpomaluje na přibližně 7 m/s (25 km/h), což odpovídá teoretickému limitu boční přilnavosti pro poloměr 6,7 m při $\mu = 1{,}0$. Na trati B srazil závodní režim s Pure Pursuit celkem 9 kuželů. Příčinou je nízký minimální poloměr zatáčky (7,2 m) v kombinaci s vyšší rychlostí – vozidlo nestihne včas zpomalit a při výjezdu z ostrých zatáček překročí hranice trati. I po započtení penalizace 2 s za každý sražený kužel (celkem 18 s) by Pure Pursuit zůstal výrazně rychlejší oproti reaktivnímu režimu.

Trať	Metrika	Reaktivní	Závodní (PP)	Závodní (Stanley)
A	Celkový čas – 10 kol [s]	933,7	410,8	428,6
	Nejlepší kolo [s]	93,3	35,0	36,6
	Sražené kužely	0	0	0
B	Celkový čas – 10 kol [s]	747,3	423,0	445,2
	Nejlepší kolo [s]	74,8	38,4	40,7
	Sražené kužely	0	9	0
C	Celkový čas – 10 kol [s]	1266,7	807,6	874,7
	Nejlepší kolo [s]	126,7	75,0	82,6
	Sražené kužely	0	0	0

Tabulka 4.8: Porovnání tří režimů řízení na třech testovacích tratích

Stanley controller dosahuje na všech testovaných tratích o 4–9 % pomalejších časů než Pure Pursuit. Rozdíl je nejmenší na rychlé trati A (nejlepší kolo 36,6 s vs. 35,0 s, tj. +4,6 %) a největší na technické trati C (82,6 s vs. 75,0 s, tj. +10 %). Příčinou je odlišný princip obou regulátorů: Pure Pursuit s adaptivním lookaheadem se dívá dále dopředu a může proto začít natáčet volant před vstupem do zatáčky, čímž efektivně řeže její apex. Stanley naproti tomu pracuje s odchylkou na přední nápravě a začíná korekci až ve chvíli, kdy se vozidlo od trajektorie odchýlí, což vede k přesnějšímu, ale konzervativnějšímu průjezdu.

Zrychlení závodního režimu má dvě příčiny. Zaprvé, proměnlivý rychlostní profil umožňuje plné využití rovinek. Zadruhé, trajektorie odvozená ze SLAM mapy je hladká a pokrývá celý okruh, zatímco reaktivní režim plánuje pouze ze 3 až 5 párů kuželů viditelných v daném okamžiku.

Klíčový rozdíl se projevuje v bezpečnosti na úzké trati B. Zatímco Pure Pursuit zde srazil 9 kuželů kvůli vybočení na výjezdu z ostrých zatáček, Stanley projel celých 10 okruhů bez jediné kolize. Pro reálné nasazení v disciplíně Trackdrive, kde se penalizace za sražený kužel sčítají v rámci celkového výsledku, by tedy Stanley představoval bezpečnější volbu. Pure Pursuit je naopak vhodnější pro tratě s mírnějšími poloměry, kde lze plně využít jeho schopnost dopředného plánování trajektorie.

4.9.3 Výkon pipeline

Výpočetní latence byla měřena jako doba zpracování jednoho snímku od příjmu senzorických dat po odeslání řídicího příkazu. Měření probíhalo v závodním režimu po dobu 10 okruhů na trati C. Tabulka 4.9 uvádí mediánové a maximální časy jednotlivých fází.

Mediánová latence 17 ms odpovídá frekvenci 58 Hz, což s rezervou 33 ms převyšuje frekvenci senzorů 20 Hz. Maximální naměřená latence 27 ms nastává při souběhu periodické reoptimalizace SLAM grafu a zpracování snímku s vysokým počtem detekcí (více než 15 kuželů). Ani v tomto případě systém neztratí žádný snímek – zpracování dokončí před příchodem dalšího.

Fáze pipeline	**Medián [ms]**	**Max [ms]**
Detekční pipeline	14	24
SLAM backend (odometrie + asociace + iSAM2)	3	17
Řízení (Pure Pursuit / Stanley + PID + cone repulsion)	< 1	1
Celkem	**17**	**27**

Tabulka 4.9: Latence jednotlivých fází pipeline v závodním režimu

4.9.4 Porovnání s existujícími přístupy

Architektura systému vychází z přístupu týmu KA-RaceIng z Karlsruhe Institute of Technology [30], který s podobnou pipeline vyhrál FS Germany, FS Czech Republic i FS Hungary v roce 2021. Oba systémy sdílejí řadu společných návrhových rozhodnutí – grafový SLAM backend, LiDAR jako primární senzor pro detekci kuželů s volitelnou kamerovou klasifikací barvy a plánování trajektorie přes středové body párů kuželů proložené kubickým splajnem. KA-RaceIng používá pro optimalizaci grafu knihovnu g2o, implementovaný systém knihovnu GTSAM s inkrementálním optimalizátorem iSAM2.

Přes uvedené shody se systémy liší v několika aspektech. V datové asociaci landmarků využívá KA-RaceIng Mahalanobisovu vzdálenost, která zohledňuje nejistotu odhadu pozice landmarku a snižuje tak riziko chybného přiřazení na větší vzdálenosti. Implementovaný systém používá euklidovskou vzdálenost s fixním prahem 1,5 m, což je jednodušší přístup, který se na rozměrech typické FS trati (řádově stovky metrů s kužely vzdálenými přibližně 5 m od sebe) osvědčil bez výrazného zhoršení kvality mapy.

V plánování trajektorie KA-RaceIng doplňuje středovou čáru o optimalizaci minimální křivosti, která podle autorů zkracuje čas na kolo o více než 10 %. Implementovaný systém tuto optimalizaci neprovádí a jako trajektorii používá přímo středovou čáru doplněnou o korekce z algoritmu cone repulsion. Nasazení optimalizace závodní linie představuje hlavní prostor pro další zkrácení času na kolo.

Z hlediska řídicích algoritmů nasazuje KA-RaceIng MPC pro příčné řízení a PI regulátor pro podélné, čímž dosahuje příčných zrychlení přes 1,6 g i na úzkých tratích. Implementovaný systém volí geometrické regulátory Pure Pursuit s adaptivním lookaheadem a Stanley, které jsou jednodušší na implementaci a ladění. Tato volba odpovídá zaměření práce na demonstraci celé autonomní pipeline v simulátoru, nikoli na maximalizaci dynamického výkonu vozidla.

Odlišný přístup představuje systém týmu AMZ Driverless z ETH Zürich [13]. AMZ provozuje dvě plně nezávislé percepční pipeline – LiDAR s Euclidean clustering a kamerovou pipeline s detekcí kuželů sítí YOLOv3 – a pro mapování používá FastSLAM 2.0 založený na částicovém filtru. Po uzavření smyčky přepíná na nelineární MPC, které přímo minimalizuje čas na kolo s ohledem na limity pneumatik a hranice trati, a dosahuje bočních zrychlení až 1,5 g. Výpočetní nároky a složitost

tohoto systému jsou však výrazně vyšší. Pro první průjezd tratí nicméně i AMZ používá pure pursuit po odhadnuté středové čáře z Delaunay triangulace, což je obdobný přístup jako reaktivní režim implementovaného systému.

Přímé kvantitativní srovnání časů na kolo s výše zmíněnými týmy není možné, neboť oba testovaly na reálných vozidlech a tratích s odlišnými rozměry, povrchem i podmínkami. Implementovaný systém je však postaven na stejných principech jako systém vítězného týmu soutěže a demonstruje funkční autonomní pipeline, která v simulátoru spolehlivě projede trať bez sražení kužele v obou režimech jízdy. Na rozdíl od zmíněných systémů, které vyžadují prostředí ROS a jsou implementovány v C++, celá pipeline běží v Pythonu.

4.9.5 Limity implementace

Při interpretaci výsledků je potřeba mít na paměti několik omezení. Veškeré experimenty probíhaly v simulátoru CARLA, který zjednodušuje řadu aspektů reálného prostředí. Senzorická data neobsahují artefakty jako znečištění objektivu, vibrace šasi přenášené do LiDARu ani proměnlivé atmosférické podmínky. Přenos natrénovaných modelů a vyladěných parametrů na reálné vozidlo (sim-to-real transfer) by vyžadoval dodatečnou kalibraci a pravděpodobně i dotrénování klasifikátoru ConeCNN na reálných datech.

Dynamický model vozidla v CARLA byl nakonfigurován tak, aby se kvalitativně blížil chování monopostu Formula Student, avšak nebyl kalibrován na konkrétní reálné vozidlo. Parametry rychlostního profilu (μ, a_{accel}, a_{brake}) jsou konzervativní odhady a na reálném voze by vyžadovaly úpravu na základě telemetrických dat.

Datová asociace landmarků používá metodu nejbližšího souseda s fixním prahem, což je nejjednodušší přístup. Na tratích s úzkými průjezdy, kde jsou protilehlé kužely blízko sebe, by mohl být robustnější pravděpodobnostní přístup (například joint compatibility branch and bound).

Středová čára použitá jako trajektorie nevyužívá celou šířku trati pro řezání zatáček. Optimalizace závodní linie (racing line) ve smyslu minimalizace času na kolo a nasazení prediktivního regulátoru typu MPC představuje hlavní prostor pro další zlepšení výkonu systému.

Kapitola 5

Závěr

Cílem této diplomové práce bylo navrhnout a implementovat systém pro autonomní řízení monopostu Formula Student v simulátoru CARLA a ověřit jeho funkčnost na úlohách odpovídajících disciplínám DV Cupu.

V rámci práce byl popsán simulátor CARLA, jeho architektura a důvody pro jeho volbu oproti alternativám, jako jsou FSDS, IPG CarMaker a dSPACE AURELION. Pro potřeby testování bylo připraveno simulační prostředí zahrnující vlastní 3D modely kuželů odpovídající specifikaci Formula Student, importovaný model monopostu MFT02 a nástroj pro generování tratí s grafickým rozhraním.

Detailně byly popsány metody použité v jednotlivých fázích autonomní pipeline: konvoluční neuronové sítě pro klasifikaci barvy kuželů, RANSAC pro segmentaci roviny vozovky, DBSCAN pro shlukování bodů z LiDARu, projekční matice pro fúzi dat z LiDARu a kamery, Graph SLAM pro simultánní lokalizaci a mapování a geometrické řídicí algoritmy Pure Pursuit a Stanley.

Implementovaný systém pokrývá celý řetězec od senzorických dat po řídicí příkazy. Percepční pipeline využívá LiDAR Ouster OS2-128 pro detekci kuželů a kameru s rozlišením 1280×720 pro klasifikaci barvy pomocí lehké sítě ConeCNN s přesností 99,85%. SLAM backend založený na knihovně GTSAM a algoritmu iSAM2 buduje mapu trati s detekcí uzavření smyčky přes oranžové kužely startovní čáry. Ze SLAM mapy se generuje středová čára trati proložená kubickým splajnem a rychlostní profil zohledňující křivost trati a dynamická omezení vozidla.

Systém byl ověřen ve dvou režimech řízení. Reaktivní režim, určený pro první průjezd neznámou tratí, využívá středové body z aktuálně viditelných kuželů a Pure Pursuit s fixním lookaheadem při konstantní rychlosti 5 m/s. Závodní režim sleduje předem vypočítanou trajektorii pomocí kombinace Pure Pursuit s adaptivním lookaheadem a Stanley controlleru s proměnlivou cílovou rychlostí. Závodní režim dosáhl na technické testovací trati o 40 % rychlejšího průměrného času průjezdu oproti reaktivnímu režimu při zachování bezkoliznímu průjezdu.

Celková latence pipeline činí v mediánu 17 ms, což umožňuje zpracování senzorických dat na frekvenci 58 Hz s dostatečnou rezervou vůči frekvenci senzorů 20 Hz.

Mezi hlavní přínosy práce patří funkční demonstrace celé autonomní pipeline v otevřeném simulátoru, porovnání tří konfigurací LiDARu z hlediska schopnosti detekce kuželů, návrh lehké klasifikační sítě ConeCNN, která dosahuje vysoké přesnosti při minimální výpočetní náročnosti, a implementace dvou režimů řízení odpovídajících reálným podmínkám disciplín DV Autocross a Trackdrive.

Při zpracování praktické části této práce byl pro pomoc s tvorbou a laděním zdrojového kódu využit nástroj generativní umělé inteligence Claude od společnosti Anthropic [44]. Konkrétně byly využity modely Claude Sonnet 4.5, Sonnet 4.6 a Claude Opus 4.5, 4.6 a 4.7. Veškerý vygenerovaný kód byl řádně zkontrolován, otestován a upraven tak, aby odpovídal požadavkům zadání.

Literatura

1. *About Formula Student Germany*. 2026. Dostupné také z: https://www.formulastudent.de/fsg/about.
2. *Formula Student Czech Republic Handbook 2026 v1.0*. 2025. Dostupné také z: https://fsczech.cz/wp-content/uploads/2025/12/Formula-Student-Czech-Republic-Handbook-2026-v1.0.pdf.
3. *History of Formula SAE*. 2015. Dostupné také z: https://www.fsaeonline.com/page.aspx?pageid=c4c5195a-60c0-46aa-acbf-2958ef545b72.
4. *History of Formula Student*. 2022. Dostupné také z: https://www.imeche.org/events/formula-student/about-formula-student/history-of-formula-student.
5. *Formula Student World*. 2026. Dostupné také z: https://fs-world.org/ranking.
6. *Formula Student Germany Event Handbook 2026 v1.1*. 2026. Dostupné také z: https://www.formulastudent.de/fileadmin/user_upload/all/2026/rules/FSG26_Event_Handbook_v1.1.pdf.
7. *Formula TU Ostrava*. 2026. Dostupné také z: https://formulaostrava.cz.
8. DOSOVITSKIY, Alexey; ROS, German; CODEVILLA, Felipe; LOPEZ, Antonio; KOLTUN, Vladlen. CARLA: An Open Urban Driving Simulator. In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, s. 1–16.
9. FS-DRIVERLESS. *Formula-Student-Driverless-Simulator*. 2026. Dostupné také z: https://github.com/FS-Driverless/Formula-Student-Driverless-Simulator.
10. SHAH, Shital; DEY, Debadeepta; LOVETT, Chris; KAPOOR, Ashish. *AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles*. 2017. Dostupné z arXiv: 1705.05065 [cs.RO].
11. *CarMaker*. 2026. Dostupné také z: https://www.ipg-automotive.com/solutions/product-portfolio/carmaker.
12. *DSPACE AURELION*. 2026. Dostupné také z: https://www.dspace.com/en/pub/home/products/sw/experimentandvisualization/aurelion_sensor-realistic_sim.cfm.

13. KABZAN, Juraj; IGLESIA VALLS, Miguel de la; REIJGWART, Victor; HENDRIKX, Hubertus Franciscus Cornelis; EHMKE, Claas; PRAJAPAT, Manish; BÜHLER, Andreas; GOSALA, Nikhil; GUPTA, Mehak; SIVANESAN, Ramya; DHALL, Ankit; CHISARI, Eugenio; KARNCHANACHARI, Napat; BRITS, Sonja; DANGEL, Manuel; SA, Inkyu; DUBÉ, Renaud; GAWEL, Abel; PFEIFFER, Mark; LINIGER, Alexander; LYGEROS, John; SIEGWART, Roland. *AMZ Driverless: The Full Autonomous Racing System*. 2019. Dostupné z arXiv: 1905.05150 [cs.RO].

14. IGNATIOUS, Henry Alexander; SAYED, Hesham-El-; KHAN, Manzoor. An overview of sensors in Autonomous Vehicles. *Procedia Computer Science*. 2022, roč. 198, s. 736–741. ISSN 1877-0509. Dostupné z DOI: https://doi.org/10.1016/j.procs.2021.12.315. 12th International Conference on Emerging Ubiquitous Systems and Pervasive Networks / 11th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare.

15. LI, You; IBANEZ-GUZMAN, Javier. Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems. *IEEE Signal Processing Magazine*. 2020-07, roč. 37, č. 4, s. 50–61. ISSN 1558-0792. Dostupné z DOI: 10.1109/msp.2020.2973615.

16. LI, You; IBANEZ-GUZMAN, Javier. Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems. *IEEE Signal Processing Magazine*. 2020, roč. 37, č. 4, s. 50–61. Dostupné z DOI: 10.1109/MSP.2020.2973615.

17. LECUN, Yann; HAFFNER, Patrick; RACHMAD, Yoesoep; BOTTOU, Leon. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*. 1998-12, roč. 86, s. 2278–2324. Dostupné z DOI: 10.1109/5.726791.

18. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

19. AGARAP, Abien Fred. *Deep Learning using Rectified Linear Units (ReLU)*. 2019. Dostupné z arXiv: 1803.08375 [cs.NE].

20. KINGMA, Diederik P.; BA, Jimmy. *Adam: A Method for Stochastic Optimization*. 2017. Dostupné z arXiv: 1412.6980 [cs.LG].

21. FISCHLER, Martin A.; BOLLES, Robert C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*. 1981-06, roč. 24, č. 6, s. 381–395. ISSN 0001-0782. Dostupné z DOI: 10.1145/358669.358692.

22. ESTER, Martin; KRIEGEL, Hans-Peter; SANDER, Jörg; XU, Xiaowei. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. Portland, Oregon: AAAI Press, 1996, s. 226–231. KDD'96.

23. HARTLEY, Richard; ZISSERMAN, Andrew. *Multiple View Geometry in Computer Vision*. 2. vyd. Cambridge University Press, 2004.

24. KALMAN, R. E. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*. 1960-03, roč. 82, č. 1, s. 35–45. ISSN 0021-9223. Dostupné z DOI: 10.1115/1.3662552.

25. RIBEIRO, Maria; RIBEIRO, Isabel. *Kalman and Extended Kalman Filters: Concept, Derivation and Properties*. 2004-04.

26. DISSANAYAKE, G; NEWMAN, P; CLARK, Steve; DURRANT-WHYTE, H; CSORBA, M. A solution to the simultaneous localisation and map building (SLAM) problem. *IEEE Transactions of Robotics and Automation*. 2001-01.

27. MONTEMERLO, Michael; THRUN, Sebastian; KOLLER, Daphne; WEGBREIT, Ben. FastSLAM: a factored solution to the simultaneous localization and mapping problem. In: *Eighteenth National Conference on Artificial Intelligence*. Edmonton, Alberta, Canada: American Association for Artificial Intelligence, 2002, s. 593–598. ISBN 0262511290.

28. MONTEMERLO, Michael; THRUN, Sebastian; KOLLER, Daphne; WEGBREIT, Ben. FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. *Proc. IJCAI Int. Joint Conf. Artif. Intell.* 2003-06.

29. THRUN, Sebastian; MONTEMERLO, Michael. The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. *The International Journal of Robotics Research*. 2006, roč. 25, č. 5-6, s. 403–429. Dostupné z DOI: 10.1177/0278364906065387.

30. ALVAREZ, Andres; DENNER, Nico; FENG, Zhe; FISCHER, David; GAO, Yang; HARSCH, Lukas; HERZ, Sebastian; LARGE, Nick; NGUYEN, Bach; ROSERO, Carlos; SCHAEFER, Simon; TERLETSKIY, Alexander; WAHL, Luca; WANG, Shaoxiang; YAKUPOVA, Jonona; YU, Haocen. *The Software Stack That Won the Formula Student Driverless Competition*. 2022-10. Dostupné z DOI: 10.48550/arXiv.2210.10933.

31. ASTRÖM, K.J.; HÄGGLUND, T. *PID Controllers: Theory, Design, and Tuning*. Wiley, 1995. International Society of Automation. ISBN 9781556175169. Dostupné také z: https://books.google.cz/books?id=FsyhngEACAAJ.

32. COULTER, Craig. Implementation of the Pure Pursuit Path Tracking Algorithm. In: 1992. Dostupné také z: https://api.semanticscholar.org/CorpusID:62550799.

33. HOFFMANN, Gabriel M.; TOMLIN, Claire J.; MONTEMERLO, Michael; THRUN, Sebastian. Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing. In: *2007 American Control Conference*. 2007, s. 2296–2301. Dostupné z DOI: 10.1109/ACC.2007.4282788.

34. BORRELLI, Francesco; BEMPORAD, Alberto; MORARI, Manfred. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

35. HEILMEIER, Alexander; WISCHNEWSKI, Alexander; HERMANSDORFER, Leonhard; BETZ, Johannes; LIENKAMP, Markus; LOHMANN, Boris. Minimum curvature trajectory planning and control for an autonomous race car. *Vehicle System Dynamics*. 2020, roč. 58, č. 10, s. 1497–1527. Dostupné z DOI: 10.1080/00423114.2019.1631455.

36. MILLIKEN, William F.; MILLIKEN, Douglas L. *Race car vehicle dynamics*. Race car vehicle dynamics. Warrendale, Pa: Society of Automotive Engineers, 1995. ISBN 1560915269.

37. ROSADO, Diego. *MAD Formula Team MFT02 V1.42 Final* [online]. Overtake.gg, 2023 [cit. 2026-04-02]. Dostupné z: https://www.overtake.gg/downloads/mad-formula-team-mft02.58653/. Licencováno pod CC BY-NC-ND 4.0.

38. *Velodyne VLP-16 User Manual* [online]. 2022. [cit. 2026-04-08]. Dostupné z: https://data.ouster.io/downloads/velodyne/user-manual/vlp-16-user-manual-revf.pdf.

39. *Ouster OS2-128 Datasheet*. 2024. Dostupné také z: https://data.ouster.io/downloads/datasheets/datasheet-rev7-v2p5-os2.pdf.

40. *Valeo Mobility Kit SCALA Gen 2 Datasheet*. 2026. Dostupné také z: https://levelfivesupplies.com/wp-content/uploads/2021/09/Valeo-Mobility-Kit-SCALA-3D-laser-scanner-gen-2-data-sheet.pdf.

41. DELLAERT, Frank; CONTRIBUTORS, GTSAM. *borglab/gtsam*. Georgia Tech Borg Lab, 2022-05. 4.2a8. Dostupné z DOI: 10.5281/zenodo.5794541.

42. DELLAERT, Frank; KAESS, Michael. *Factor Graphs for Robot Perception*. Foundations a Trends in Robotics, Vol. 6, 2017. Dostupné také z: http://www.cs.cmu.edu/~kaess/pub/Dellaert17fnt.pdf.

43. KAESS, Michael; JOHANNSSON, Hordur; ROBERTS, Richard; ILA, Viorela; LEONARD, John; DELLAERT, Frank. iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, s. 3281–3288. Dostupné z DOI: 10.1109/ICRA.2011.5979641.

44. ANTHROPIC. *Claude [generativní AI]* [online]. 2026. [cit. 2026-04-24]. Dostupné z: https://claude.ai/.

Příloha A

Archiv

Příloha v IS EDISON.

A.1 Obsah archivu

- modely kuželů a monopostu soutěže Formula Student
- dataset, skript pro trénování a váhy modelu ConeCNN
- konzolová aplikace pro tvorbu vlastních tratí
- aplikace pro nastavení a monitorování experimentů
- videoukázka fungování autonomního řízení
- výsledky jednotlivých experimentů
- soubor README.md s návodem na instalaci a podrobným popisem souborů v příloze