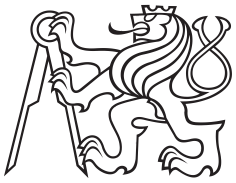
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Konvoj na bázi kamery

Vojtěch Nováček

Vedoucí: Dr. Gaël Pierre Marie Ecorchard
Obor: Robotika
Studijní program: Kybernetika a robotika
Červen 2018

Poděkování

Děkuji ČVUT, že mi je tak dobrou *alma mater*.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladus Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Datum:

.....

Abstrakt

Obsahem této práce je návrh a implementace softwarového řešení pro samostatný pohyb robotů v konvoji v čele s vedoucím vozidlem. Řešení umožňuje operátorovi řídit vedoucí vozidlo, ostatní plně autonomní vozidla ho následující stejnou cestou. Základem pro autonomní sledování vedoucího vozidla jsou technologie vyvinuté skupinou IMR využívající obrazové informace z kamer a údaje z odometrie robota.

Klíčová slova: konvoj, autonomní navigace, Map and replay, SurfNav, ROS

Vedoucí: Dr. Gaël Pierre Marie Ecorchard

Abstract

This thesis deals with the design and implementation of a software solution for the autonomous robots in a convoy led by a leading vehicle. The solution allows the operator to drive the lead vehicle while the other fully autonomous vehicles follow the same way. The solution is based on the technology developed by the IMR group using video camera information and robot odometer data.

Keywords: convoy, convoying, autonomous navigation, Map and replay, SurfNav, ROS

Title translation: Camera-Based Platooning

Obsah

Obrázky

Tabulky

ctuthesis t1606152353

Kapitola 1

Úvod

Současným trendem robotiky je nahrazení robota ovládaného operátorem jeho plně autonomní variantou. Jednou z úloh mobilní robotiky, kde se tohoto cíle dá alespoň částečně dosáhnout, je pohyb robotů v konvoji.

Konvoj je skupina robotů, která se okolním terénem pohybuje současně po pokud možno stejné trase. Je tvořen vedoucím vozidlem, které ostatní roboti následují. Pohyb vedoucího vozidla je ovládán operátorem, ostatní vozidla se podél trasy navigují již plně autonomně.

V této práci se zabývám implementací pohybu vozidel v konvoji v prostředí systému ROS, které pro svou navigaci jako jediné senzory používají jednoduché monokulární kamery instalované na robotech a údaje z odometrie robotů.

Zpracováním dat z odometrie a obrazu pomocí metod počítačového vidění vedoucí robot vytváří mapu a ostatní se podle ní po cestě navigují. Roboti si mezi sebou předávají informace potřebné pro sledovaní trasy a udržení formace v konvoji.

Jako základ řešení pro implementaci knovoje jsem použil stávající technologie a algoritmy vyvinuté skupinou *Intelligent and Mobile Robotics Group* (IMR) při ČVUT v Praze. Jde zejména o algoritmus SurfNav a navigační metodu popsanou v popsanou článku[1].

Pro úspěšnou navigace konvoje je třeba znát uraženou vzdálenost roboty,

která se určuje z odometrie robota. Odometrie robota je však zatížena aditivní chybou. Článek [1] přináší matematický důkaz, kdy za splnění určitých předpokladů je možné chybu korigovat tak, že neroste pře určitou mez.

Kapitola 2

Navigace pomocí *features* obsažených v obraze metodou SurfNav

Metoda navigace použitá pro pohyb robotů v konvoji je založená na technice *map and replay*[1]. Jde o metodu zpracovávající obrazová data z monokulárních kamer umístěných na robotech společně s informacemi o poloze robotů získaných z odometrie.

Pomocí metod počítačového vidění se z obrazu kamer detekují tzv. *features*, popsané v následující kapitole, které rozlišitelně popisují okolní prostředí. Z nich navigační metoda nejdříve ve fázi učení vytvoří mapu okolního prostředí, kterou pak používá pro fázi navigace.

Údaje získané z odometrie nejsou přesné. Nepřesnosti vznikají už přímo na senzorech, které pro určení polohy robota integrují rychlost, nebo vlivem dalších okolností, jako jsou např. prokluzující kola nebo pásy. Chyba odometrie je aditivní a narůstá s uraženou vzdáleností. To má za následek, že po ujetí větších vzdáleností je chyba už tak velká, že není pro navigaci dále použitelná.

Aby byla metoda navigace vhodná i pro přesun konvoje na větších vzdálenostech, musí se s touto vlastností vypořádat. Je potřeba zajistit, aby metoda byla dostatečně stabilní v tom smyslu, že chyba v určení pozice bude omezená a nebude divergovat. Jak je ukázáno, použitá metoda navigace toho umí dosáhnout za splnění těchto předpokladů[1]:

- robot se pohybuje po rovině,

- je k dispozici mapa k úsekům cesty obsahující *features*,
- robot na každé části cesty dokáže z detekovaných *features* a podle vytvořené mapy určit směr dalšího pohybu,
- robot může měřit, byť nepřesně, uraženou vzdálenost prostřednictím odometrie,
- kamera je namířena ve směru pohybu robota,
- a robot se pohybuje po rovné ploše,.

Postup metody SurfNav z článku[1] pro tvorby mapy a navigace dle ní je popsán v následujících kapitolách.

2.1 *Features* a *Landmarks*

Lokální obrazové *features* jsou informace o významných objektech, které lze získat výpočetními metodami z obrazu. V oboru počítačového vidění jde zejména o hrany, rohy, rozlišitelné vzory apod.[2] Obrazová data lze takto redukovat na množinu získaných *features*, často reprezentována jako jednotlivé body v obraze, a tím zjednodušit a snížit výpočetní náročnost dalšího zpravování. Důležitou vlastností *features* je, že jsou rozpoznány i v obrazech snímaných z jiné vzdálenosti nebo pod jiným úhlem. *Features* často odpovídají objektům v reálném světě, jako jsou například rohy zdí, okraje dveří apod. Tyto reálné objekty jsou v robotice označovány jako *landmarks*[2].

2.2 Tvorba mapy

Features jsou použity při tvorbě mapy. Vedoucí vozidlo, které je ovládané operátorem, snímá kamerou prostředí před sebou a zpracovává každý takto získaný snímek. V obrazu detekuje a extrahuje *features* a ukládá je pro další zpracování společně informací o vzdálenosti od startovní pozice. Dále je porovnává s již nalezenými *features* z předchozích dat. Pokud dojde ke shodě, doplní informace o aktuální uražené vzdálenosti *features* spolu s vektorem počáteční a koncové pozice ukládá do mapy.

2.3 Navigace

Následující vozidla mají k dispozici aktuální mapu vytvořenou vedoucím vozidlem. Při pohybu zpracovávají jednotlivé snímky a získávají z nich *features*. Pokud dojde ke shodě, vytvoří vektory s počátečními a koncovými pozicemi a porovná je s vektory pro dané *features* z mapy. Porovnáním zjistí úhly horizontálních odchylek vektorů, zpracuje histogram a zjistí převládající směr vyjádřený úhlem vychýlení od sledované trasy. Podle tohoto úhlu se koriguje řízení, aby se odchylka kompenzovala.

Kapitola 3

Přehled technologií a použitých metod

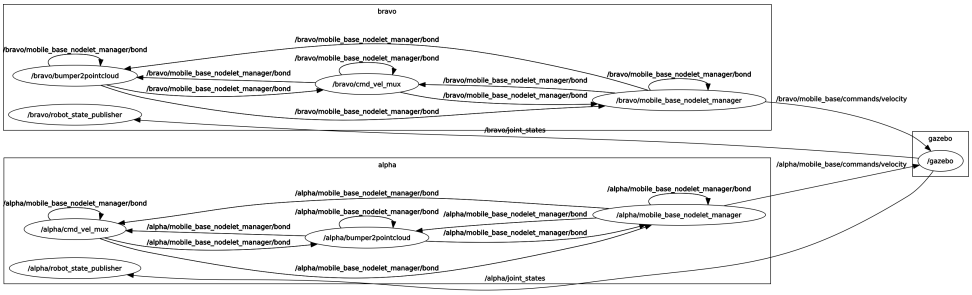
V této kapitole je popsán souhrn hlavních technologií, které byly použity pro implementaci systému konvoje robotů včetně vybraných metod počítačového vidění pro zpracování obrazu.

3.1 Robot Operating System

Robot Operating System (ROS)[6] je open-source vývojová platforma a prostředí pro běh robotických aplikací. Je vyvinuta s ohledem na provoz v heterogenním prostředí, kdy mezi sebou mohou komunikovat aplikace a komponenty běžící na různých HW platformách v síťovém prostředí s důrazem na robustnost a nízkou latenci odezvy.

Jádrem ROS jsou základní nástroje a knihovny umožňující běh dalších komponent. Software jednotlivých komponent je organizován do balíků (*ROS packages*). Součástí standardní distibuce ROSu je bohatá kolekce těchto balíků, které pokrývají většinu běžných funkcionalit a algoritmů v používaných v robotice a také rozhraní k dalším knihovnám a nástrojům. Obsahuje také software pro správu těchto balíků, další potřebné balíky tak lze jednoduše přidávat.

Procesy aplikací z balíků se spouštějí jako tzv. uzly (*nodes*), které mezi sebou vzájemně komunikují. Uzly jsou zpravidla koncipovány tak, aby vykonávaly



Obrázek 3.1: Grafová struktura spuštěných uzlu dvou robotů v systému ROS.

jednu určitou činnost, např. obsluhu hardwaru kamery, navigaci nebo řízení robota.

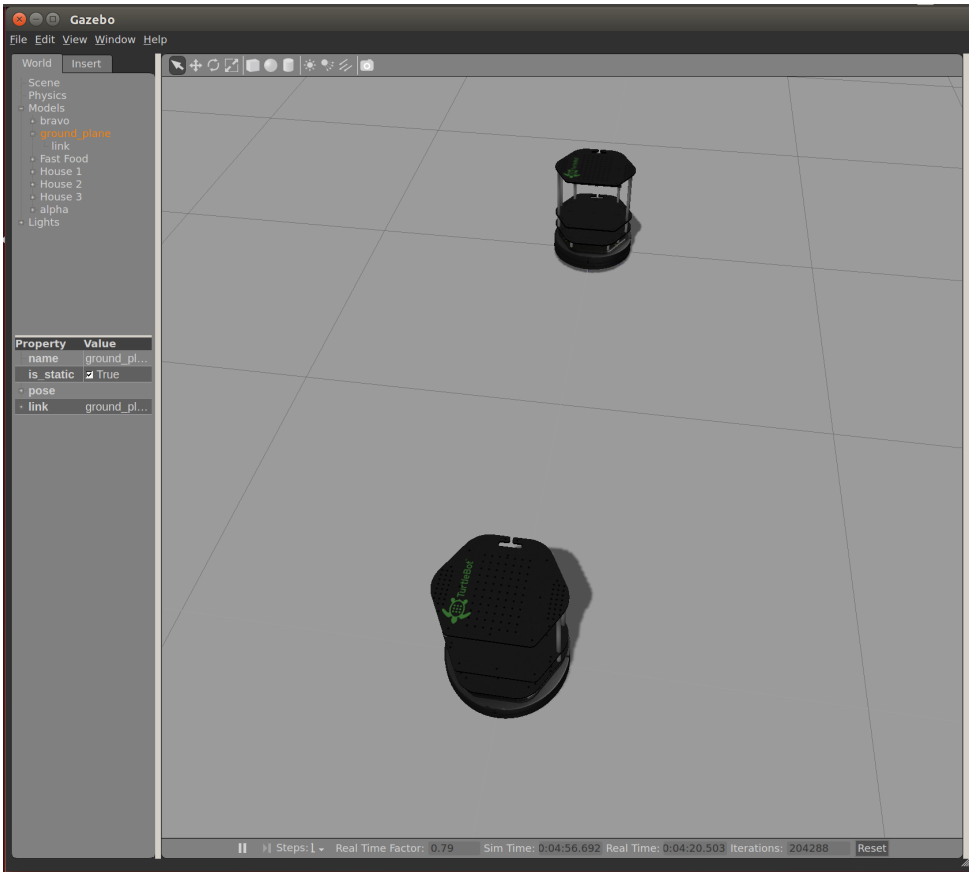
Komunikace mezi jednotlivými uzly je zprostředkováná pomocí předávání zpráv nebo volání RPC služeb (*service calls*). Pro předávání zpráv ROS používá návrhového vzoru *Publish–subscribe*. Uzel odesílající zprávy (*publisher*) nejdříve uveřejní službu pod zvoleným předmětem (*topic*) s popisem zprávy. Uzel, který chce zprávy přijímat (*subscriber*), se k odběru zpráv daného předmětu přihlásí. Komunikační vazby mezi uzly mají grafovou strukturu. Při běhu typické aplikace se tak spouští mnoho uzlů zároveň. Spouštění uzlů, komunikaci mezi nimi a registraci služeb obsluhuje hlavní řídící uzel (*ROS Master*).

Příklad spuštěných uzlů je znázorněn na obrázku 3.1.

3.2 Gazebo

Gazebo[7] je open-source robotický simulátor. Pomocí Gazeba lze simulovat realistický pohyb robotů v modelovém prostředí. Díky použitým knihovnám *ODE*, *Bullet*, *Simbody* a *DART* je zaručeno fyzikálně realistické chování simulace modelu a okolního prostředi.

Hlavní výhodou Gazeba je možnost navrhnout a testovat algoritmy bez použití reálných robotů a tím značně urychlit vývoj. Pro Gazebo existuje množství modelů robotů odpovídajích svým skutečným protějškům, jako jsou např. TurtleBot nebo Husky. Na obrázku 3.2 je zobrazeno prostředí Gazebo s modely robotů TurtleBot.



Obrázek 3.2: Dvojice robotů TurtleBot v simulátoru Gazebo.

3.3 OpenCV

Open Source Computer Vision (OpenCV)[8] je open-source knihovna vyvinutá pro zpracování obrazu a počítačové vidění s důrazem na aplikace běžící v reálném čase.

Pro softwarové řešení v této práci jsem použil zejména funkce implementující algoritmy a metody pro detekci, exktrakci a porovnání *fetaures*.

3.4 Metody počítačového vidění pro manipulaci s *Features*

Pro navigační metodu použitou v této práci existují různé metody pro detekci, extrakci a porovnání obrazových *features*. Navzájem se od sebe odlišují způsobem a kvalitou, jakým *features* detekují a ukládájí pro další zpracování. Pro navigační metodu jsem vybral takové, které mají vysokou výpočetní účinnost a jsou tak vhodné pro zpracování obrazu v reálném čase i na méně výkonných počítačích, které slouží pro běh robotů. Zároveň však ale detekují dostatečný počet *features*.

- *Features From Accelerated Segment Test (FAST)*

  Features From Accelerated Segment Test (FAST)[10][11] je metoda pro detekci *fetaures* nalézáním rohů v obraze. Metoda pro detekci používá 16 pixelů ležící na *Bresenhamově* kružnici o poloměru 3. Feature je detekována ve středu p této kružnice, pokud alespoň $N = 12$ z těchto pixelů mají všechny intenzitu větší nebo všechny intenzitu menší než p o nějakou hraniční hodnotu t.

  Hlavní výhodou této metodu je vysoká výpočetní účinnost, proto je vhodná pro rychlé zpracování obrazu v reálném čase. V závislosti na dané úloze je možné měnit hodnoty N a t. S menším N roste počet nalezených *features*, ale na úkor výpočetní účinnosti.

- *Binary Robust Independent Elementary Features (BRIEF)*

  Pro další zpracování je třeba nalezené *features* nějakým způsobem popsat, aby mezi nimi šlo jednoduše vyhledávat a vzájemně je porovnat. K takovému popisu slouží tzv. *deskriptor*. Deskriptory pro jednotlivé *features* by měly být pokud možno mezi sebou maximálně rozlišitelné, ale zároveň neměnné při transformaci obrazu.

  Binary Robust Independent Elementary Features (BRIEF)[12] je jednou z metod pro vytváření deskriptorů pro detekované *features*. Narozdíl od jiných metod jako jsou *SIFT* nebo *SURF* nevytváří pro deskriptor vektor o mnoha dimenzích, ale pouze binární řetězec. Podobnost deskriptorů tak lze zjistit výpočetně jednoduchým užitím Hammingovy vzdálenosti, což lze na počítači snadno realizovat pomocí funkce XOR. Ve výsledku je velmi rychlé tyto deskriptory vytvářet a hledat mezi nimi.

- *Fast Library for Approximate Nearest Neighbors (FLANN)*

  Fast Library for Approximate Nearest Neighbors (FLANN)[13] je knihovna řešící úlohu hledání nejbližšího souseda v prostoru o vysokém počtu dimenzí.

Je použita v algoritmu pro srovnání nalezených *features*, kdy mezi *features* hledá podobné.

Kapitola 4

Implementace

4.1 Vysvětlení pojmů

V popisu implementace jsou použity pojmy, které přibližně odpovídají pojmům diskutovaných v předchozím textu.

- *Feature*

  Feature je základním prvkem informace pro tvorbu mapy a odpovídá jednotlivým *features*, které byly popsány v kapitole 2.1.

- *Deskriptor*

  Deskriptor je datová struktura pro vyhledávání a porovnání *features* vytvořená algoritmen BRIEF 3.4.

- *Landmark*

  Pokud je stejná feature detekována v obrazech v určitém rozmezí vzdálenosti, je k ní vytvořen Landmark. Landmark je datová struktura obsahující

  - jedinečný identifikátor,
  - deskriptor dané feature,
  - pozici feature v obraze při počáteční detekci,
  - pozici feature v obraze, kdy byla feature detekována naposledy,
  - vzdálenost první detekce od startu segmentu

- a vzdálenost poslední detekce od startu segmentu.

Landmark může být dvojího typu. Buď už v dalších zpracovávaných obrazech není fetaure landmarku detekována a v takovém případě je landmark *dokončený*. V opačném případě aktualizace landmarku stále pokračuje a landmark je *nedokončený*.

- *Segment*

  Pojem segment je použitý v dalším textu ve dvou významech. Buď ve smyslu části trasy, po které se vozidla pohybují. Nebo jako datová struktura, jejíž součástí je mapa složená z jednotlivých landmarků, které byly vytvořeny zpracováním *features* v obraze. Dále obsahuje orientaci robota na začátku segmentu a uraženou vzdálenost.

4.2 Vozidla v konvoji

Vozidla se terénem pohybují v konvoji tak, že jedno následuje druhé. Pro implementaci programu řídící konvoj se rozlišují tři druhy vozidel:

- *Vedoucí vozidlo*

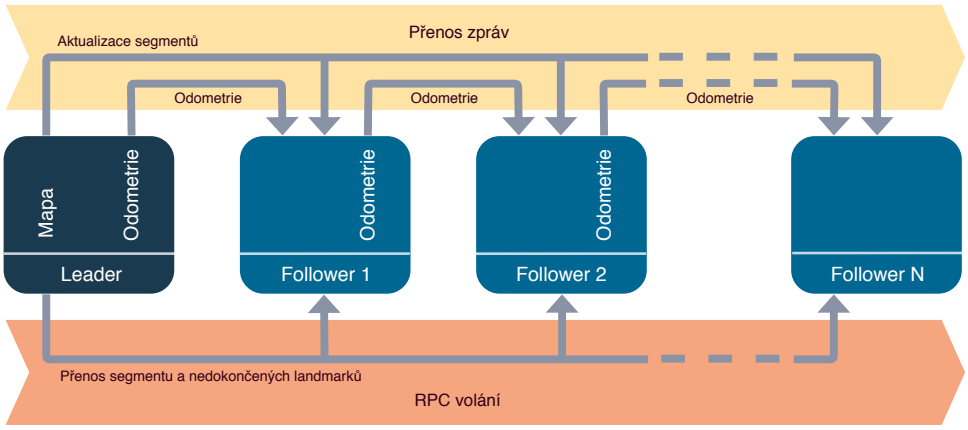
  Vedoucí vozidlo je v konvoji jediné a pohybuje se v čele konvoje. Jako jediné je řízeno operátorem. Zpracovává obrazová data z kamery a podle nich vytváří mapu okolí pro jednotlivé segmenty trasy. Po vyžádání mapu poskytuje sledujícím vozidlům, které se pohybují za ním v konvoji.

  Operátor řídí vozidlo přímo přes uzel ovládající HW robota prostřednictvím klávesnice, joysticku apod. nebo přímým zasíláním daných zpráv v systému ROS či použitím jiné aplikace. Jediným vstupem pro vedoucí vozidlo je odometrie robota a obrazová data z kamery.

- *Sledující vozidlo*

  Sledující vozidlo autonomně sleduje trasu podle mapy, kterou projelo vedoucí vozidlo. Zpracovává obrazová data z kamery a porovnává je s obdrženou mapou. Porovnáním koriguje natáčení robota, aby se vozidlo drželo projeté trasy.

  Kromě své odometrie potřebuje znát také odometrii sousedního vozidla před ním, aby si od něho mohl zachovat dostatečný odstup. O odstup vozidla za ním se nestará, protože je již analogicky zaručeno sledujícím vozidlem za ním. Rozestupy mezi vozidly mohou být libovolně velké a jejich velikost, pokud neklesne pod určitou mez, neovlivňuje kvalitu navigace.



Obrázek 4.1: Schéma konvoje s vyznačenou vzájemnou komunikaci mezi vozidly v konvoji.

Ke svému pohybu nepotřebuje znát odometrii vedoucího vozidla. Vzhledem k tomu, že je vedoucí vozidlo vždy v čele konvoje, je zaručeno, že mapa pro trasu již existuje.

- *Sousední vozidlo*

  Sousedním vozidlem se míní vozidlo pohybující se před sledujícím vozidlem. Může to být buď vedoucí vozidlo, nebo jiné sledující vozidlo v konvoji.

Sledující vozidla konvoje jsou na začátku v blízkosti startovní pozice vedoucího vozidla. Po startu vedoucího vozidla se ostatní přesouvají na jeho startovní pozici a postupně se řadí do konvoje. Vzdálenost by neměla být příliš velká, protože pro přesun vozidel na startovní pozici se sledující vozidla spoléhají výhradně na nepřesnou odometrii.

Schéma konvoje a jejich vzájemné komunikace je zobrazena na obrázku 4.1.

4.3 Softwarové řešení

Softwarové řešení implementující vozidla v konvoji se skládá ze dvou programů. První je pro uzel vedoucího vozidla tvořícího mapu (*leader*) a druhý pro uzly sledujících vozidel (*follower*).

4.3.1 Vedoucí robot (*leader*)

Program vedoucího robota se při spuštění přihlásí k odběru zpráv odometrie robota a obrazových dat z kamery. Zároveň uveřejní službu pro zasílání zpráv aktualizací landmarků v daném segmentu a zpětné vazby pro dohledovou aplikaci. Dále oznámí RPC služby pro počáteční přenos segmentu a nedokončených landmarků.

Na začátku trasy si pro první segment uloží počáteční polohu a orientaci robota. Průběžně přijímá zprávy z odometrie a kamery a z nich tvoří mapu aktuálního segmentu trasy. S tvorbou mapy pokračuje do té doby, než dojde operátorem k zastavení robota, kdy si uloží koncovou polohu a orientaci robota. Při opětovném rozjezdu se porovná rozdíl mezi aktuální orientací a orientací při zastavení. Pokud je rozdíl větší, než nastavená maximální hodnota, aplikace ukončí zpracování aktuálního segmentu trasy a segment s mapou uloží. Tím je zaručen požadavek kolinearity navazujích segmentů z předpokladů metody navigace 2. Ihned poté začne s tvorbou mapy pro nový segment.

Během tvorby mapy vedoucí robot zasílá prostředictvím zpráv nově přidané nebo aktualizované landmarky. Program zasílá landmarky včetně ještě nedokončených, protože musí být zaručeno, že se podle nich může sledující robot navigovat, i když detekce daného landmarku stále ještě pokračuje.

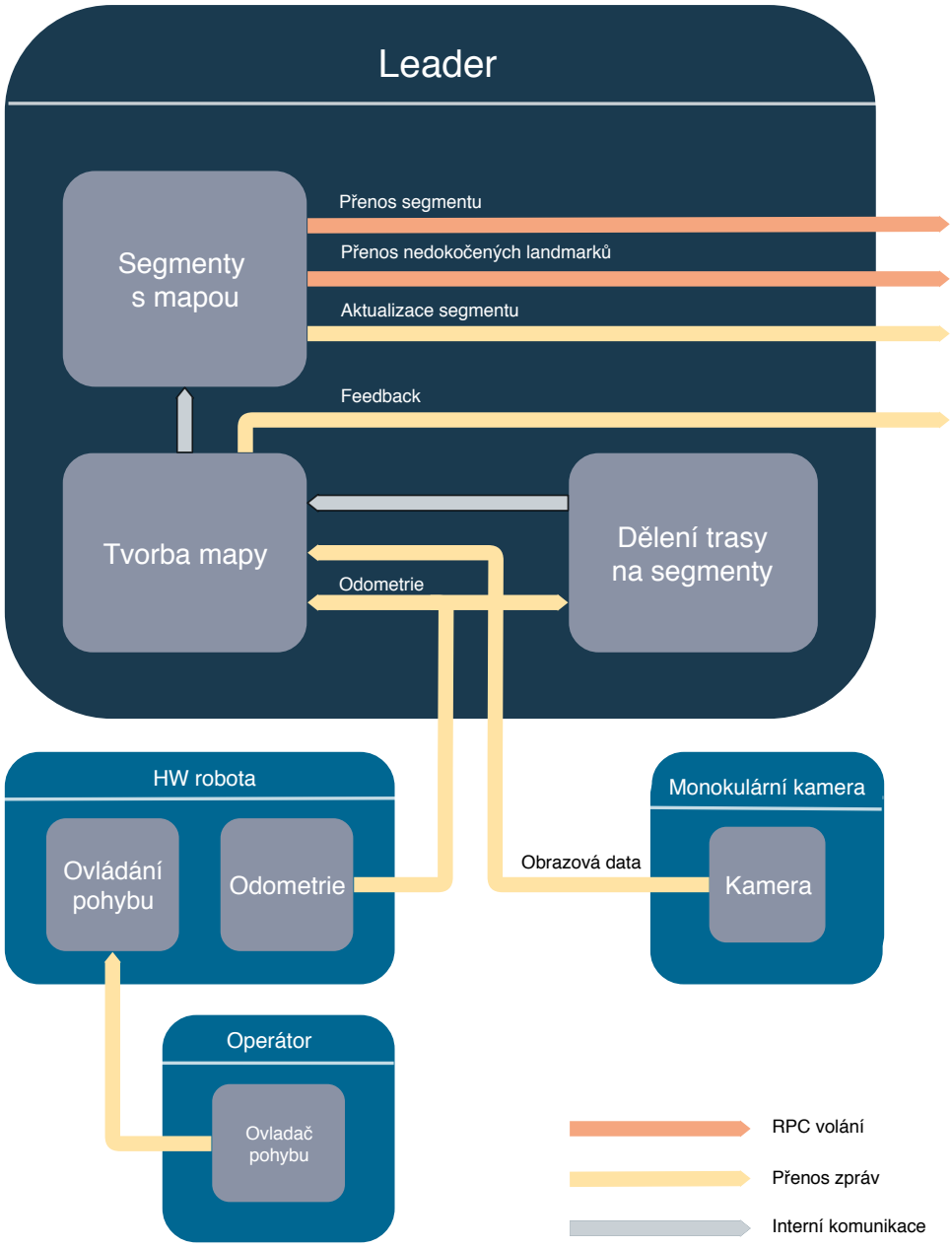
Postupně takto vytváří mapy pro další segmenty dokud není program operátorem ukončen.

Operátor řídící vedoucího robota ovládá pouze pohyb robota. Robot autonomně vytváří mapu a dělí trasu na jednotlivé segmenty.

Schéma procesů a komunikace s ostatními uzly systému je znázorněna na obrázku 4.2.

4.3.2 Sledující robot (*follower*)

Program sledujícího robota se při spuštění přihlásí k odběru zpráv odometrie robota, odometrie sousedního robota a obrazových dat z kamery. Zároveň uveřejní službu pro zasílání zpráv ovládání dopředné a úhlové rychlosti robota a zpětné vazby pro dohledovou aplikaci.



Obrázek 4.2: Schéma procesů a komunikace vedoucího vozidla s ostanímí uzly v systému ROS.

Po spuštění uzlu se sledujícím robotem robot nejdříve vyčká na první zprávy odometrie své a sousedního robota, protože bez nich by nemohl zahájit navigaci. Tím je také zajištěno, že uzly robotů jsou v systému ROS již inicializovány.

Sledující robot zjišťuje pomocí zpráv odometrie vzdálenost k sousednímu robotu, který jede v konvoji před ním, a udržuje si od sousedního robota odstup. Tím je zabráněno vzniku kolizí robotů a zároveň umožňuje vedoucímu robotu vytvořit dostatečně rozsáhlou mapu, podle které se sledující robot naviguje. Vzdálenost odstupu lze nastavit přes *Parameter Server* ROSu. Příliš malý odstup by mohl vést k tomu, že podstatnou část výhledu kamery zaplní sousední robot. To by mohlo zapříčinit detekci falešných landmarků na pohybujícím se sousedním robotovi před ním a ve výsledku ke zhoršení nebo úplnému znemožnění navigace.

Protože vedoucí i sledující roboti společně tvoří konvoj a startují současně, mají sledující roboti jiné startovací polohy. Na začátku se tedy sledující robot musí přemístit na startovací pozici vedoucího robota bez použití navigace dle mapy, která pro tento úsek trasy neexistuje. Protože jsou roboti do konvoje řazeny v malých rozestupech v řádu jednotek metrů, chyba odometrie není velká a neurčitost v poloze je dále kompenzována metodou navigace.

Po dosažení startovací pozice prostřednictvím RPC volání získá od vedoucího robota informace pro první segment trasy včetně mapy a přihlásí se k odběru zpráv s aktualizacemi dat pro daný segment. Jednou s informací segmentu je počáteční orientace robota, podle které se robot natočí, aby kamera směřovala stejným směrem jako při vytváření mapy.

Poté je již robot připraven k navigaci podle obrazových *features* a následuje trasu vedoucího robota. Zpracováním obrazových dat z kamery detekuje a extrahuje *features*, které porovná s *features* uložené v mapě a určí odchylku od sledované trasy vyjádřenou úhlem posunutí. Úhel posunutí je jednoduchým regulátorem převeden na úhlovou rychlost. Dopředná rychlost je regulována P regulátorem tak, aby saturace motorů robota byla plynulá a nedocházelo ke skokovým změnám v rychlosti, které by mohly způsobovat další nepřesnosti v odometrii robota. Řízení rychlostí regulátory je umístěno v hlavní programové smyčce, která se opakuje na frekvenci 100 Hz. Tím je zaručena dostatečná plynulost řízení bez ohledu na frekvenci zpracování obrazu jednotlivých snímků z kamery, která bývá zpravidla řádově menší a může kolísat. Robot také kontroluje zbývající délku do konce segmentu a pokud se k jeho konci blíží, snižuje postupně rychlost pro regulátor (resp. nastavuje *setpoint* regulátoru). Tím je docíleno, že robot zastaví v blízkém okolí konce segmentu a nezvyšuje nepřesnost metody navigace. Nastavené rychlosti jsou zprávou odeslány a uzel ovládající HW robota dle ní provede požadovavou změnu rychlosti na motorech robota.

Segment trasy, který robot následuje, nemusí být v okamžiku zahájení sledování ještě vedoucím robotem ukončen. Proto vedoucí robot průběžně publikuje zprávy s aktualizacemi informací o segmentu včetně nově přidaných

nebo změněných landmarků, podle kterých sledující robot upravuje svou mapu a uraženou vzdálenost vedoucího robota.

Na druhé straně může být vedoucí robot před sledujícím robotem o několik segmentů napřed. V takovém případě se robot naviguje podle mapy, kterou získal na počátku segmentu a která je pro daný segment již hotová. Další aktualizace segmentu ignoruje.

Během cesty uraženou vzdálenost porovnává a pokud dosáhne velikosti délky segmentu, robot se zastaví a navigaci tohoto segmentu ukončí. Poté si vyžádá od vedoucího robota informace k dalšímu segmentu trasy. Po natočení robota na počáteční orientaci segmentu sledující vozidlo naviguje podél nového segmentu.

Stejným způsobem pokračuje v navigaci i v dalších segmentech trasy, dokud jsou vedoucím vozidlem k dispozici.

Schéma procesů a komunikace s ostatními uzly systému je znázorněna na obrázku 4.3.

4.4 Struktura programového kódu

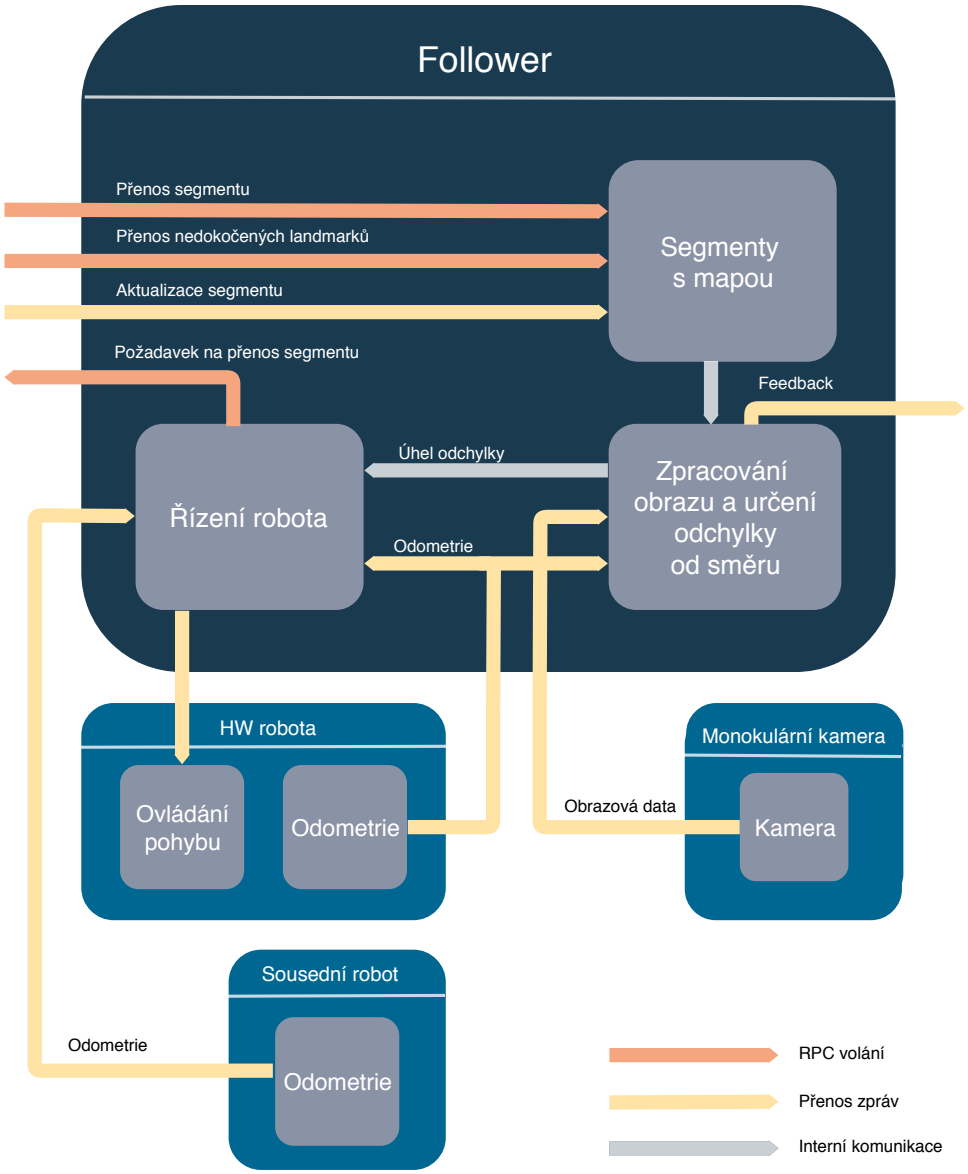
Softwarové řešení jsem implementoval v programovacím jazyku C++ v prostředí ROS s využitím knihoven OpenCV a Boost.

Pro implementaci navigační metody jsem upravil kód převzatý z ROS balíku `lama_featurenav` projektu *Large Maps Framework (LaMa)* [9] vyvinutý skupinou IMR.

Také jsem pro snažší kontrolu funkčnosti algoritmů vytvořil dohledovou aplikace, ve které lze sledovat informace o pohybujících se robotech.

Jednotlivé komponenty jsou organizovány do několika ROS balíků:

1. `fnc_base`



Obrázek 4.3: Schéma procesů a komunikace sledujícího vozidla s ostaními uzly v systému ROS.

Základní balík obsahující kód pro spuštění uzlu pro vedoucí a sledující vozidlo. Zároveň je přiložen *launch* soubor pro spuštění uzlů pro celý konvoj.

2. `fnc_gazebo`

   Balík obsahující *launch* soubory pro spuštění simulace v simulátoru Gazebo.

3. `fnc_bringup`

Balík obsahující *launch* soubory pro spuštění reálných robotů TurtleBot.

4. `fnc_dashboard`

   Balík obsahující aplikaci pro dohled probíhající navigace. V aplikaci se zobrazuje obraz z kamery vedoucího a sledujícího vozidla, základní informace o vozidlech, grafy s dopřednou a úhlovou rychlostí a grafy s detekovanými *features* a *features* se shodou pro daný úsek. Dále je k dispozici přehledová mapka o aktuální poloze robotů s vyznačenou stopou projeté trasy.

Kapitola 5

Simulace a experimenty

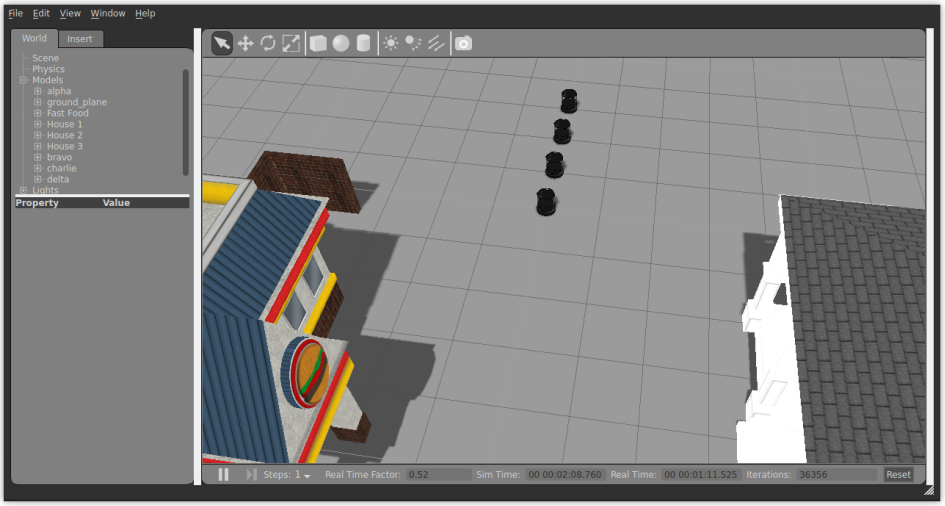
Pro ověření navrhnutého systému jsem konvoj nejprve simuloval na testovací trase v simulátoru Gazebo. Po úspěšném testu jsem systém ověřil v praxi na robotech TurtleBot2.

5.1 Simulace

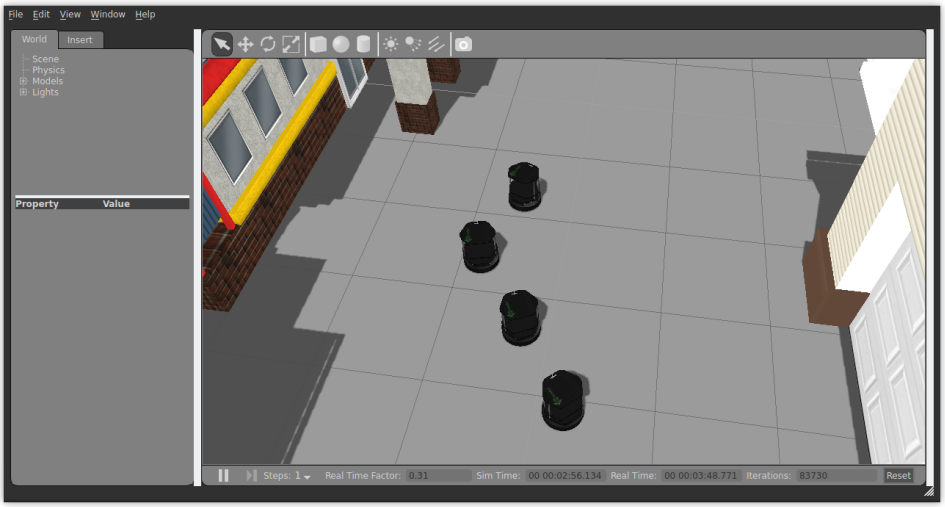
Pro účely simulace jsem vytvořil konvoj o celkem čtyřech vozidlech - jedním vedoucím robotem a třemi sledujícími. Pro snažší orientaci jsem roboty pojmenoval postupně jako *Alpha*, *Bravo*, *Charlie* a *Delta*. Na začátku simulace je vedoucí robot *Alpha* umístěn na startovní pozice a ostatní jsou umístěny v rozestupech 1 m za ním. Situace na počátku simulace je zachycena na obrázku 5.1.

Po startu jsem ovládal vedoucího robota *Alpha* a projel s ním trasy složené z několika různě dlouhých segmentů. Obrázek 5.2 zachycuje konvoj robotů v průběhu simulace.

Testoval jsem pohyb po uzavřené trase s poměrně krátkými segmenty. Vedoucího robota jsem vedl tak, aby startovní pozice prvního byla přibližně shodná se startovní pozicí posledního segmentu. Celková trasa byla dlouhá 28,67 m a skládala se z devíti segmentů. Pro vyhodnocení přesnosti metody jsem pro každý robot a pro každý segment zaznamenal pozici na začátku



Obrázek 5.1: Postavení robotů na začátku simulace v simulátoru Gazebo.



Obrázek 5.2: Pohybující se konvoj robotů v průběhu simulace v simulátoru Gazebo.

segmentu a určil vzdálenosti od pozice vedoucího robota. Výsledné hodnoty jsou zaznamenány v tabulce 5.1.

Je z ní patrné, že nezávisle na délce segmentu se odchylky pozic na konci jednotlivých segmentů pohybují na velmi malých hodnotách. I v případě maximálních zaznamenaných hodnot jde o chybu navigace v řádech centimetrů. Nejdůležitějším faktem ale je, že se chyba dle předpokladu nezvyšuje s narůstající uraženou vzdáleností.

Projetá trasa je také pro názornost zobrazena na obrázku 5.3, kde je

ID segmentu	**Délka segmentu**	**Bravo**	**Charlie**	**Delta**
0	2,942 m	0,035 m	0,035 m	0,012 m
1	2,099 m	0,028 m	0,024 m	0,066 m
2	2,402 m	0,059 m	0,073 m	0,093 m
3	3,393 m	0,023 m	0,056 m	0,114 m
4	3,741 m	0,038 m	0,081 m	0,116 m
5	1,737 m	0,046 m	0,023 m	0,050 m
6	5,907 m	0,055 m	0,024 m	0,047 m
7	2,599 m	0,039 m	0,056 m	0,008 m
8	3,004 m	0,051 m	0,063 m	0,106 m
Max. hodnota		0,059 m	0,081 m	0,116 m

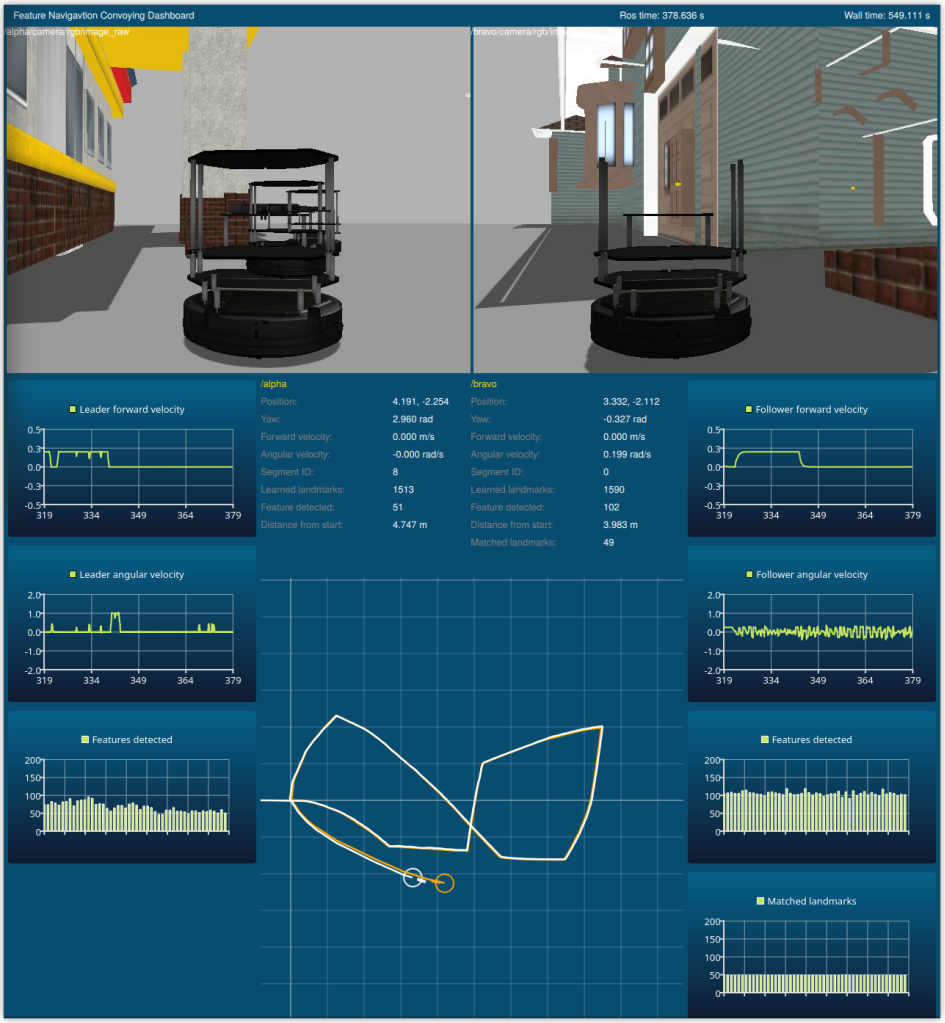
Tabulka 5.1: Tabulka se zaznamenanými vzdálenostmi pozic robotů na začátku každého segmentu od pozice vedoucího robota pro trasu s krátkými segmenty.

ID segmentu	**Délka segmentu**	**Bravo**	**Charlie**	**Delta**
0	15,827 m	0,048 m	0,076 m	0,125 m
1	13,997 m	0,007 m	0,072 m	0,083 m
2	3,080 m	0,068 m	0,184 m	0,087 m
3	3,778 m	0,053 m	0,229 m	0,065 m
Max. hodnota		0,068 m	0,229 m	0,125 m

Tabulka 5.2: Tabulka se zaznamenanými vzdálenostmi pozic robotů na začátku každého segmentu od pozice vedoucího robota pro trasu obsahující delší segmenty.

v dohledové aplikaci zachycen průběh simulace po dokončení trasy. Z pohledu kamery robota *Alpha*, kterého jsem po zastavení otočil zpět směrem k projeté trase, lze vidět postavení robotů v konvoji. Z obrázku je patrné, že v simulátoru metoda navigace pracuje velmi dobře a zobrazená stopa (sledujícího robota *Bravo*) přesně kopíruje trasu projetou vedoucím robotem. Je to dáno zejména dostatečným počtem detekovaným *features* v modelovém prostředí, které je složeno převážně z jednobarevných ploch a kontrastních textur.

Dále jsem testoval trasu složenou s delších segmentů, kdy největší z nich byl dloubý přes 15 m. Výsledné hodnoty jsou zaznamenány v tabulce 5.2. Za povšimnutí stojí vyrázně horší hodnoty robota *Charlie*, který patrně detekoval landmarky na dvou robotech jedoucích před ním a tím došlo posunu maxim v histogramu při zpracování obrazu.



Obrázek 5.3: Průběh simulace v dohledové aplikaci. Aplikace je rozdělená na dvě poloviny a přehledovou mapkou se stopami projetých tras robotů. Vlevo jsou zobrazeny údaje vedoucího robota *Alpha*, vpravo sledujícího robota *Bravo*. U obou se zobrazuje průběh dopředné a úhlové rychlosti v čase a počet detekovaných *features* při zpracování obrazu. Pro sledujícího robota je zobrazen také počet landmarků, které se pro daný úsek trasy podařilo přiřadit landmarkům z mapy.

5.2 Testovací prostředí

Testování probíhalo ve vnitřních prostorech laboratoře robotiky v budově *CIIRC* pod umělým osvětlením a na rovné ploše. Prostředí bylo přiměřeně členité, což umožňovalo dostatečnou detekci landmarků.

Pro snímání obrazu byly na robotech připevněny monochromatické mono-

kulární kamery. U metody navigace díky algoritmu použitém v metodě *FAST* 3.4 pro detekci *features* v obrazu nezáleží na tom, zda je obraz barevný nebo černobílý.

Vzhledem k povaze zasílání zpráv v systému ROS může docházet ke zpoždění nebo ztrátám odesílaných zpráv. Pokud je přijatá zpráva příliš stará s tedy již neaktuální odometrií, není tato zpráva pro navigaci použita. Aby metoda navigace správně fungovala, je třeba mít data z odometrie co nejvíce aktuální. Proto bylo potřeba na operačních systémech robotů synhronizovat čas, aby byly časové odchylky pokud možno co nejmenší.

5.3 Experimenty

Pro experimenty jsem měl k dispozici dva roboty TurtleBot2 [14] s instalovanými monokulárními monochromatickými kamerami. Sledujícího robota jsem umístil za vedoucího s odstupem 2 m.

Pro sledujícího robota jsem nastavil maximální rychlost na $0{,}15\,\mathrm{m\,s^{-1}}$, kamery snímaly obraz o frekvenci 10 HZ.

Na obrázku5.4 je zachycen vedoucí robot z pohledu kamery sleducícího robota v průběhu experimentu.

Počet detekovaných features a landmarků byl v reálném prostředí řádově nižší (v řádu jednotek až desítek), ale metodě navigace stačí pro navigaci i jediný landmark při každém zpracování obrazu

Podobně jako v simulaci se sledující robot velmi přesně držel trasy vedoucího robota.

V článku[1] se předpokládá pohyb po rovných segmentech konstantní rychlostí. Experimenty potvrdily správnost navigace také při pohybu po křivkách a s regulovanou rycholstí. Při pohybu po křivkách nesmí však úhlová rychlost robota přesáhnout jistou mez závislou na parametrech robota, jejíž překročení by mohlo zamezit identifikaci landmarků v obraze a následné tvorbě mapy z nich.

Obrázek 5.4: Sledující robot následuje trasu projetou vedoucím robotem.

Kapitola 6

Závěr

Cílem této práce bylo implementovat a ověřit v praxi pohyb robotů v konvoji založený na navigační metodě SurfNav vyvinutou skupinou *IMR*. V konvoji je vedoucí robot ovládán operátorem a během jízdy terénem si autonomně vytváří mapu okolního prostředí a trasu dělí na jednotlivé úseky. Sledující vozidla projetou trasu sledují již plně autonomně.

Jedním z hlavních požadavků na navigaci vozidel v konvoji bylo, aby metoda navigace byla přesná i na delších vzdálenostech, přestože v navigaci použitá odometrie získaná senzory je zatížena aditivní chybou. Ukázalo, že metoda SurfNav chybu odometrie dokáže plně kompenzovat. Implementoval jsem regulátory pro řízení rychlosti, aby pohyb robotů v konvoji byl plynulý a nedocházelo k dalšímu zvyšování nepřesnosti odometrie.

Dalším požadavkem bylo, aby se celý konvoj pohyboval současně a zpracování obrazu probíhalo na jednotlivých robotech v reálném čase i na méně výkonném hardwaru. Zvolil jsem pro řešení úlohy vhodné metody a algoritmy počítačového vidění, které toto umožnují při zachování kvality navigace.

Program konvoje jsem implementoval v systému ROS s použítím open-source knihoven a vytvořil pro něj dohledovou aplikaci.

Během simulace i experimentů se ukázalo, že v při pohybu konvoje v prostředí, které splňuje požadavky kladené metodou, je navigace vozidel v konvoji velmi přesná. A to i v případě, kdy je obraz okolního prostředí výrazně pozměněn záběrem na sousední vozidlo.

Příloha A

Literatura

[1] Krajník, T., Faigl, J., Vonaásek, V., Košnar, K., Kulich, M. and Přeučil, L., *Simple yet stable bearing-only*, Journal of Field Robotics, doi: 10.1002/rob.20354, 2010.

[2] Thrun S., Burgard W., Fox D., *Probabilistic Robotics,* , The MIT Press, 2005.

[3] Saska, M. Kasl, Z. Přeučil, L., *Motion Planning and Control of Formations of Micro Aerial Vehicles*, Proceedings of The 19th World Congress of the International Federation of Automatic Control. Pretoria: IFAC, 2014, p. 1228-1233. ISSN 1474-6670. ISBN 978-3-902823-62-5., 2014

[4] Nitsche, M., Pire, T., Krajník, T., Kulich, M., Mejail, M., *Monte Carlo Localization for teach-and- epeat feature-based navigation*, Advances in Autonomous Robotics Systems. Heidelberg:Springer, 2014, p. 13-24. ISSN 0302-9743. ISBN 978-3-319-10400-3., 2014

[5] Mázl, R., Kulich, M., Přeučil, L., *Statistical and Feature-Based Methods for Mobile Robot Position Localization*, Database and Expert Systems Applications. Berlin: Springer, 2001, vol. 1, p. 518-526. ISBN 3-540-42527-6., 2001

[6] The Robot Operating System, *oficiální stránky projektu*, `http://www.ros.org/`, 2018.

[7] Gazebo, *oficiální stránky projektu*, `http://gazebosim.org/`, 2018.

[8] OpenCV, *oficiální stránky projektu*, `https://opencv.org/`, 2018.

[9] *General Documentation of the Large Maps Framework*, `http://wiki.ros.org/LargeMapsFramework`, 2018.

[10] Rosten, E., Drummond T. *Fusing points and lines for high performance tracking*, IEEE International Conference on Computer Vision. 2: 1508–1511. doi:10.1109/ICCV.2005.104, 2005.

[11] Rosten, E., Drummond T. *Machine learning for high-speed corner detection*, European Conference on Computer Vision. 1: 430–443. doi:10.1007/11744023_34, 2006.

[12] Calonder, M., Lepetit, V, Strecha, C., and Fua, P., *BRIEF: Binary Robust Independ Elementary Features*, Computer Vision - ECCV (pp.778-792) 2010.

[13] Muja, M., Lowe, D, *FLANN - Fast Library for Approximate Nearest Neighbors, User Manual*, `https://www.cs.ubc.ca/research/flann/uploads/FLANN/flann_manual-1.8.4.pdf`,

[14] *Specification for TurtleBot Compatible Platforms*, `http://www.ros.org/reps/rep-0119.html`, 2011

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Vojtěch N o v á č e k

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Konvoj na bázi kamery

Pokyny pro vypracování:

Mezistupněm mezi současnou technologií a plně autonomními vozidly představuje tzv. *platooning* řešení, kde je člověkem řízeno vedoucí vozidlo, zatímco další autonomně řízená vozidla ho následují stejnou cestou. Cílem této práce je zkombinovat ověřené technologie skupiny IMR tak, aby bylo možné realizovat *platooning* pomocí kamery jako jediného senzoru.

Úkoly:

- vývoj algoritmu pro sledování vedoucího vozidla založeného na algoritmu SurfNav vyvinutého skupinou IMR, který mapuje okolní prostředí extrahováním deskriptorů z obrazového toku,
- vylepšení předchozího algoritmu doplněného o informace o hloubce jednotlivých *features*,
- experimenty na skutečných robotech.

Seznam odborné literatury:

[1] Saska, M. - Kasl, Z. - Přeučil, L., Motion Planning and Control of Formations of Micro Aerial Vehicles, In: Proceedings of The 19th World Congress of the International Federation of Automatic Control. Pretoria: IFAC, 2014, p. 1228-1233. ISSN 1474-6670. ISBN 978-3-902823-62-5., 2014
[2] Nitsche, M. - Pire, T. - Krajník, T. - Kulich, M. - Mejail, M., Monte Carlo Localization for teach-and-repeat feature-based navigation, In: Advances in Autonomous Robotics Systems. Heidelberg: Springer, 2014, p. 13-24. ISSN 0302-9743. ISBN 978-3-319-10400-3., 2014
[3] Krajník, T. - Faigl, J. - Vonásek, V. - Košnar, K. - Kulich, M. - Přeučil, L., Simple Yet Stable Bearing-only Navigation, In: Journal of Field Robotics 27, p. 511-533, 2010
[4] Mázl, R. - Kulich, M. - Přeučil, L., Statistical and Feature-Based Methods for Mobile Robot Position Localization, In: Database and Expert Systems Applications. Berlin: Springer, 2001, vol. 1, p. 518-526. ISBN 3-540-42527-6., 2001

Vedoucí bakalářské práce: Dr. Gaël Pierre Marie Ecorchard

Platnost zadání: do konce letního semestru 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 12. 1. 2017