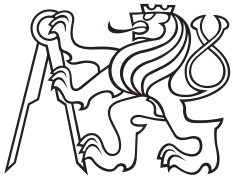
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra měření

Digitální dvojče autonomního vozidla

Jan Randa

Vedoucí: Ing. Michal Sojka, Ph.D.
Obor: Otevřená informatika
Studijní program: Internet věcí
Květen 2024

Poděkování

Děkuji Ing. Michalovi Sojkovi, Ph.D. za vedení práce a Ing. Adamovi Kollarčíkovi za nápad a formulaci Kalmanova filtru.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu. Dále prohlašuji, že jsem při psaní práce použil umělou inteligenci, konkrétně pro lepší formulaci vět a upravování obrázků.

V Praze, 24. května 2024

Abstrakt

Digitální dvojče vozidla je technologie, která může usnadnit vývoj autonomních vozidel. Cílem práce bylo vytvořit digitální dvojče pro Porsche Cayenne, na kterém se vyvíjí Autonomous Lane-Keeping System (ALKS) v rámci projektu CERTICAR. Byl zvolen přístup použití simulátoru CARLA společně s Kalmanovým filtrem. Ukázalo se, že tento přístup je možný a jeho implementace byla úspěšná. Výsledné digitální dvojče z této práce bude používáno při dalších fázích projektu CERTICAR.

Klíčová slova: digitální dvojče, automotive, autonomní vozidlo, Kalmanův filtr, ROS, CARLA

Vedoucí: Ing. Michal Sojka, Ph.D.
CIIRC
Jugoslávských partyzánů 1580/3
160 00 Praha 6, Dejvice

Abstract

A vehicle's digital twin is a technology that facilitates the development of autonomous vehicles. The goal of the work was to create a digital twin of the Porsche Cayenne, for which the Autonomous Lane-Keeping System (ALKS) is being developed as part of the CERTICAR project. The approach of using the CARLA simulator together with the Kalman filter was chosen. This approach has been shown to be possible and its implementation has been successful. The resulting digital twin from this work will be used in the next phases of the CERTICAR project.

Keywords: digital twin, automotive, autonomous vehicle, Kalman filter, ROS, CARLA

Title translation: Digital twin of an autonomous car

Obsah

Obrázky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Randa** Jméno: **Jan** Osobní číslo: **507662**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Otevřená informatika**

Specializace: **Internet věcí**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Digitální dvojče autonomního vozidla

Název bakalářské práce anglicky:

Digital twin of an autonomous car

Pokyny pro vypracování:

1. Seznamte se se simulátorem CARLA a s projektem vyvíjejícím systém Automated Lane-Keeping System (ALKS) implementovaném ve frameworku ROS 2. Projekt v současné době funguje na reálném vozidle a je možné jej testovat pomocí simulátoru CARLA.
2. Navrhněte a implementujte benchmarky pro měření rychlosti, přesnosti a latence simulace a optimalizujte nastavení simulátoru či simulačních scénářů, aby vše mohlo běžet v reálném čase na vzdáleném serveru.
3. Rozšiřte implementaci ALKS pro reálné vozidlo o komunikaci s digitálním dvojčetem v simulátoru. Zajistěte, aby stavy digitálního dvojčete konvergovaly ke stavům reálného auta, ale zároveň aby bylo možné detekovat větší odchylky mezi simulací a reálným autem a to v reálném čase. Cílem je, aby drobné odchylky mezi simulací a realitou byly ignorovány, ale aby bylo možné detekovat větší chyby např. v řídicím softwaru ALKS, které se neprojevují v samotné simulaci, ale jen v reálném světě.
4. Implementujte následující funkcionalitu zahrnující komunikaci z digitálního dvojčete do řídicího systému ALKS: a) Zastavení vozidla při detekci velké odchylky, b) řízení vozidla na základě vodorovného značení a okolních vozidel ze simulátoru digitálního dvojčete. Prozkoumejte i možnost použít digitální dvojče pro simulaci možného budoucího vývoje aktuální situace.
5. Otestujte správnou funkčnost vyvinutých komponent jednak při použití dvou simulátorů CARLA současně a jednak na datech z provozu
reálného vozidla. Pokud to bude možné otestujte vše i s reálným vozidlem v reálném čase.
6. Výsledky důkladně zdokumentujte.

Seznam doporučené literatury:

[1] S. Y. Gelbal, B. Aksun-Guvenc, and L. Guvenc, “Vehicle in Virtual Environment (VVE) Method.” arXiv, Dec. 21, 2022. doi: 10.48550/arXiv.2212.11469.
[2] C. Schwarz and Z. Wang, “The Role of Digital Twins in Connected and Automated Vehicles,” IEEE Intelligent Transportation Systems
Magazine, vol. 14, no. 6, pp. 41–51, Nov. 2022, doi: 10.1109/MITS.2021.3129524.
[3] CARLA documentation: https://carla.readthedocs.io/en/0.9.15/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Michal Sojka, Ph.D. vestavěné systémy CIIRC

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **17.01.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce:
do konce letního semestru 2024/2025

Ing. Michal Sojka, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. | Datum převzetí zadání

Podpis studenta

Kapitola 1
Úvod

V oblasti vývoje autonomních vozidel se setkáváme s problémy z velké škály oblastí, například elektrotechnika, robotika, kyberbezpečnost, umělá inteligence a strojové učení. Vzhledem k tomu, že tyto technologie jsou tak rozmanité a složité, je nezbytné najít způsob, jak co nejvíce tento proces zjednodušit.

Jedním z řešení, jak zefektivnit vývoj autonomních vozidel, je využití digitálního dvojčete. Digitální dvojče je virtuální reprezentace reálného vozidla, která simuluje jeho chování v reálném čase a zároveň může ovlivňovat jeho řízení. Jeden z problémů, který digitální dvojče řeší je situace, kdy vývoj probíhá v simulátoru, ale testování na reálném vozidle a tím vzniká problém, protože se reálné vozidlo chová jinak, než simulace. S dvojčetem může výzkumný tým pozorovat odchylky při testování a díky okamžité dostupnosti dat ze simulace (dvojčete) bude jednodušší chyby opravit hned na místě.

1.1 Cíle práce

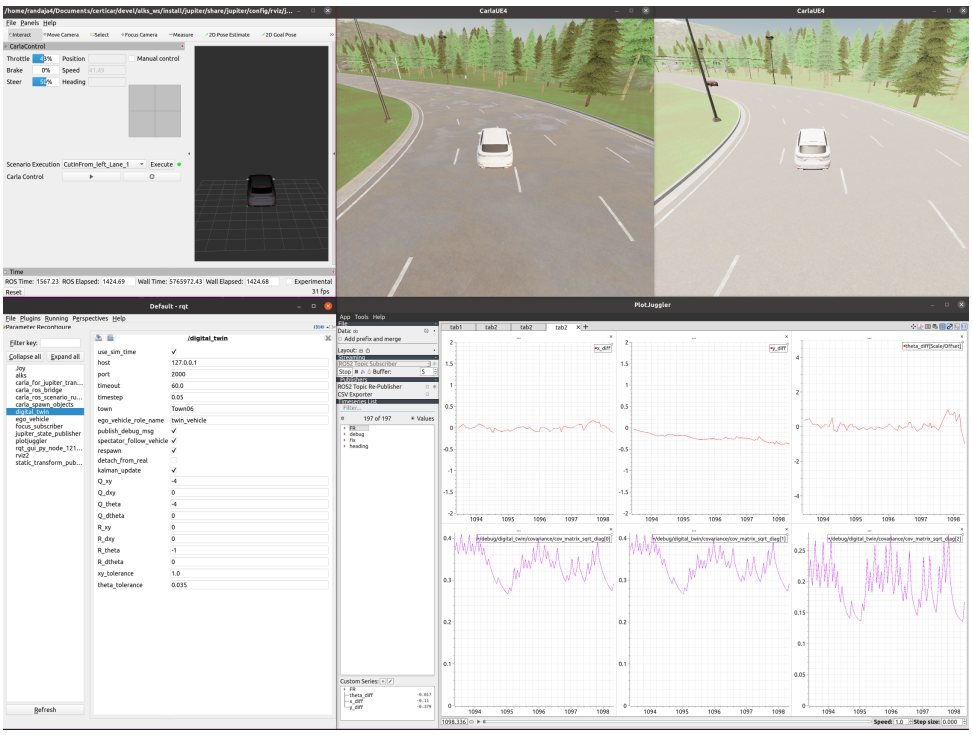
Cílem práce bylo vytvořit digitální dvojče pro Porsche Cayenne, na kterém se vyvíjí Autonomous Lane-Keeping System (ALKS). Požadované vlastnosti dvojčete jsou:

1. zastavení vozidla při detekci velké odchylky,
2. řízení vozidla na základě vodorovného značení a okolních vozidel ze simulátoru digitálního dvojčete.
3. možnost použít jak s reálným vozidlem, tak s vozidlem ze simulátoru.

Ukázka výsledného řešení je na obrázku 1.1.

1.2 Struktura práce

Za úvodní částí následuje kapitola Použité technologie, která uvádí a krátce popisuje využité technologie. Pak kapitola Analýza a návrh, která rozšiřuje požadavky na digitální dvojče a uvádí návrh, podle které je dvojče implementováno. V kapitole Implementace je pohled do kódu digitálního dvojčete



Obrázek 1.1: Vpravo nahoře jsou okna dvou simulátorů CARLA (vlevo dvojče, vpravo simulátor reálného vozidla), vlevo nahoře vizualizační nástroj RViz, vlevo dole program rqt pro nastavení parametrů a vpravo dole vizualizační nástroj Plotjuggler. Plotjuggler zobrazuje odchylky stavových veličin (x, y, θ) červenou barvou, pod nimi pak fialovou odpovídající kovariance.

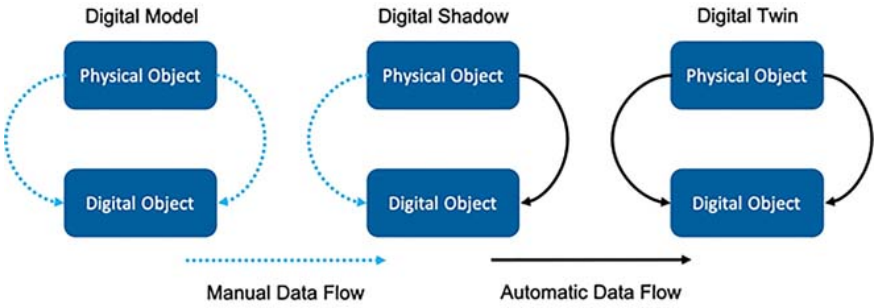
a návod pro jeho spuštění. Kapitola Vyhodnocení zhodnocuje dosažené výsledky a obsahuje grafy, jakožto důkazy jejich dosažení. Závěr shrnuje obsah práce a jakých výsledků bylo dosaženo.

Kapitola 2
Použité technologie

Tato část představuje a popisuje jednotlivé technologie, které byly v práci použity.

2.1 Digitální dvojče

Existuje mnoho definic digitálního dvojčete [KKT+18]. Většina odborníků rozeznává mezi podobnými projekty tři termíny: *digitální model*, *digitální stín* a *digitální dvojče*. Ty se liší ve způsobu komunikace mezi reálným a digitálním objektem.



Obrázek 2.1: Digitální model, digitální stín a digitální dvojče (převzato z [FFDB20])

Podle [KKT+18, FFDB20, SW22] *digitální model* označuje digitální reprezentaci, buď plánovaného, nebo již existujícího objektu v reálném světě, která nepoužívá žádnou formu automatické výměny dat s reálným objektem. Změna stavu reálného objektu nemá vliv na stav digitálního modelu a naopak.

Na základě definice digitálního modelu je definován *digitální stín*. Pokud navíc existuje automatizovaný jednosměrný tok dat z reálného objektu, jedná se o digitální stín. Změna stavu reálného objektu vyvolá změnu stavu objektu digitálního, ale ne naopak.

Pokud existuje tok dat oběma směry, jedná se o *digitální dvojče*. Změna stavu reálného objektu vede ke změně stavu dvojčete a naopak.

2.2 ALKS a projekt CERTICAR

ALKS (Automatic Lane-Keeping System)[Wor22] je systém, který po aktivaci řidičem samostatně udržuje automobil při jízdě v pruhu řízením podélného a příčného pohybu vozidla. Systém reaguje na dopravní situace a je schopen plně převzít řízení po delší dobu jízdy. Řidič je zároveň schopen kdykoliv potlačit funkci systému. Systém musí zvládat všechny situace včetně poruch a nesmí ohrozit bezpečnost cestujících ve vozidle ani jiných účastníků silničního provozu.

V rámci projektu CERTICAR [SVK+24], ve kterém ČVUT spolupracuje s firmou TÜV SÜD, který se zabývá testováním a certifikací systémů autonomního řízení vozidel, vyvíjí ČVUT jednoduchou verzi systému ALKS pro automobil Porsche Cayenne. Spolupráce s ČVUT umožňuje firmě zkoumat efektivitu testů pro certifikaci v rámci vývoje řídicího systému.

2.3 ROS

ROS (Robot Operating System) [MFG+22] je open source softwarový framework, který usnadňuje vývoj komplexních aplikací pro řízení robotů. Název je trochu zavádějící, protože se nejedná o operační systém.

ROS framework umožňuje vytváření komplexních aplikací prostřednictvím kompozice jednotlivých ROS nodů [MSCG23]. Každý node představuje samostatný proces, který má definovanou funkcionalitu a může komunikovat s ostatními nody pomocí tzv. ROS messages, services nebo actions. Komunikace mezi jednotlivými nody probíhá asynchronně a umožňuje vývojářům implementovat různé strategie pro propojení jednotlivých prvků robotického systému.

Pro oddělení kódu se jednotlivé nody, které spolu nějak souvisí, se můžou seskupit do jednoho balíku, který odpovídá jedné složce v souborovém systému.

Pro usnadnění spouštění celé aplikace a správu jednotlivých nodů je možné využít tzv. launch soubory. Tyto soubory obsahují informace o tom, které nody mají být spuštěny a s jakými parametry. Díky tomu lze jednoduše spouštět a spravovat složité systémy s mnoha nody a jejich komplexními interakcemi.

Výhodou této architektury je vysoká modularita a rozšiřitelnost. Další výhodou ROSu je, že jednotlivé komponenty mohou být implementovány v různých programovacích jazycích. Mezi nejpoužívanější jazyky pro implementaci patří C++ a Python.

2.4 CARLA

CARLA [DRC+17] je open source simulátor primárně pro vývoj autonomních vozidel. Simulace umožňuje efektivní vývoj a testování systémů autonomního

řízení. Použití simulace je téměř nutností pro testování v krizových dopravních situacích nebo nepříznivých klimatických podmínkách.

Simulátor CARLA používá architekturu server – klient, kde samotná simulace běží na serveru a jeden, nebo více klientů dostává průběžně informace o simulovaném světě a může simulátoru posílat příkazy. Od programátora se tedy očekává, že vytvoří svého klienta a skrze něj řídí dění v simulaci. Vývojáři simulátoru poskytují dvě knihovny pro implementaci klientů, jedna pro jazyk C++, druhá pro Python.

Výhodou simulátoru CARLA je, že existuje ROS node `carla-ros-bridge`, který se zároveň chová jako klient simulátoru a zpřístupňuje tak simulátor i dalším ROS komponentám.

2.5 Kalmanův filtr

Kalmanův filtr je algoritmus k odhadu neznámých proměnných na základě série změřených hodnot sledovaných v čase [Wik24b].

Je běžně aplikován v navigaci, řízení a kontrole vozidel, v analýze časových řad, zpracování signálů a ekonometrii. Kalmanův filtr je také používán v plánování pohybu robotů, optimalizaci trajektorií a modelování řízení pohybu centrální nervové soustavy.

Algoritmus funguje v reálném čase s fází predikce a fází aktualizace a spoléhá se na předpoklad, že systém je lineární a chyby mají normální rozdělení. Za těchto předpokladů je odhad Kalmanova filtru optimální odhad. Pro nelineární systémy byly vyvinuty rozšíření jako je rozšířený Kalmanův filtr (EKF).

Kapitola 3
Analýza a návrh

V této kapitole jsou rozebrány a rozšířeny požadavky na digitální dvojče. Pak následuje návrh vycházející z těchto požadavků, na základě kterého je dvojče implementováno.

3.1 Analýza požadavků

Tato část popisuje požadavky na digitální dvojče vozidla a jeho funkce. Cílem textu je specifikovat a odůvodnit potřebné vlastnosti a funkcionality digitálního dvojčete, podle kterých se bude řídit vlastní návrh.

3.1.1 Požadavek 1: Konvergence k reálnému vozidlu

Prvním požadavkem je zajištění toho, aby digitální dvojče vozidla konvergovalo ke stavu reálného vozidla. Stavy je zde myšleno zejména souřadnice x a y (v rovině země) a natočení kolem svislé osy z. Zejména je nutností, aby se při zastavení stavy sjednotily a nezůstávala konstantní odchylka.

Toto musí být splněno i v případě, že je vozidlo řízeno ručně, tedy když není v čele řízení systém ALKS, ale člověk.

Konvergence stavů však nesmí být moc rychlá, aby bylo možné detekovat odchylky představující chybové stavy.

3.1.2 Požadavek 2: Možnost použití simulátoru

Druhým požadavkem je vyvinutí digitálního dvojčete tak, aby bylo možné mít namísto reálného vozidla vozidlo ze simulátoru. To umožní při vývoji testovat chování vozidla s dvojčetem i v případech, kdy není přístupné reálné vozidlo.

3.1.3 Požadavek 3: Vizualizace stavu vozidel

Dalším požadavkem je možnost vizualizace stavu vozidel v reálném čase pomocí grafů. Toto umožní snadno monitorovat stav obou vozidel a konvergenci digitálního dvojčete a rychle odhalit potenciální chyby.

3.1.4 Požadavek 4: Automatické zastavení vozidla

Dalším důležitým požadavkem je implementace mechanismu automatického zastavení reálného vozidla v případě větších odchylek mezi stavem vozidla a jeho digitálním dvojčetem. Tím se minimalizuje riziko možných nebezpečných a neočekávaných situací.

Hlavním cílem tohoto požadavku je ale ukázat to, že digitální dvojče je schopné dávat jakékoli příkazy řídicímu systému a tím ovlivňovat chování reálného vozidla.

3.1.5 Požadavek 5: Řízení podle simulovaných objektů

Dalším požadavkem je umožnění řízení automobilu podle simulovaných objektů, což umožní testovat reakce vozidla na různé situace a podmínky, které není nutné mít fyzicky v reálném světě. Takovýmito objekty můžou být další vozidla na vozovce, chodci, nebo jakékoli jiné překážky. První výhodou oproti reálným objektům je, že v případě kolize s virtuálním objektem neutrpí automobil žádné škody. Druhou výhodou je cena přípravy různých scénářů jak finanční, tak časová.

3.1.6 Požadavek 6: Nastavitelné parametry

Posledním požadavkem je možnost nastavení parametrů digitálního dvojčete za běhu. Tento požadavek neslouží jen jako usnadnění práce s dvojčetem, ale je velmi důležitý z toho důvodu, že se simulace vozidla typicky chová hůře při vyšších rychlostech a je tedy nutností například navýšit toleranci odchylek pro zastavení vozidla.

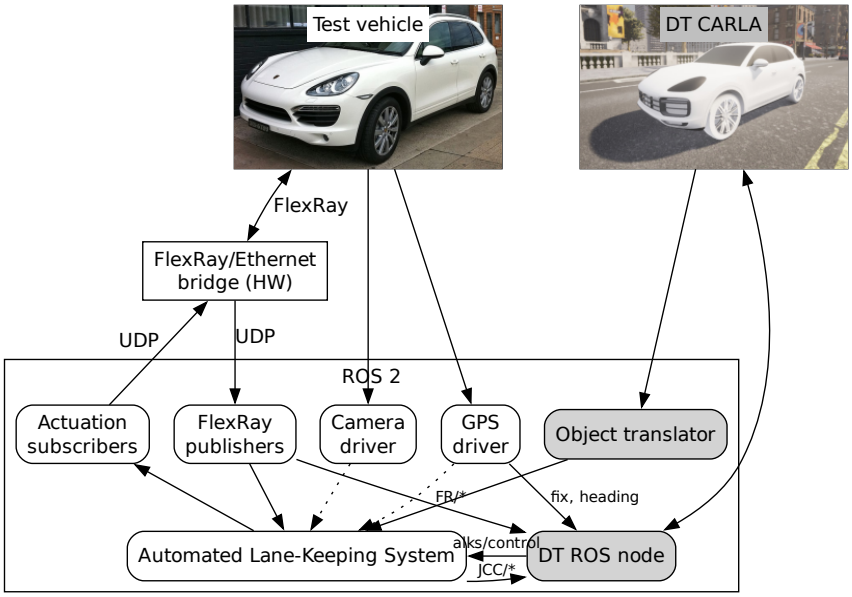
3.2 Návrh

Tato sekce navazuje na analýzu požadavků a věnuje se návrhu digitálního dvojčete.

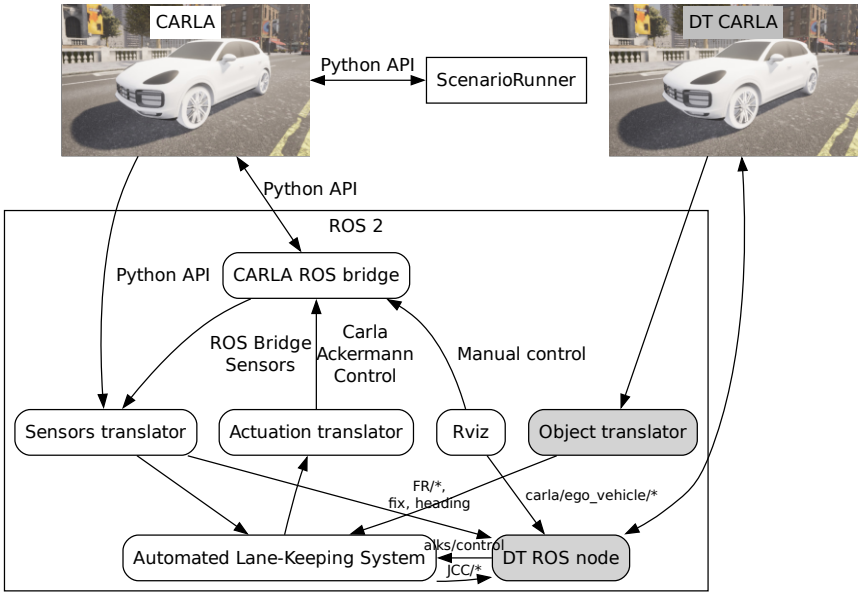
3.2.1 SW architektura

Projekt ALKS je postaven na frameworku ROS (Robot Operating System), což je framework, který usnadňuje vývoj komplexních robotických systémů. Z toho důvodu je digitální dvojče implementováno jako ROS node, což umožňuje jednoduchou integraci s ostatními komponentami projektu. Také lze jednoduše splnit požadavky na nastavitelné parametry za běhu dvojčete, které ROS podporuje, a vizualizace, které umožní program Plotjuggler.

Pro simulaci a vizualizaci dvojčete je využit simulátor CARLA, který je primárně určen pro vývoj, testování a validaci algoritmů řízení vozidel. V projetku se také již pro simulaci používá a primárně z toho důvodu byl zvolen i pro digitální dvojče. ROS node digitálního dvojčete je zároveň hlavním klientem simulátoru.



Obrázek 3.1: Softwarová architektura projetku s digitálním dvojčetem reálného automobilu. Šedá barva označuje komponenty vyvíjené v této práci.



Obrázek 3.2: Softwarová architektura projetku se simulovaným reálným vozidlem a digitálním dvojčetem

Obrázek 3.1 popisuje jednotlivé komponenty, které se podílí na řízení automobilu, a komunikaci mezi nimi. Není příliš podstatné, jak a co všechny komponenty dělají, ale zaměříme se na komponenty digitálního dvojčete (v obrázku šedou barvou) a komponenty, které přímo komunikují s digitálním dvojčetem.

Hlavní komponentou digitálního dvojčete je *DT ROS node*, tedy ROS node digitálního dvojčete, který pomocí Kalmanova filtru, dat ze sběrnice automobilu FlexRay a externí GPS a řídicích signálů z komponenty *ALKS* aktualizuje stav vozidla v simulátoru *DT CARLA*.

V případě detekce větších odchylek obou vozidel pošle *DT ROS node* signál pro zastavení komponentě *ALKS*, která následně vozidlo zastaví a počká na pokyn uživatele pro opětovné zahájení jízdy.

V konfiguraci s reálným vozidlem ze simulátoru lze vozidlo ovládat přes program RViz, interaktivní vizualizační nástroj pro ROS. Ten publikuje specifické zprávy, které nejsou přítomny při konfiguraci s automobilem v realitě. Existuje ale podobný mechanismus, který dá vědět *ALKS*, že řidič převzal řízení. Tyto systémy musí dvojče také brát v potaz pro sledování vozidla i v případě, že je řízeno manuálně.

Další komponentou digitálního dvojčete je *Object translator*, která bere data o objektech a vodorovném značení ze simulátoru a předává je *ALKS*. Tato funkcionalita již existuje v komponentě *Sensors translator* z obrázku 3.2.

Obrázek 3.2 popisuje architekturu projektu, při použití simulátoru namísto reálného vozidla. Tento případ se nijak z hlediska digitálního dvojčete nemění (až na způsob manuálního řízení) díky tomu, že je použito stejné API jako pro komponentu *ALKS*, pro kterou byla tato konfigurace navržena.

3.2.2 Použité signály

Následují použité ROS topics z obrázků 3.2 a 3.1 pro komunikaci s ostatními komponentami a popis informace, kterou nesou.

- `fix` – pozice vozidla z GPS,
- `heading` – orientace z GPS,
- `FR/ESP_PAG/ESP21` – rychlost vozidla z FlexRay sběrnice,
- `FR/ZFAS/EML01` – úhlová rychlost z FlexRay sběrnice,
- `FR/EPS/LHEPS04` – natočení kol z FlexRay sběrnice (pro manuální řízení),
- `JCC/acceleration_control` – požadované zrychlení vozidla z ALKS,
- `JCC/curvature_control` – požadované natočení kol z ALKS,
- `carla/ego_vehicle/vehicle_control_manual_override` – pokyn pro přepnutí na manuální řízení z RVizu,
- `carla/ego_vehicle/vehicle_control_cmd_manual` – manuální řízení z RVizu,

3.2.3 Režim simulátoru

Simulátor CARLA má několik režimů, které vyplývají z nastavení dvou parametrů [car] a ovlivňují jak se chová čas simulátoru vůči reálnému času. Kombinace těchto parametrů jsou:

- Asynchronní mód s variabilním časovým krokem simulace – výchozí nastavení simulátoru, kdy má server plnou kontrolu nad během času; čas v simulátoru běží stejně rychle jako reálný čas,
- Asynchronní mód s fixním časovým krokem simulace – server počítá další kroky jak rychle je to možné; čas v simulátoru běží rychleji, nebo pomaleji než reálný čas v závislosti na délce časového kroku,
- Synchronní mód s fixním časovým krokem simulace – klient určuje kdy server počítá další krok simulace; čas v simulaci běží v závislosti na rozdílu délky časového kroku a doby mezi pokyny klienta pro další krok simulace,
- Synchronní mód s variabilním časovým krokem simulace – není vhodné z toho důvodu, že simulátor nemůže využít techniky physics substepping a simulace je tudíž velmi nepřesná.

Režim simulátoru reálného vozidla musí být asynchronní mód s variabilním časovým krokem, aby simulace běžela v reálném čase. Režim simulátoru dvojčete by mohl být jak asynchronní s variabilním krokem, tak synchronní s fixní délkou kroku. Byl ale vybrán synchronní mód.

3.2.4 Kalmanův filtr

V této části je problém digitálního dvojčete formulován matematickými rovnicemi a jsou zde odvozeny použité vzorce pro výpočet stavu dvojčete.

Mějme stavové vektory $\mathbf{x}_\mathrm{R}, \mathbf{x}_\mathrm{T}$, vektory měření $\mathbf{y}_\mathrm{R}, \mathbf{y}_\mathrm{T}$, kovarianční matici měření $\mathbf{R}$, kovarianční matici procesu $\mathbf{Q}$, počáteční kovarianční matici $\mathbf{P}$ a (lineární nebo linearizovaný) model měření $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \mathbf{y}_\mathrm{T} = \mathbf{C}\mathbf{x}_\mathrm{T}$.

- Časový krok (time/prediction step):

$$\mathbf{P}_{new} = \mathbf{A}\mathbf{P}\mathbf{A}^\top + \mathbf{Q} \tag{3.1}$$

- Datový krok (data/update step):

$$\mathbf{K} = \mathbf{P}\mathbf{C}^\top(\mathbf{C}\mathbf{P}\mathbf{C}^\top + \mathbf{R})^{-1} \tag{3.2}$$

$$\mathbf{x}_{\mathrm{T,new}} = \mathbf{x}_\mathrm{T} + \mathbf{K}(\mathbf{y}_\mathrm{R} - \mathbf{y}_\mathrm{T}) \tag{3.3}$$

$$\mathbf{P}_\mathrm{new} = (\mathbf{I} - \mathbf{K}\mathbf{C})\mathbf{P} \tag{3.4}$$

V našem případě:

$$\mathbf{x} = \mathbf{y} = \begin{bmatrix} x \\ y \\ \theta \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}, \mathbf{C} = \mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.5}$$

kde x je souřadnice na ose x, y je souřadnice na ose y, θ je úhel natočení vozidla kolem osy z a veličiny s tečkou značí jejich derivace v čase (rychlosti). Matice A získaná diskretizací (Eulerovou metodou) rigid body modelu a linearizací je:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.6}$$

Po zohlednění všech těchto skutečností se rovnice zjednoduší následovně:

- Časový krok (time/prediction step):

$$\mathbf{P}_{new} = \mathbf{A}\mathbf{P}\mathbf{A}^{\top} + \mathbf{Q} \tag{3.7}$$

- Datový krok (data/update step):

$$\mathbf{K} = \mathbf{P}(\mathbf{P} + \mathbf{R})^{-1} \tag{3.8}$$

$$\mathbf{x}_{\mathrm{T,new}} = \mathbf{x}_{\mathrm{T}} + \mathbf{K}(\mathbf{y}_{\mathrm{R}} - \mathbf{y}_{\mathrm{T}}) \tag{3.9}$$

$$\mathbf{P}_{\mathrm{new}} = (\mathbf{I} - \mathbf{K})\mathbf{P} \tag{3.10}$$

Kapitola 4
Implementace

Tato část obsahuje detailnější vhled do kódu digitálního dvojčete. Dále následuje návod pro spuštění všech potřebných komponent.

4.1 Kód

Funkcionalita digitálního dvojčete je implementována v jednom ROS balíku.

```
.
|-- config
|   `-- plotjuggler.xml
|-- digital_twin
|   |-- digital_twin.py
|   |-- __init__.py
|   `-- modules
|       |-- carla_client.py
|       |-- conversions.py
|       `-- kalman_filter.py
|-- launch
|   |-- real.launch.py
|   `-- twin.launch.py
...
```

Tento balík obsahuje kód digitálního dvojčete, dva launch soubory a konfigurační soubor pro vizualizační program PlotJuggler. Kód je rozdělen do modulů, ačkoliv obsahuje pouze jeden ROS node. Moduly slouží jako knihovny pro hlavní soubor `digital_twin.py`, který node obsahuje.

ROS node má vnitřní timer běžící na frekvenci 20 Hz, který periodicky volá hlavní funkci `tick`. Nezávisle na tom zpracovává přicházející zprávy a ukládá z nich informace do vnitřních proměnných.

Při volání funkce `tick` se provede časový krok Kalmanova filtru a i datový krok, pokud v mezičase přišel nový stav reálného vozidla.

```
# digital_twin.py
class DigitalTwin(rclpy.node.Node):
    ...
```

```
    def tick(self):
        self.kalman_update()
        self.carla_client.tick()

    def kalman_update(self):
        t = time.monotonic()
        dt = t - self.last_kalman_update_time
        self.last_kalman_update_time = t
        self.kalman_filter.time_step(dt)

        if all(self.y_r_updated):
            self.y_r_updated = [False] * 6

            y_r = self.y_r
            y_t = self.carla_client.get_vehicle_state()
            x_t_new = self.kalman_filter.data_step(y_r, y_t)
```

Implementace rovnic Kalmanova filtru je v souboru `kalman_filter.py` ve třídě `DTKalmanFilter`. Parametry Kalmanova filtru (matice $\mathbf{P}, \mathbf{Q}, \mathbf{R}$) jsou odvozeny z parametrů ROS nodu. Počáteční kovarianční matice $\mathbf{P}$ je vždy jednotková a matice $\mathbf{Q}, \mathbf{R}$ se odvodí z parametrů $\mathrm{Q_{xy}}$, $\mathrm{Q_{theta}}$, $\mathrm{Q_{dxy}}$, $\mathrm{Q_{dtheta}}$, $\mathrm{R_{xy}}$, $\mathrm{R_{theta}}$, $\mathrm{R_{dxy}}$ a $\mathrm{R_{dtheta}}$ takto:

$$\mathbf{Q} = \mathrm{diag}\left(10^{\mathrm{Q_{xy}}}, 10^{\mathrm{Q_{xy}}}, 10^{\mathrm{Q_{theta}}}, 10^{\mathrm{Q_{dxy}}}, 10^{\mathrm{Q_{dxy}}}, 10^{\mathrm{Q_{dtheta}}}\right),$$

$$\mathbf{R} = \mathrm{diag}\left(10^{\mathrm{R_{xy}}}, 10^{\mathrm{R_{xy}}}, 10^{\mathrm{R_{theta}}}, 10^{\mathrm{R_{dxy}}}, 10^{\mathrm{R_{dxy}}}, 10^{\mathrm{R_{dtheta}}}\right).$$

```
# modules/kalman_filter.py
class DTKalmanFilter:
    ...
    def time_step(self, dt):
        A = np.identity(6) + dt * np.eye(6, k=3)
        self.P = A @ self.P @ A.T + self.Q

    def data_step(self, y_r, y_t):
        self.K = self.P @ np.linalg.inv(self.P + self.R)
        self.P = (np.identity(6) - self.K) @ self.P
        y_diff = np.array(y_r) - np.array(y_t)
        y_diff[2] = (y_diff[2] + np.pi) % (2 * np.pi) - np.pi
        return np.array(y_t) + self.K @ y_diff
```

Modul `carla_client.py` obsahuje třídu `DTCarlaClient`, která slouží jako abstrakce nad Python API simulátoru. Simulátor dvojčete běží synchronně s vnitřním timerem nodu s fixním časovým krokem [car]. Ostatní parametry jako je adresa a port serveru lze nastavit přes parametry ROS nodu.

Modul `conversions.py` obsahuje různé konverze veličin a je používán v ROS nodu pro zpracování příchozích dat.

4.2 Návod pro uživatele

Návod je pouze pro situaci, kdy spouštíme digitální dvojče pro simulované vozidlo, ne reálné, a to z toho důvodu, že tato situace zatím nebyla testována a příkazy pro spuštění nebyly tedy vyzkoušeny.

Nejprve spustíme oba simulátory. Nemusí běžet zároveň na jednom stroji, ale pak je nutné změnit parametry IP adresy. Pokud jsme na jednom stroji, tak musíme vybrat odlišné porty pro simulátor dvojčete a simulátor reálného vozidla argumentem `-carla-rpc-port`.

```
$ .../carla/CarlaUE4.sh -carla-rpc-port=<CARLA_PORT>
$ .../carla/CarlaUE4.sh -carla-rpc-port=<CARLA_TWIN_PORT>
```

Po spuštění simulátorů se mohou spustit příslušné komponenty v ROSu přes launch soubory v balíku `digital_twin`. V nich se používají proměnné prostředí pro určení adres a portů obou simulátorů.

```
$ export CARLA_HOST=<CARLA_HOST>
$ export CARLA_PORT=<CARLA_PORT>
$ export CARLA_TWIN_HOST=<CARLA_TWIN_HOST>
$ export CARLA_TWIN_PORT=<CARLA_TWIN_PORT>
```

Jejich výchozí hodnoty jsou adresa a port `127.0.0.1:2000`.

Pokud máme tyto proměnné v prostředí odkud spouštíme launch soubory můžeme pokračovat.

```
$ ros2 launch digital_twin real.launch.py
$ ros2 launch digital_twin twin.launch.py
```

Po načtení všech komponent by se mělo zobrazit okno aplikace RViz s modelem vozidla a v obou oknech simulátoru by měl být stejný model automobilu. Po chvíli by se měli obě vozidla rozjet.

Vizualizace jsou připravené pomocí programu PlotJuggler. Načtení konfiguračního souboru se provede takto:

```
$ plotjuggler -l .../digital_twin/config/plotjuggler.xml
```

Pro přenastavení parametrů za běhu systému je vhodné použít RQt[1] nebo nějaký ekvivalent.

```
$ rqt
```

Výsledná obrazovka bude vypadat podobně jako na 1.1.

[1] https://github.com/ros-visualization/rqt

Kapitola 5
Vyhodnocení

V této části jsou rozebrány výsledky práce, tedy funkčnost digitálního dvojčete, na třech scénářích. Dále jsou diskutovány časové vlastnosti systému.

5.1 Testovací scénáře

K testování bylo ve všech scénářích použito pouze simulované reálné vozidlo, jelikož v době testování nebyl automobil k dispozici. Ve všech scénářích byla použita mapa `Town06` z balíčku přídavných map simulátoru CARLA.

5.1.1 Sledování reálného vozidla

Tento scénář testuje funkci sledování reálného vozidla. Systém ALKS byl ve všech měření nastaven na rychlost 50 km/h. Toto je maximální rychlost, kterou se snaží po celou dobu jízdy udržovat.

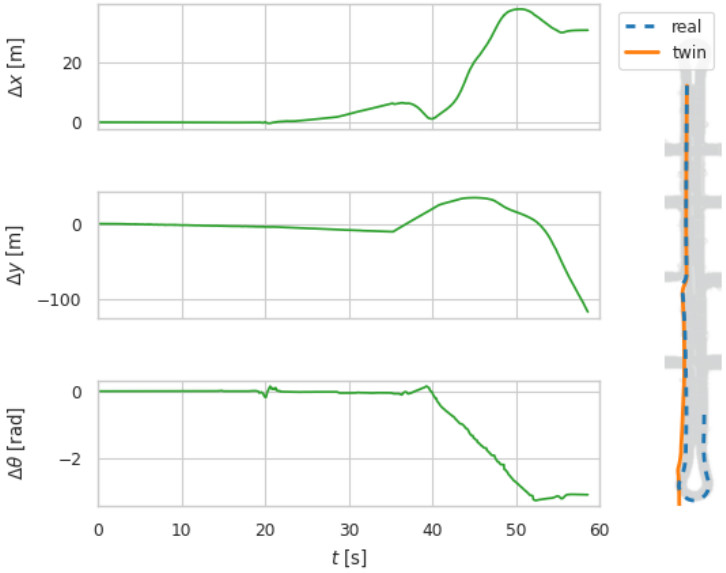
První obrázek 5.1 ukazuje paralelní jízdu dvou vozidel ze simulátoru, které jsou řízeny stejnými signály. Je vidět, že se po nějaké době začnou rozcházet. Tato situace ukazuje jak se pohybuje vozidlo v simulátoru dvojčete bez aktualizování jeho stavu pomocí Kalmanova filtru, a tedy proč je tento přístup nutný. Další obrázek 5.2 pak ukazuje stejný scénář s aktualizováním stavu vozidla dvojčete pomocí Kalmanova filtru.

Na obrázcích 5.4, 5.5 a 5.6 je ukázáno, že dvojče úspěšně sleduje stav vozidla po celou dobu jízdy. V horní části každého grafu je průběh veličiny obou vozidel v čase. Spodní část ukazuje rozdíl těchto řad. Grafy ukazují pouze 40 sekundový úsek, ale takto se dvojče chová i při delších pokusech.

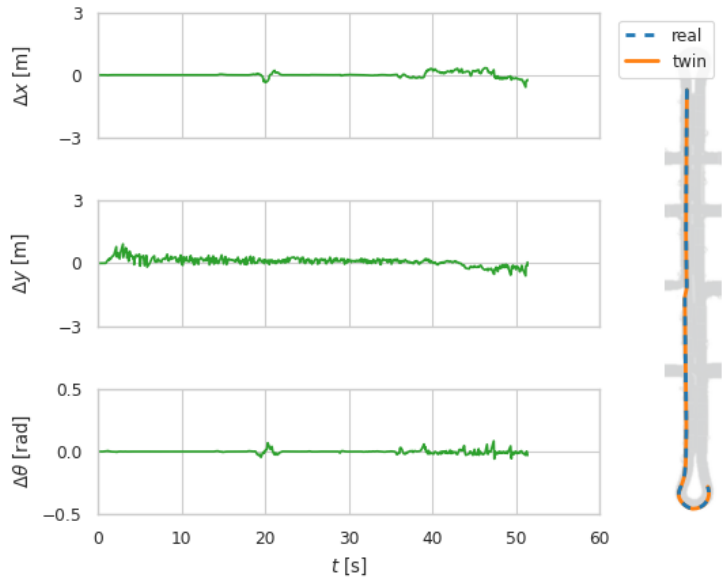
Velikosti odchylek závisí na parametrech Kalmanova filtru, na zpoždění síťové komunikace mezi reálným vozidlem a dvojčetem a také na trajektorii a rychlosti vozidel.

V tomto konkrétním případě byly použity hodnoty parametrů Kalmanova filtru $q_{\mathrm{xy}} = r_{\mathrm{xy}} = -3, q_\theta = -1$, ostatní rovny nule, což odpovídá maticím

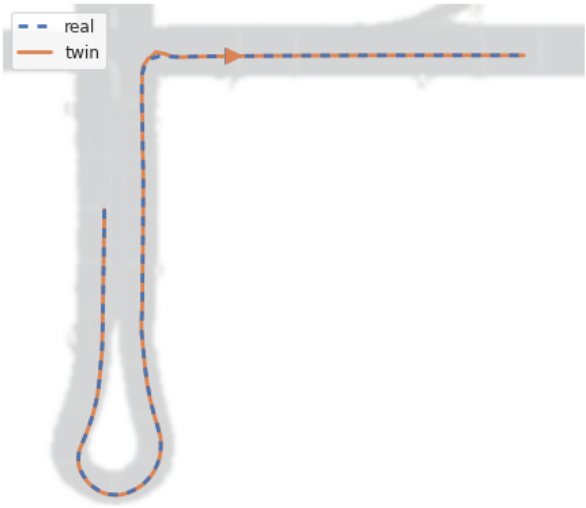
$$\mathbf{Q} = \mathrm{diag}\left(10^{-3}, 10^{-3}, 10^{-1}, 1, 1, 1\right), \quad \mathbf{R} = \mathrm{diag}\left(10^{-3}, 10^{-3}, 1, 1, 1, 1\right).$$



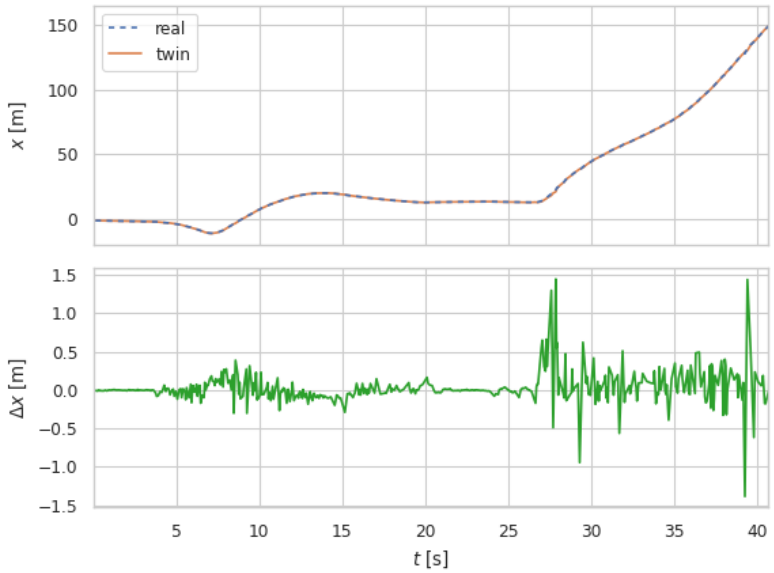
Obrázek 5.1: Jízda vozidel bez použití KF



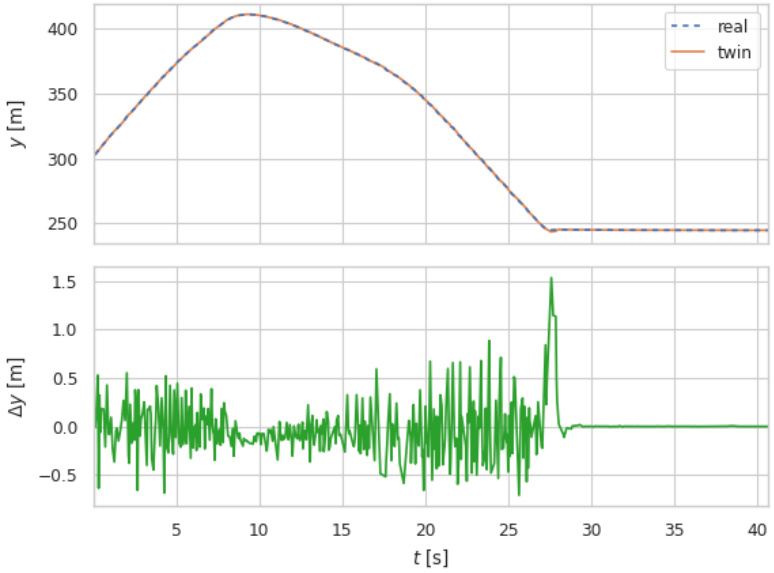
Obrázek 5.2: Jízda vozidel s použitím KF



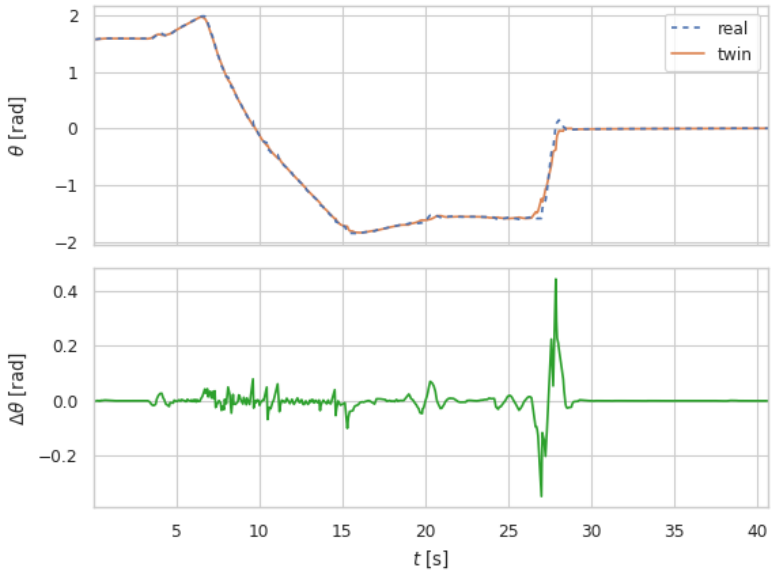
Obrázek 5.3: Mapa s trajektorií vozidel



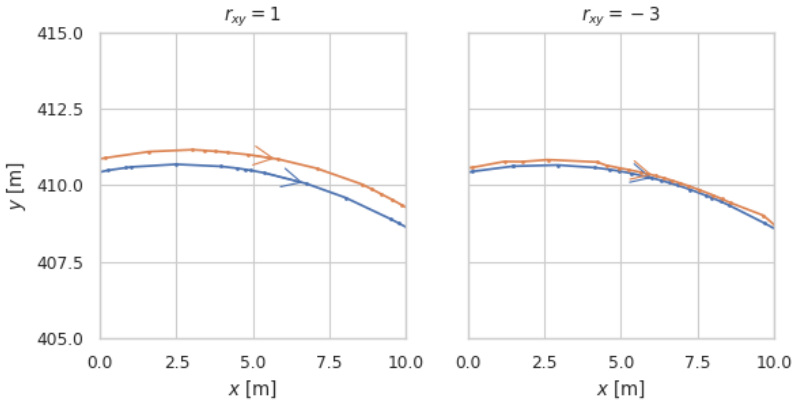
Obrázek 5.4: Vývoj a rozdíl mezi stavy vozidel z hlediska souřadnice x



Obrázek 5.5: Vývoj a rozdíl mezi stavy vozidel z hlediska souřadnice y

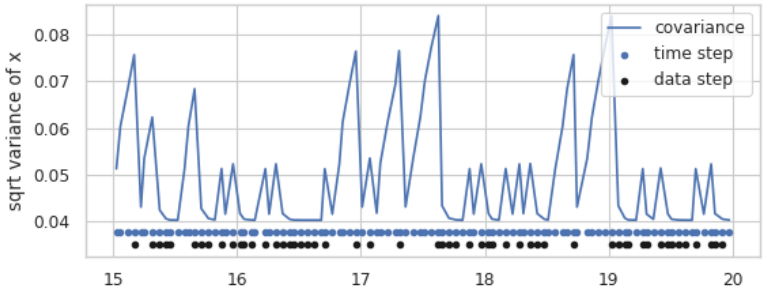


Obrázek 5.6: Vývoj a rozdíl mezi stavy vozidel z hlediska natočení kolem osy z



Obrázek 5.7: Vliv parametrů na trajektorii dvojčete

Obrázek 5.7 ukazuje, že změnou parametrů lze upravovat chování dvojčete. V grafech jsou zobrazeny dráhy dvojčete a reálného vozidla při stejném scénáři, ale s různými hodnotami parametru r_{xy}. Modrá křivka odpovídá reálnému vozidlu, oranžová dvojčeti. Tečky na trajektorii ukazují pozici vozidel při datovém kroku. Šipky ukazují směr pohybu a zároveň odpovídají jednomu datovému kroku, aby byla lépe vidět vzájemná pozice vozidel.



Obrázek 5.8: Vliv zpoždění na kovarianci

Na obrázku 5.8 je vidět růst kovariancí matice $\mathbf{P}$, při prodlevách mezi jednotlivými měřeními stavů reálného vozidla. Modré tečky značí čas, kdy se provedl časový krok Kalmanova filtru a černé značí datový krok. Modrá čára je položka kovarianční matice $\mathbf{P}$ v prvním řádku a prvním sloupci.

Z grafu je vidět, že v místech, kde jsou delší prodlevy mezi jednotlivými datovými kroky filtru, je kovariance vyšší. Kovariance se postupně zvyšuje při každém časovém kroku, kdy neproběhlo nové měření, a tudíž neproběhl datový krok, kde by se kovariance zase snížila. Příčina časových prodlev

měření může být zatížení systému, které zpomalí komunikaci mezi dvojčetem a reálným vozidlem.

Kovariance Kalmanova filtru typicky koreluje s odchylkou [Wik24b, Wik24a]. To ale pouze za předpokladu, že vybraný model věrně modeluje skutečnost a zároveň je lineární. To v našem případě neplatí. Jako model je použito simulované vozidlo ze simulátoru, které není lineárním systémem.

5.1.2 Nouzové zastavení při detekci odchylek

V následujícím scénáři byla do systému uměle zavedena chyba do digitálního dvojčete, která spočívá ve změně znaménka hodnot požadovaného natočení kol, které přicházejí z ALKS. Vozidlo v simulátoru digitálního dvojčete pak zatáčí na druhou stranu, než vozidlo ve skutečnosti.

Tato funkce je užitečná v situacích, kdy jsou řídicí algoritmy vozidla odladěny v simulátoru a fungují tam správně, ale při testech s reálným vozidlem by se chovaly výrazně jinak.

Nesrovnalosti pak částečně kompenzuje Kalmanův filtr v závislosti na parametrech. Proto je důležité správné nastavení jak parametrů filtru, tak nastavení tolerancí odchylek. Kovariance měření (reálného vozidla) $\mathbf{R}$ musí být určitě větší než kovariance procesu (dvojčete) $\mathbf{Q}$, tzn. že předpokládáme větší chybu u reálného vozidla, než v simulaci. Jinak se totiž stav vozidla dvojčete téměř okamžitě nastaví na novou změřenou hodnotu a není tedy prostor pro detekci větší odchylky. Na druhou stranu nesmí být o moc větší, jinak bude odchylka moc velká.

Pro tento scénář byly zvoleny tyto matice $\mathbf{Q}$ a $\mathbf{R}$:

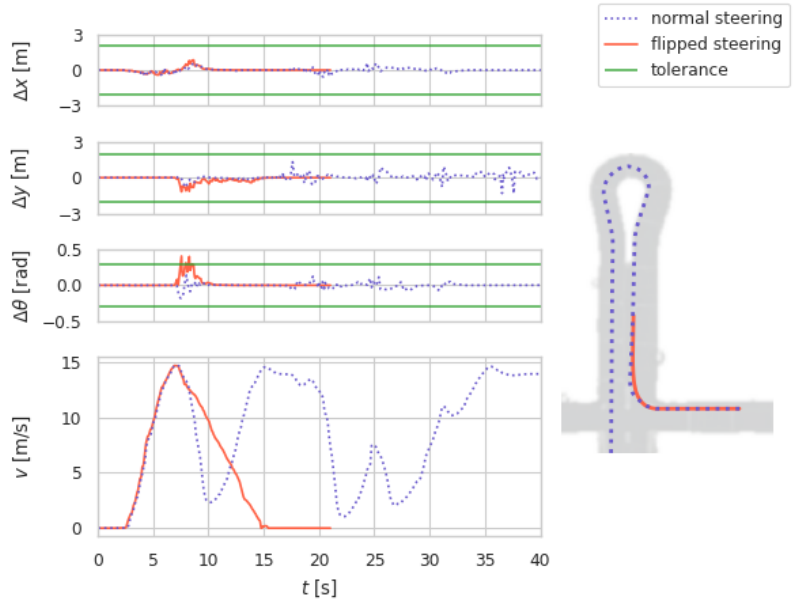
$$\mathbf{Q} = 10^{-2} \cdot \mathbf{I}, \quad \mathbf{R} = 10^{-1} \cdot \mathbf{I}.$$

K nim byly vybrány tolerance $\pm 2\,\mathrm{m}$ pro souřadnice x a y a tolerance $\pm 0{,}3\,\mathrm{rad}$ pro odchylku v natočení vozidel, což odpovídá přibližně 17 stupňům.

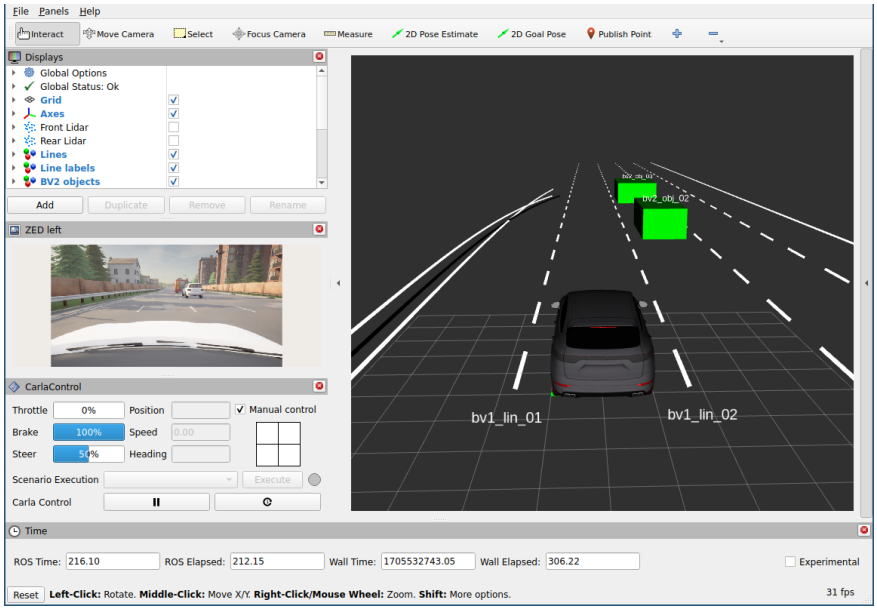
Obrázek 5.9 ukazuje funkčnost automatického brždění. V levé horní části jsou zobrazeny odchylky a tolerance, vlevo dole rychlost reálného vozidla a vpravo trajektorie reálného vozidla. Červenou křivkou je vyznačen scénář s chybou ve znaménku natočení kol, modrou tečkovanou křivkou tentýž scénář bez této chyby. Funkčnost brzdy je vidět z grafu rychlosti, kde od překročení tolerance natočení kol postupně klesá až na nulu, kde zůstane až do pokynu řidiče.

5.1.3 Virtuální objekty

Posílání objektů a vodorovného značení ze simulátoru dvojčete funguje podle očekávání. Automobil je řízen systémem ALKS na základě objektů a vodorovného značení ze simulátoru dvojčete (viz obrázek 5.10).

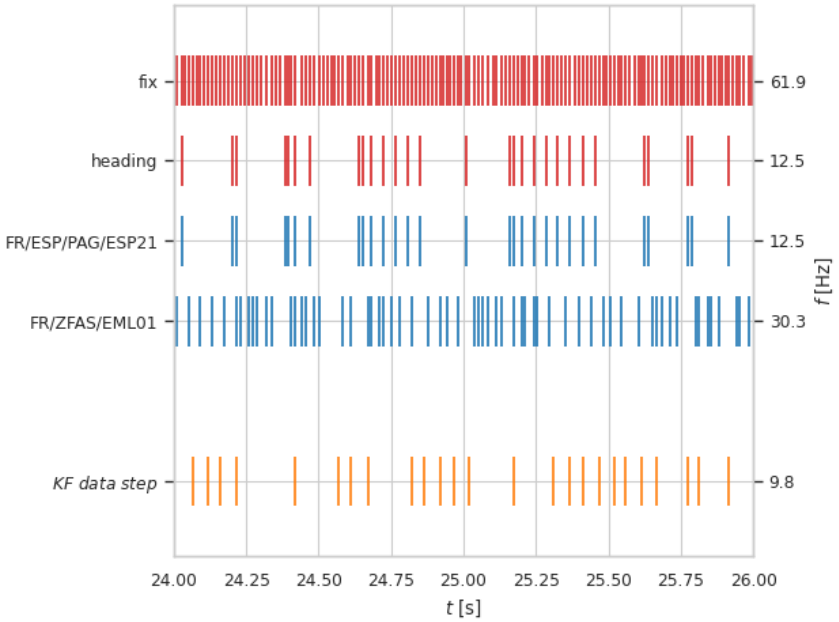


Obrázek 5.9: Zastavení vozidla při detekci odchylek v úhlu mezi vozidly



Obrázek 5.10: Objekty a vodorovné značení ze simulátoru

5.2 Analýza časových vlastností



Obrázek 5.11: Časové vlastnosti přicházejících zpráv a datového kroku

Obrázek 5.11 zobrazuje příchozí časy zpráv z ROSu svislými čarami a jejich frekvence na pravé ose. Poslední linka shora nezobrazuje příchozí zprávy, ale odchozí zprávy, které slouží pro vizualizaci stavů vozidel a jejich odchylek. Ta se publikuje při datovém kroku Kalmanova filtru.

Datový krok se provádí až když v každém proudu zpráv, týkající se stavu reálného vozidla, dorazila nová zpráva. Proto je frekvence publikování těchto zpráv (průměrně 9,8 Hz) menší, než minimální frekvence příchozích proudů zpráv (12,5 Hz).

Zároveň je ze vzoru prodlev zpráv `heading` a prodlev datového kroku vidět, že mezi zpracováním zpráv `heading` a publikováním stavu *KF data step* je relativně velké zpoždění (cca 200 ms). To může být způsobeno výpočetní náročností serializace zpráv v Pythonu, která je výrazně vyšší než v kompilovaných jazycích.

Z grafu je vidět, že by bylo lepší zpracovávat jednotlivé proudy zpráv nezávisle a aktualizovat odhad stavu na základě neúplných informací. Kalmanův filtr toto umožňuje a pravděpodobně by to v podobných situacích vedlo ke snížení rozdílu mezi stavy reálného vozidla a dvojčete.

Nezapomeňme, že takto se systém chová, když běží oba simulátory na jednom počítači. Ten je tedy celkově více zatížen, než když by komunikoval s reálným automobilem, ale zpoždění tvořené síťovou komunikací je výrazně menší.

Kapitola 6

Závěr

Požadavky kladené na tuto práci byly úspěšně naplněny. Požadavkem byla implementace digitálního dvojčete s možností nouzového brždění při detekci větších odchylek a řízení na základě objektů ze simulátoru dvojčete. Tyto vlastnosti byly testovány na pěti scénářích, kde bylo použito simulace namísto reálného automobilu.

Dvojče při jízdě rychlostí až 50 km/h sleduje reálné vozidlo s odchylkami menšími než 2 m a 0,3 rad. Typicky jsou hodnoty nižší kolem 0,5 m a 0,1 rad. Odchylky závisí na parametrech Kalmanova filtru, zejména na kovariančních maticích šumů, a je možné je optimalizovat pro daný scénář. Při překročení zmíněných maximálních hodnot dojde k zastavení vozidla. Z analýzy časových vlastností vyplynulo možné vylepšení algoritmu, které by mohlo vést k dalšímu snížení odchylek.

Výsledné digitální dvojče z této práce bude používáno při dalších fázích projektu CERTICAR.

Příloha A

Literatura

[car] *CARLA Simulator — synchrony and time-step*, [Accessed 24-05-2024].

[DRC+17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun, *CARLA: An open urban driving simulator*, Proceedings of the 1st Annual Conference on Robot Learning, 2017, pp. 1–16.

[FFDB20] Aidan Fuller, Zhong Fan, Charles Day, and Chris Barlow, *Digital twin: Enabling technologies, challenges and open research*, IEEE Access **8** (2020), 108952–108971.

[KKT+18] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn, *Digital twin in manufacturing: A categorical literature review and classification*, IFAC-PapersOnLine **51** (2018), no. 11, 1016–1022, 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.

[MFG+22] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall, *Robot operating system 2: Design, architecture, and uses in the wild*, Science Robotics **7** (2022), no. 66, eabm6074.

[MSCG23] Steven Macenski, Alberto Soragna, Michael Carroll, and Zhenpeng Ge, *Impact of ROS 2 node composition in robotic systems*, IEEE Robotics and Autonomous Letters (RA-L) (2023).

[SVK+24] Michal Sojka, Jiří Vlasák, Adam Killarčík, Zdeněk Hanzálek, Vladislav Kocián, Jakub Dvořák, Leonid Tulin, Michael Bryan, and Lucie Kotanová, *CERTICAR - odborná zpráva o řešení projektu v roce 2023*, Tech. report, České vysoké učení technické a firma TÜV SÜD, 2024.

[SW22] Chris Schwarz and Ziran Wang, *The role of digital twins in connected and automated vehicles*, IEEE Intelligent Transportation Systems Magazine **14** (2022), no. 6, 41–51.

[Wik24a] Wikipedia contributors, *Extended kalman filter — Wikipedia, the free encyclopedia*, 2024, [Online; accessed 15-05-2024].

[Wik24b] _____, *Kalman filter — Wikipedia, the free encyclopedia*, 2024, [Online; accessed 10-May-2024].

[Wor22] Working Party on Automated/Autonomous and Connected Vehicles (GRVA), *Proposal for the 01 series of amendments to un regulation no. 157 (automated lane keeping systems)*, 2022, [Accessed: 22-05-2024].