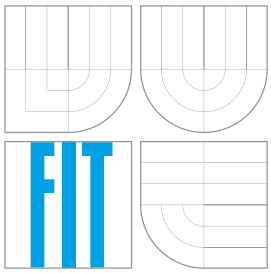
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

LOKALIZACE ROBOTA POMOCÍ OPENSTREET MAPY

ROBOT LOCALIZATION USING OPENSTREET MAP

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ZDENĚK RAJNOCH

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAROSLAV ROZMAN, Ph.D.

BRNO 2016

Zadání diplomové práce/16184/2015/xrajno04

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2015/2016

Zadání diplomové práce

Řešitel: **Rajnoch Zdeněk, Bc.**
Obor: Inteligentní systémy
Téma: **Lokalizace robota pomocí OpenStreet mapy**
Robot Localization Using OpenStreet Map
Kategorie: Umělá inteligence

Pokyny:

1. Prostudujte problematiku lokalizace robotů pomocí částicových filtrů. Nastudujte knihovnu Robotického Operačního Systému (ROS).
2. Navrhněte program, který lokalizuje robota pomocí OpenStreet mapy na základě dat ze senzorů (Lidar, Kinect, 3D laser, kompas, enkodéry, IMU). Přibližné místo polohy robota bude dáno poskytnutým výřezem z OpenStreet mapy.
3. Navržený program implementujte a zaintegrujte jej do ROSu.
4. Program otestujte na reálném robotovi v reálném prostředí. Výsledek zhodnoťte a navrhněte případná vylepšení.

Literatura:

- Thrun S.: Probabilistic Robotics, MIT Press, 2005, ISBN 0262201623

Při obhajobě semestrální části projektu je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Rozman Jaroslav, Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 25. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Tato práce se zabývá lokalizací robota v rámci výřezu z OpenStreet mapy. Pro rekonstrukci uražené trajektorie jsou použity senzory robota IMU, odometrie a kompas, ta je poté porovnána s referenční GPS trajektorií. Následná lokalizace probíhá pomocí rozšířené Monte Carlo lokalizace a shlukování. Jako implementační jazyk je použit C++ a middleware ROS.

Abstract

Goal of this thesis is localization of mobile robot in OpenStreet map segment. Robot IMU, odometry and compass sensors are used for trajectory reconstruction, which is compared to reference GPS trajectory. Extended Monte Carlo localization and clusterization are used for robot localization. Software is implemented in C++ with ROS middleware.

Klíčová slova

Lokalizace, robot, ROS, OpenStreet mapy, GPS, lidar, gyroskop, akcelerometr, kamera, IMU, Kinect, kompas, odometrie, částicový filtr, Kalmanův filtr, Monte Carlo lokalizace

Keywords

Localization, robot, ROS, OpenStreet maps, GPS, lidar, gyroscope, accelerometer, IMU, camera, Kinect, odometry, compass, particle filter, Kalman filter, Monte Carlo localization

Citace

RAJNOCH, Zdeněk. *Lokalizace robota pomocí OpenStreet mapy.* Brno, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Rozman Jaroslav.

Lokalizace robota pomocí OpenStreet mapy

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Rozmana, Ph.D. Další informace mi poskytl Ing. Zdeněk Materna. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Zdeněk Rajnoch
24. května 2016

Poděkování

Děkuji panu Ing. Jaroslavu Rozmanovi, Ph.D., za jeho odborné vedení při vypracování této práce.

Obsah

Kapitola 1

Úvod

Tato práce se zabývá návrhem a implementací programu pro lokalizaci robota v rámci výřezu z OpenStreet mapy. Pro orientaci robota v prostoru se využívá senzorů, které jsou upevněny na robota, a dokáží zachytit různé aspekty jeho pohybu. Každý ze senzorů funguje na jiném principu a má své silné a slabé stránky pro použití k tomuto účelu. Některé ze senzorů, které lze pro lokalizaci robota využít, jsou popsány v kapitole 2. Kapitola 3 popisuje lokalizační algoritmy využívané pro lokalizaci robota, kapitola 4 popisuje robotický operační systém – ROS, který je využit jako implementační platforma a spravuje komunikaci mezi senzory a vytvořeným programem. Kapitola 5 popisuje strukturu a formát mapových podkladů OpenStreet map. Kapitoly 6 a 7 obsahují návrh a implementaci vytvořeného systému a jeho následné testování a zhodnocení.

Pro implementaci práce byl použit školní robot Toad[1] vyobrazený na obrázku 1.1. Tento robot disponuje dvěma moduly GPS, IMU, kompasem, odometrií, dvěma lidary a senzorem Kinect 360. Senzory využité v této práci jsou IMU, kompas, odometrie a GPS pro udávání referenční pozice.

Ze semestrálního projektu byly přejaty teoretické kapitoly 2, 3, 4 a 5, přičemž došlo v průběhu řešení diplomové práce k jejich dalším úpravám.

Obrázek 1.1: Robot Toad.

[1] http://www.fit.vutbr.cz/research/groups/robo/eqp.php.cs

Kapitola 2

Senzory

V této kapitole jsou popsány některé ze senzorů, které je možné použít pro lokalizaci robota. Některé ze senzorů jsou schopné globální lokalizace jako například GPS, jiné jsou vhodné pro lokální lokalizaci. Každý ze senzorů má určité silné a slabé stránky – je třeba brát v úvahu jejich přesnost, dosah, spotřebu elektrické energie, váhu, velikost i cenu. Pro spolehlivou lokalizaci je pak vhodné slučovat data z více senzorů dohromady.

2.1 Lidar

Tato kapitola je založena na [3].

Lidar je zařízení schopné s velkou přesností měřit vzdálenost objektů. Má podobný princip funkce jako radar, ale místo rádiových signálů využívá laserové paprsky – zařízení vyšle světelný paprsek a na základě rozdílu času mezi vysláním paprsku a přijetím odraženého paprsku vypočte vzdálenost objektu. Výslednou polohu detekovaného bodu pak lze získat za pomocí této vzdálenosti, znalosti, pod kterým úhlem byla vzdálenost měřena a pozice lidaru. Výhodou lidaru pro měření vzdálenosti je jeho vysoká přesnost.

(a) (b)

Obrázek 2.1: Ukázka 2D lidaru SICK LMS100 2.1(a) a vizualizace jeho scanu okolí 2.1(b) z projektu Roboauto 2014[1]

[1] http://www.roboauto.cz

Lidarů existují dvě základní varianty – 2D lidar a 3D laser. 2D lidar obsahuje jeden vysílač a přijímač světelného paprsku na otočné hlavě a je tak schopen snímání po čáře. 3D lasery mají vysílačů a přijímačů více a jsou tak schopné vytvoření kompletní 3D hloubkové mapy okolí. Lidarů je značné množství typů, které se od sebe liší frekvencí snímání, dosahem laserového paprsku, pořizovací cenou, vahou a dalšími parametry.

2.2 GPS

Tato kapitola je založena na [2].

Globální poziční systém, nebo též NAVSTAR, je lokalizační systém umožňující zjistit polohu zařízení téměř kdekoli na povrchu Země a lze ho tak použít ke globální lokalizaci. Původně se jednalo o vojenský systém americké armády, později však byl zpřístupněn i pro veřejnost. Později byly vytvořeny další systémy schopné globální lokalizace jako například evropský projekt Galileo, nebo Ruský GLONASS. Systém GPS se skládá ze tří základních částí – kosmického, kontrolního a uživatelského segmentu.

Kosmický segment se skládá z 24 satelitů obíhajících Zemi, přičemž 21 z nich je aktivních a 3 jsou záložní. V současnosti je však používáno 31 satelitů, protože byly vypouštěny nové satelity bez odstavení těch starých. Každý ze satelitů obsahuje vysoce přesné atomové hodiny a čas mezi satelity je udržován synchronizovaný. Každý ze satelitů vysílá svou aktuální polohu a čas.

Kontrolní segment se skládá z množství pozemních stanic, přičemž primární stanice se nachází v Colorado Springs. Stanice monitorují signál z družic a posílají jim zpět korekce.

Uživatelský segment tvoří přijímače GPS signálu. Přijímače přijímají signál ze satelitů a na základě jejich signálu vypočtou svou pozici. Přijímače jsou pouze pasivní – satelity nemají informace o přijímači ani o jeho poloze. Přijímače mají také své hodiny, které musí být synchronizovány se satelity kvůli určení pozice. Signál je přijímán z většího množství satelitů a přijímače porovnávají svůj aktuální čas s časy ze zpráv zaslaných satelity, což umožňuje výpočet přibližné vzdálenosti od každého ze satelitů a na základě toho odvozenou polohu. Na získání 2D pozice je zapotřebí minimálně tří satelitů, při čtyřech a více satelitech lze pak vypočíst také nadmořskou výšku. Čím více satelitů je aktuálně viditelných, tím přesnější je vypočtená pozice. Současně lze přijímat signál až z 12 satelitů, pokud se nacházíte na místě s dobrým výhledem a signál neblokují překážky – například stromy nebo budovy. Uvnitř budov je signál příliš slabý. GPS přijímače jsou také schopné vypočíst přesnost s jakou je pozice určena. Přesnost systému se může pohybovat od jednotek centimetrů až po desítky metrů. GSP používá systém souřadnic WGS84.

2.3 Kompas

Tato kapitola je založena na [14].

Kompas je senzorem využívajícím pro svou činnost magnetického pole Země. Geomagnetické pole je generováno tekutým vnějším jádrem Země a je značně symetrické – lze na něj nahlížet jako na obrovský magnet v centru Země, kde magnetické pole vychází z jižní hemisféry a vrací se na severní. Je třeba mít na paměti, že magnetický střed je od osy zemské rotace mírně odchýlen. Lokální magnetické pole však mohou ovlivňovat i další faktory – záření z ionosféry, magnetické rudy v půdě nebo i elektrické vedení. Ty jej mohou značně vychýlit.

Na každém místě na Zemi lze lokální magnetické pole popsat vektorem, který má svůj směr a velikost a zařadit jej do souřadného systému, kde x ukazuje na geografický sever, z ukazuje ke středu gravitační síly a y určíme pravidlem pravé ruky. Úhel mezi geografický severem a vertikální složkou vektoru se nazývá deklinace, úhel mezi vertikální složkou vektoru a východem pak inklinace.

Kompasů je značné množství druhů, jejich nejstarším a nejjednodušším zástupcem je kompas jehličkový. Jehličkový kompas je pouze zmagnetizovaná jehla posazená na stojánek s velmi nízkým třením, a může se tak volně pohybovat v horizontálním i vertikálním směru. Kompasy také bývají často napuštěny tekutinou jako například směs vody a alkoholu nebo čistý olej, aby se zabránilo samovolné oscilaci jehly. Působením okolního magnetického pole se pak jehla natočí paralelně k magnetickým siločárám.

Dalším a modernějším zástupcem je kompas elektronický, který eliminuje značné množství okolností způsobujících výchylky jehly, jako je například změna rychlosti pohybu a umožňují digitální čtení hodnot. Většina elektronických kompasů je založena na magnetometru, který měří intenzitu jedné nebo více složek zemského magnetického pole. Základní elektronické kompasy se skládají ze dvou magnetometrů svírajících pravý úhel a připevněných na vodorovné podložce – každý z magnetometrů pak měří jednu ze složek magnetického pole, jeden horizontální po ose x a druhý vertikální po ose y. Vyjádříme-li pak intenzity magnetického pole z těchto senzorů jako B_x a B_y, pak výsledný úhel natočení oproti magnetickému severu můžeme získat vztahem

$$\psi = arctan(B_y/B_x)$$

2.4 Gyroskop

Tato kapitola je založena na [7].

Gyroskopy jsou senzory schopné zjistit úhlovou výchylku v určité ose. Existují 3 základní typy gyroskopů – s otočným závažím, optické a vibrační.

2.4.1 Gyroskop s otočným závažím

Gyroskopy s otočným závažím pracují na principu zachování úhlového momentu a fungují na principu druhého Newtonova zákona. Pokud je otočné závaží upevněno v otočném pouzdře tak, aby mohlo volně rotovat podél obou os kolmých ke sledované rotační ose, pak zůstane zarovnáno vzhledem k inerciálnímu prostoru, i když se pouzdro otáčí. Natočení gyroskopu ve sledované ose pak způsobí vychýlení otočného závaží. Na základě velikosti tohoto vychýlení je možné zjistit hledaný úhel.

2.4.2 Optický gyroskop

Optické gyroskopy pracují na tom principu, že v daném materiálu cestuje světlo konstantní rychlostí. Pokud je světlo vysláno oběma směry do nerotující uzavřené smyčky složené ze zrcadel nebo optického kabelu, uražená vzdálenost obou paprsků bude stejná. Pokud však bude cesta paprsku natočená okolo osy kolmé ke své rovině, potom se budou odrazné plochy vzdalovat pro paprsek letící ve směru rotace a přibližovat pro paprsek ve směru opačném. Tomuto jevu se říká Sagnacův efekt. Změřením rozdílu délky cest obou paprsků lze určit změnu úhlu.

Laserový gyroskop využívá uzavřenou trubici s minimálně třemi úseky naplněnou helioneonovým plynem. V jednom z rohů trubice je umístěno polopropustné zrcadlo a detektor paprsků a v ostatních se nachází vysoce reflektivní zrcadla. Dále je zde použita anoda a katoda pro vytvoření velkého rozdílu potenciálu napříč plynem a pro vytvoření elektrického pole. Atomy plynu mohou absorbovat energii z elektrického pole a tím se vybudit. Ve vybuzeném stavu jsou však nestálé a nakonec se vrátí do svého stálého stavu, přičemž však přebytečnou energii uvolní ve formě fotonu. Atomy mohou být vybuzeny také jinými fotony, přičemž z nich uvolněný foton bude mít stejnou vlnovou délku, fázi a trajektorii – tento jev se nazývá koherence. Pokud nedochází k žádné rotaci, mají fotony z obou směrů stejnou vlnovou délku. V případě rotace se prodlouží délka trubice ve směru rotace a trubice v opačném směru smrští, což způsobí zvýšení vlnové délky a snížení frekvence ve směru pohybu. Opačný jev nastane v opačném směru. Detektor za polopropustným zrcadlem je schopný tyto změny vyhodnotit a určit z nich natočení ve směru sledované osy.

Gyroskopy s optickým kabelem využívají širokopásmového světelného zdroje, jehož světlo je rozděleno do dvou stejných paprsků a vysláno z obou směrů do cívky namotané z optického kabelu. Po průchodu cívkou jsou tyto dva paprsky opět složeny na detektoru a je pozorována jejich interference. Pokud dojde k rotaci kolem sledované osy, změní se fáze paprsků. Na základě rozdílu fází je pak možné zjistit úhel natočení.

2.4.3 Vibrační gyroskop

Vibrační gyroskopy používají ke své činnosti vibrační prvek, například strunu nebo rameno, který je udržován v jednoduchém harmonickém pohybu. Všechny pracují na stejném principu – detekují Coriolisovu sílu vibračního prvku, když dojde k natočení gyroskopu. Coriolisovo zrychlení vybuzuje jednoduchý harmonický pohyb na ose kolmé k vibračnímu prvku a amplituda tohoto pohybu pak odpovídá úhlové rychlosti. V praxi bývá pohyb vibračního prvku omezen pouze na jednu sledovanou osu kolmou k vybuzovači vibrací tak, že pouze rotace na sledované vstupní ose způsobí oscilaci na ose výstupní, která je kolmá ke vstupu i vybuzovači vibrací.

2.5 Akcelerometr

Tato kapitola je založena na [7].

Akcelerometry se používají pro zjištění změny rychlosti pohybu v rámci jedné osy. Jednoduchý akcelerometr je například závaží zavěšené mezi dvě pružiny ve směru snímané osy. Pokud dojde ke zrychlení nebo zpomalení pohybu, závaží bude pokračovat s původní rychlostí natahující jednu strunu a stlačující druhou, dokud nedojde opět k vyrovnání rychlostí závaží a objektu, na kterém je akcelerometr připevněn. Výslednou změnu rychlosti pak lze zjistit na základě vzdálenosti, o kterou se závaží posunulo. Akcelerometry ve směru osy z pak využívají pouze jedné struny a zemské gravitace.

2.5.1 Kyvadlové akcelerometry

Kyvadlové akcelerometry využívají závaží zavěšeného na kyvadlové ručičce. Tento způsob umožňuje závaží volný pohyb v rámci snímané osy a zároveň je ono závaží pevně zajištěno ve zbylých směrech. K přenosu vychýlení závaží je využito jedné nebo dvou strun přenášejících pohyb kyvadla. Tento design je velmi jednoduchý, má však tři základní nedostatky – rozlišovací schopnosti snímacího zařízení jsou relativně omezené, síla přenášená strunou je

pouze aproximací lineární funkce a projevuje se zde hystereze a nelinearita. Na zavěšené závaží pak působí gravitační síla, což způsobuje další nelinearitu. K vyřešení těchto problému se používá uzavřené smyčky – je využíváno točivého momentu k udržení ručičky kyvadla v konstantní pozici bez ohledu na sílu, která na akcelerometr působí. Tento mechanismus detekuje odchylky od rovnovážné pozice kyvadla a vrací jej do ní.

2.5.2 Rezonanční akcelerometry

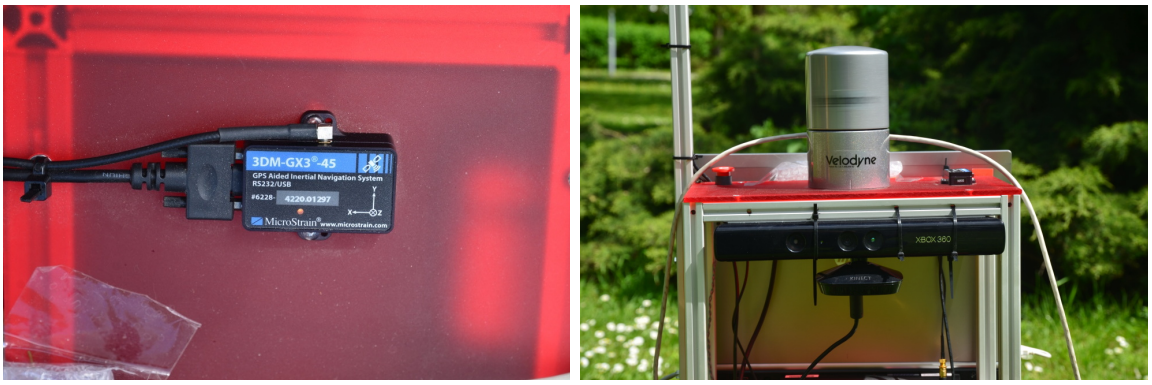
Rezonanční akcelerometry používají také závaží a kyvadlovou ručičku jako kompasy kyvadlové, avšak závaží je udržováno v rovnováze vibračním ramenem, které výrazně omezuje jeho pohyblivost. Když na závaží působí síla, rameno odtlačuje nebo přitahuje závaží. Rameno je udržováno na své rezonanční frekvenci – když je stlačeno, rezonanční frekvence se sníží, pokud je roztaženo, tak se naopak zvýší. Zrychlení se poté vypočte na základě této rezonanční frekvence. Přesnost akcelerometru lze dále zvýšit použitím dvou protilehlých ramen – když je první stlačeno, druhé se roztáhne, a naopak. Dále mohou působit na jedno nebo i více závaží.

2.6 IMU

Tato kapitola je založena na [7].

IMU neboli inerciální měřící jednotka se skládá z množství akcelerometrů a gyroskopů, přičemž akcelerometry měří sílu pohybu a gyroskopy jeho úhel. Většina IMU používá kombinaci tří akcelerometrů a tří gyroskopů – každý pro jednu z os x, y a z pro získání tří-dimenzionálního měření – využívá se tak jako navigační senzor schopný sledování pohybu zařízení ve 3D prostoru. IMU také obsahují jednoduché procesory pro převod jednotek jednotlivých senzorů, kompenzaci jejich chyb a kontroly hraničních hodnot, čímž se detekuje selhání senzoru a provádí také vzorkování výstupních hodnot. Výstupem IMU jsou změny síly a úhlu v jednotlivých osách. Inerciální senzory vykazují konstantní chybovost, která může být laboratorně zjištěna a uložena do paměti pro úpravu výstupních hodnot senzoru. Tyto chyby jsou teplotně závislé, a proto je měření prováděno pro větší množství teplot a IMU jsou vybaveny teploměrem pro zjištění nejvhodnějších kalibračních hodnot. Vzhledem k tomu, že teplota nemusí být na všech senzorech konstantní, je u výkonnějších IMU navíc implementována kontrola teploty jednotky. Dalším zdrojem chyb IMU je vzdálenost jednotlivých senzorů – pro zjištění přesného pohybu jednoho bodu v prostoru by musely být všechny senzory také v jednom jediném bodě. Velikost této chyby lze ovšem také vypočítat a opravit procesorem IMU.

V praxi se IMU dělí podle způsobu použití, a tím tedy požadované přesnosti a ceny jednotky. Nejpřesnější jednotky jsou používány pro námořní navigaci a mají odchylku menší než 1.8 kilometru za den. Velmi přesné jednotky jsou také používány v letectví s přesností 1.5 kilometru během první hodiny letu. Méně přesné jednotky se pak používají v automobilovém průmyslu, například pro řízení airbagů nebo v robotice.



(a) (b)

Obrázek 2.2: Senzory Microstrain 3DM-GX3-45 obsahující IMU, GPS a elektronické kompasy 2.2(a) a Kinect 360 s lidarem velodine 2.2(b) upevněné na robotovy Toad.

2.7 Kamera

Kamera je senzorem, který dokáže zachytit obraz okolí podobně jako lidské oko – je schopná zachytit paprsky světla odražené od snímaných objektů a na jejich základě vytvořit obraz. Základ kamery tvoří senzor snímající tyto paprsky a objektiv, který tyto paprsky na senzor směřuje. Kamery se liší značným množstvím parametrů ovlivňujících její výkon a schopnosti – například modul pro automatické zaostření na sledovaný objekt nebo senzor pro zjištění správného vyvážení bílé barvy.

Senzory se liší svou velikostí, rozlišením, citlivostí na světlo, dynamickým rozsahem, schopností podání barev, snímkovací frekvencí, minimálním a maximálním časem snímání a výrobní technologií, kdy nejčastěji bývají používány CMOS a CCD. V případě objektivů pak minimální a maximální ohniskovou vzdáleností, schopností propouštět odražené světlo, jejich ostrostí a minimální zaostřovací vzdáleností, a s tím související schopností snímat malé předměty.

V robotice se kamery často využívají k rozpoznávání a klasifikaci nejrůznějších objektů, které slouží k rozpoznání okolí a volbě vhodné akce. Dále umožňují například sledování bodů v obraze, na základě čehož je možné spočítat trajektorii robota nebo sledovaného objektu.

2.8 Kinect

Tato kapitola je založena na [25] a [24].

Kinect je senzor integrující do jednoho zařízení kameru, mikrofony, infračervený vysílač a infračervenou kameru. Původně byl firmou Microsoft vyvinut jako náhrada herního ovladače, kdy místo mechanického ovládání byly snímány pohyby uživatele a jeho gesta. Díky svým schopnostem, nízké ceně a později i volně dostupnému SDK[1] se však brzy začal používat také v robotice.

Kromě snímání obrazu vestavěnou kamerou je Kinect schopen pomocí infračerveného vysílače a infračervené kamery vytvořit hloubkovou mapu snímaného okolí. Infračervený paprsek z vysílače je rozdělen na několik paprsků tvořící konstantní vzor. Tento vzor je

[1] https://dev.windows.com/en-us/kinect

poté snímán infračervenou kamerou a porovnává s referenčním vzorem pro zjištění hloubky jednotlivých bodů.

Kinectu v současné době existují dvě verze – starší Kinect 360 a novější Kinect one. Jejich parametry jsou uvedeny v tabulce 2.1.

Parametr	**Kinect 360**	**Kinect one**
Rozlišení RGB kamery	640x480 px	1920x1080 px
Snímkovací frekvence kamery	30 fps	30 fps
Rozlišení infračervené kamery	320x240 px	512x424 px
Minimální změřitelná hloubka	40cm	50cm
Maximální změřitelná hloubka	~4.5m	~4.5m
Horizontální úhel snímání	57 stupňů	70 stupňů
Vertikální úhel snímání	43 stupňů	60 stupňů

Tabulka 2.1: Srovnání parametrů senzorů Kinect 360 a Kinect one [24].

2.9 Odometrie

Tato kapitola je založena na [16].

Odometrické senzory slouží k měření uražené vzdálenosti za použití měření úhlové rychlosti otáčení motoru. V případě použití více než jednoho motoru jsou pak schopné měřit i natočení robota na základě rozdílu rychlosti těchto motorů. Při pohybu robota je nutné počítat se šumem, který senzory produkují, podkluzováním kol robota v terénu i ne zcela přesným poloměrem a rozestupem kol robota, ze kterých výpočet rychlosti vychází. Použití odometrie je dobré především při použití na krátkých vzdálenostech, s rostoucí vzdáleností dochází k akumulaci chyby měření.

2.9.1 Typy odometrických senzorů

Jednofázové enkodéry jsou umístěné nejčastěji na hřídeli motoru nebo kolech robota. Otáčení této součásti při pohybu generuje stavy 0 nebo 1. Jednoduchou variantou takového enkodéru je připevnění děrovaného disku na otáčivou součást a detekování situace, kdy senzor prochází nad dírou.

Čtyřfázové enkodéry jsou vylepšením jednofázových enkodérů. Využívají dvou jednofázových enkodérů, které produkují vlnové délky posunuté o 90 stupňů – pokud se robot pohybuje jedním směrem, signál A předchází signál B, pokud se pohybuje druhým směrem, předcházejí se naopak.

Tachometry jsou senzory připevněné k hlavní hřídeli motoru. Fungují jako generátory energie, přičemž výstupní napětí odpovídá úhlové rychlosti motoru.

Dalším ze způsobů je měření elektromagnetického pole. Přímé měření elektromagnetického pole spočívá v krátkodobém odpojení napájení motoru a následném měření napětí produkovaného motorem. Nepřímé měření elektromagnetického pole spočívá v umístění malého rezistoru mezi motor a uzemnění. S pomocí Ohmova zákona je možné na něm vypočítat napětí.

Kapitola 3

Lokalizační algoritmy

Tato kapitola je založena na [20].

V této kapitole jsou popsány jednotlivé algoritmy použité pro lokalizaci robota. Kalmanův filtr je použit pro filtraci dat ze vstupních senzorů, Monte Carlo lokalizace je pak použita k samotné lokalizaci v mapě. Lokalizaci robota tvoří tři základní problémy – nalezení pozice robota, sledování pozice a problém uneseného robota, kdy již byla pozice robota nalezena, ale došlo k její náhlé změně.

3.1 Kalmanův filtr

Kalmanův filtr je jednou z implementací Bayesovského filtru a používá se pro filtraci a predikci v lineárních systémech. Implementuje výpočet odhadu ve spojitých stavech, v diskrétních a hybridních jej nelze využít. Filtr počítá odhad $bel(x_t)$ v určitém čase t, který je pro tento čas reprezentována střední hodnotou μ_t a kovariancí Σ_t.

Algoritmus 1: KALMAN_FILTER

Input: $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$
Output: μ_t, Σ_t

1: $\overline{\mu_t} = A_t \mu_{t-1} + B_t u_t$
2: $\overline{\Sigma_t} = A_t \Sigma_{t-1} A_t^T + R_t$
3: $K_t = \overline{\Sigma_t} C_t^T \left(C_t \overline{\Sigma_t} C_t^T + Q_t \right)^{-1}$
4: $\mu_t = \overline{\mu_t} + K_t \left(z_t - C_t \overline{\mu_t} \right)$
5: $\Sigma_t = \left(I \ - K_t C_t \right) \overline{\Sigma_t}$
6: **return** μ_t, Σ_t

Algoritmus výpočtu Kalmanova filtru je ukázán v algoritmu 1. Vstupem algoritmu je odhad v čase $t-1$, kontrolní hodnota u_t a naměřené hodnoty z_t, výstupem pak nový odhad v čase t. Na řádcích 1 a 2 je vypočítán odhad $\overline{bel}(x_t)$ pro další krok před započtením naměřených hodnot z_t. Odhad je získán interpolací kontrolní hodnoty u_t. Střední hodnota je upravena na základě deterministické verze přechodové funkce. Úprava hodnot kovariance pak počítá s faktem, že nový stav vychází ze stavu předchozího v lineární matici A_t. Tato matice je do kovariance násobena dvakrát, protože se jedná o kvadratickou matici.

Výsledný odhad se $bel(x_t)$ je následně vypočten na řádcích 3 až 5 započtením naměřených hodnot. Proměnná K_t se nazývá Kalmanův zisk a udává, do jaké míry naměřené

hodnoty ovlivňují výsledný odhad. Řádek 4 upravuje střední hodnotu na základě Kalmanova zisku a řádek 5 pak výslednou kovarianci.

3.2 Částicový filtr

Částicový filtr je neparametrickou implementací Bayesovského filtru. Pro odhad výstupních hodnot $bel(x_t)$ používá soubor náhodných částic, které jsou generovány na základě Gaussovského rozložení předpokládaného výstupu. Částice jsou zde označeny jako

$$\chi_t := x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}$$

Každá částice $x_t^{[m]}$, kde $1 \leq m \leq M$ a M udává celkový počet částic, je reprezentací stavu v čase t. V praxi se používá velké množství částic M. Základní myšlenkou tohoto přístupu je aproximovat odhad $bel(x_t)$ tímto setem částic χ_t. Pravděpodobnost, že bude částice x_t zařazena do setu χ_t by měl být úměrná výstupní hodnotě Bayesova filtru $bel(x_t)$ která ji reprezentuje:

$$x_t := x_t^{[m]} \sim p(x_t | z_{1:t}, u_{1:t})$$

Důsledkem toho je, že čím více je určitý region stavového prostoru zaplněn částicemi, tím pravděpodobněji se v tomto regionu nachází také skutečný hledaný stav.

Stejně jako v případě Kalmanovy filtrace tento algoritmus počítá odhad $bel(x_t)$ na základě odhadu $bel(x_{t-1})$ z předchozího kroku – set částic χ_t je v každém kroku aktualizován na základě setu částic χ_{t-1} z kroku předchozího.

Algoritmus 2: PARTICLE_FILTER

Input: (χ_{t-1}, u_t, z_t)
Output: χ_t

1: $\overline{\chi_t} = \chi_t = \emptyset$
2: **for** $m = 1$ to M **do**
3: *sample* $x_t^{[m]} \sim p\left(x_t | u_t, x_{t-1}^{[m]}\right)$
4: $w_t^{[m]} = p\left(z_t | x_t^{[m]}\right)$
5: $\overline{\chi_t} = \overline{\chi_t} + \langle x_t^{[m]}, w_t^{[m]} \rangle$
6: **end for**
7: **for** $m = 1$ to M **do**
8: draw m with probability $\approx w_t^{[m]}$
9: add $x_t^{[m]}$ to χ_t
10: **end for**
11: **return** χ_t

Vstup algoritmu tvoří předchozí sada částic χ_{t-1}, kontrolní hodnota u_t a naměřené hodnoty z_t. Na řádku 3 jsou tvořeny nové částice $x_t^{[m]}$ reprezentující hypotetický stav. Řádek 4 slouží k výpočtu takzvaného faktoru důležitosti částice $w_t^{[m]}$, který se používá pro započtení naměřených hodnot. Lze na něj také nahlížet jako na váhu částice, která reprezentuje pravděpodobnost, že se jedná o hledanou pozici.

Na řádcích 7 až 10 probíhá převzorkování. Algoritmus vybere ze souboru dočasných částic $\overline{\chi_t}$ M částic a zařadí je do nového setu částic, přičemž částice jsou do nového setu vybrány na základě jejich váhy w_t. Důsledkem toho se mohou částice s vysokou vahou v novém setu opakovat a částice s nízkou vahou mohou být z nového setu částic vyřazeny.

3.3 Monte Carlo lokalizace

Monte Carlo lokalizace je metoda schopná lokální i globální lokalizace robota v mapě a pro odhad pozice $bel(x_t)$ využívá sady částic. Svým principem vychází z částicového filtru s použitím pravděpodobného pohybového a váhového modelu. Základní Monte Carlo lokalizace používá pro odhad pozice $bel(x_t)$ sadu M částic $\chi_t := x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}$

Algoritmus 3: MonteCarloLocalization

Input: $(\chi_{t-1}, u_t, z_t, m)$
Output: χ_t

1: $\overline{\chi_t} = \chi_t = \emptyset$
2: **for** $m = 1$ to M **do**
3: $x_t^{[m]} = sample\_motion\_model(u_t, x_{t-1}^{[m]})$
4: $w_t^{[m]} = measurement\_model(z_t, x_t^{[m]}, m)$
5: $\overline{\chi_t} = \overline{\chi_t} + \langle x_t^{[m]}, w_t^{[m]} \rangle$
6: **end for**
7: **for** $m = 1$ to M **do**
8: draw m with probability $\approx w_t^{[m]}$
9: add $x_t^{[m]}$ to χ_t
10: **end for**
11: **return** χ_t

Na řádku 3 probíhá výpočet další předpokládané pozice částic, který používá současnou sadu částic. Na řádku 4 jsou aplikována naměřená data ze senzorů pro vyhodnocení váhy částice. Řádky 7 až 10 slouží k vytvoření nového setu částic na základě vah částic současného setu.

Počáteční odhad částic $bel(x_0)$ pro algoritmus je dán náhodným vygenerováním částic s rovnoměrným rozložením, kterým je přiřazena stejná počáteční váha.

3.3.1 Monte Carlo lokalizace s přidáním náhodných částic

Jak je z algoritmu 3 patrné, tato metoda není schopná vyřešit problém uneseného robota, kdy již došlo k jeho lokalizaci, ale pozice se poté změnila. Řešením tohoto problému je rozšíření původní metody přidáváním náhodných částic na základě sledování krátkodobých a dlouhodobých změn vah částic jak je ukázáno v algoritmu 6.

Na řádku 7 je zde počítána průměrná váha částice a řádky 9 – 10 počítají krátkodobé, respektive dlouhodobé změny, přičemž musí platit $0 \leq \alpha_{slow} \ll \alpha_{fast}$. Nakonec je na řádcích 12 – 17 na základě pravděpodobnosti rozhodnuto, zda bude přidána do setu náhodná částice nebo dojde k jejímu převzorkování jako v případě předchozí varianty algoritmu.

Algoritmus 4: AUGMENTED_MCL

Input: $(\chi_{t-1}, u_t, z_t, m)$
Output: χ_t

1: static w_{slow}, w_{fast}
2: $\overline{\chi_t} = \chi_t = \emptyset$
3: **for** $m = 1$ to M **do**
4: $x_t^{[m]} = sample\_motion\_model(u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = measurement\_model(z_t, x_t^{[m]}, m)$
6: $\overline{\chi_t} = \overline{\chi_t} + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: $w_{avg} = w_{avg} + \frac{1}{M} \cdot w_t^{[m]}$
8: **end for**
9: $w_{slow} = w_{slow} + \alpha_{slow} \cdot (w_{avg} - w_{slow})$
10: $w_{fast} = w_{fast} + \alpha_{fast} \cdot (w_{avg} - w_{fast})$
11: **for** $m = 1$ to M **do**
12: **if** $Rand(0,1) < max(0.0, 1.0 - \frac{w_{fast}}{w_{slow}})$ **then**
13: add random pose to χ_t
14: **else**
15: draw m with probability $\approx w_t^{[m]}$
16: add $x_t^{[m]}$ to χ_t
17: **end if**
18: **end for**
19: **return** χ_t

3.3.2 Pohybový a váhový model

Poloha robota v rámci 3D prostoru bývá specifikována šesti veličinami – polohou vzhledem k osám x, y a z a trojicí Eulerovských úhlů, které specifikují jeho rotace vzhledem k těmto osám. V rámci lokalizace je možné zjednodušení, kdy je pozice robota specifikovaná jako vektor $(x, y, \Theta)^T$, kde x a y značí polohu robota v souřadném systému a úhel Θ jeho natočení vůči ose x, čímž můžeme specifikovat jeho umístění v rámci mapy.

Vstupem algoritmu pohybového modelu 5 je poloha robota v čase x_{t-1} a vstup ze senzorů u_t, který udává změnu polohy, výstupem pak je nová poloha x_t. Pohyb robota reprezentovaný měřením u_t je možné reprezentovat ze tří základních částí – počáteční rotace před pohybem robota δ_{rot1}, uražená vzdálenost δ_{trans} a finální rotace robota po dokončení pohybu δ_{rot2}. Na řádcích 5 až 7 dochází k připočtení náhodných veličin, které reprezentují nepřesnost a šum obsažených v měření ze senzorů. Jako funkce *sample* zde může sloužit například normální nebo trojúhelníkové rozložení se středem 0 a rozptylem udaným výrazem v závorce. Na řádcích 9 až 11 pak dochází k výpočtu nové předpokládané pozice robota v čase t.

Algoritmus 5: SAMPLE_MOTION_MODEL_ODOMETRY

Input: (u_t, x_{t-1})
Output: x_t

$\delta_{rot1} = atan2(\overline{y}' - \overline{y}, \overline{x}' - \overline{x}) - \overline{\Theta}$
$\delta_{trans} = \sqrt{(\overline{x} - \overline{x}')^2 + (\overline{y} - \overline{y}')^2}$
$\delta_{rot2} = \overline{\Theta}' - \overline{\Theta} - \delta_{rot1}$

$\hat{\delta}_{rot1} = \delta_{rot1} - sample(\alpha_1\delta_{rot1} + \alpha_2\delta_{trans})$
$\hat{\delta}_{trans} = \delta_{trans} - sample(\alpha_3\delta_{trans} + \alpha_4(\delta_{rot1} + \delta_{rot2}))$
$\hat{\delta}_{rot2} = \delta_{rot2} - sample(\alpha_1\delta_{rot2} + \alpha_2\delta_{trans})$

$x' = x + \hat{\delta}_{trans} sin(\Theta + \hat{\delta}_{rot1})$
$y' = y + \hat{\delta}_{trans} cos(\Theta + \hat{\delta}_{rot1})$
$\Theta' = \Theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$

return $x_t = (x', y', \Theta')^T$

Pro váhový model bylo použito vlastní řešení popsané v kapitole 6.3.1.

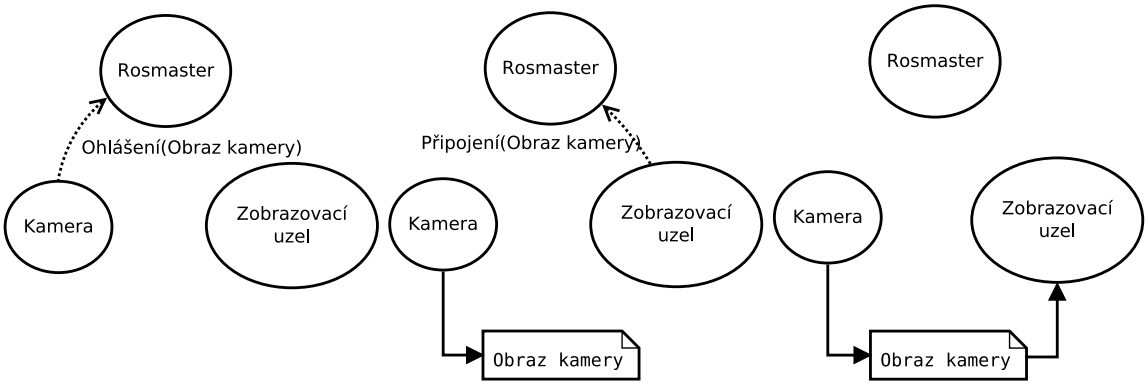
Kapitola 4

ROS

Tato kapitola je založena na [17].

ROS, neboli Robotic Operating System je souborem knihoven a aplikací nad operačním systémem, který slouží jako middleware pro aplikační systémy z oblasti robotiky. ROS také poskytuje značnou míru abstrakce. Robotické senzory mohou mít značně rozdílné komunikační protokoly, a vzhledem k rozmanitosti jednotlivých robotů vzniká problém se znovupoužitelností kódu. ROS tento problém řeší tak, že jednotlivé aplikace se skládají z uzlů, které jsou spolu schopné asynchronní komunikace pomocí mechanismu zpráv. Jádro každé aplikace tvoří soubor uzlů roscore [4], který je vyžadován pro běh ostatních uzlů. Roscore se skládá ze tří základních částí:

Rosmaster [13] slouží jako směrovač pro ostatní uzly v síti. Udržuje si seznam publikujících i přijímajících uzlů pro všechna témata i služby – publikující uzel se mu nejdříve ohlásí s názvem tématu, případně služby, který si rosmaster zaregistruje, a připojující se uzly pomocí něj zjišťují, který uzel v síti dané téma, nebo službu publikuje. Rosmaster takto slouží pouze pro vzájemnou lokalizaci uzlů, po navázání komunikace již spolu dotyčné uzly komunikují peer-to-peer.



Obrázek 4.1: Ukázka komunikace uzlů přes hlavní uzel rosmaster [13]. Ohlášení tématu 4.1(a), připojení k tématu 4.1(b) a vzájemná komunikace uzlů 4.1(c)

Parameter server [5] je další součástí roscore. Uzly jej mohou využít pro ukládání a načítání parametrů za běhu. Data v něm jsou globálně viditelná a mohou být kontrolována a měněna za běhu podle potřeby. Parameter server je implementován pomocí XMLRPC knihoven a běží v rámci Rosmaster uzlu – jeho API je dostupné přes běžné XMLRPC knihovny.

Názvy parametrů dodržuje standardní konvence ROSu – obsahují název uzlu následovaný názvem tématu a nakonec název parametru a jeho hodnotu. Tato hierarchie slouží pro předcházení kolizím názvů mezi jednotlivými uzly a umožňuje také parametry používat jednotlivě nebo ve stromové struktuře. Příklad parametrů může být následující:

```
/camera/left/name: leftcamera
/camera/left/exposure: 1
/camera/right/name: rightcamera
/camera/right/exposure: 1.1
```

Z těchto parametrů můžeme získat jednotlivé hodnoty – například pro parametr /camera/right/exposure hodnotu 1.1, nebo podstromy /camera/left, /camera/right a /camera, které obsahují strukturu se všemi parametry níže ve stromu. Datové typy parametrů mohou být 32-bitový integer, bool, řetězec, double, datum, list nebo binární data zakódovaná do base64[1].

Rosout [8] je konzolový výstup pro jednotlivé uzly programu. Má definováno 5 úrovní závažnosti [23]:

DEBUG	Pomocné zprávy pro ladění programu
INFO	Malé množství informací, které mohou být pro uživatele přínosné
WARN	Varování za běhu uzlu
ERROR	Chybové stavy uzlu, které jsou však opravitelné
FATAL	Fatální chyby uzlu

4.1 Komunikace uzlů

Jednotlivé uzly v ROSu spolu komunikují pomocí zasíláni zpráv. Zpráva je pevně typovaná datová struktura, která může obsahovat datová primitiva jako integer, bool a další, ale také pole a konstanty. ROS obsahuje značné množství již předdefinovaných zpráv – pro komunikaci se senzory, geografické zprávy a další. V případě potřeby je možné si definovat i zprávy vlastní. Uzly publikují tyto zprávy do témat, ke kterým jsou přihlášeny, a jiné uzly je z těchto témat přijímají. V rosu existují dva typy komunikace – peer-to-peer, kde uzel zasílá neustále data do tématu a přijímá je 0 až n dalších uzlů, nebo pomocí rosservice, kde jsou data konkrétnímu uzlu zaslána na vyžádání. Zatímco u témat je znám pouze název tématu, přičemž zpracování a typová kontrola zpráv jsou na přijímajících uzlech, v případě rosservice má komunikace pevně daný typ zprávy pro požadavek i odpověď. Jelikož je ROS jazykově neutrálním nástrojem, který umožňuje kromě jazyka C++ použití i dalších, jako je python a LISP, používá pro zprávy jazykově neutrální interface IDL, který slouží k popisu zpráv zasílaných mezi uzly. IDL využívá k definici zprávy velmi krátkých textových souborů, které popisují jednotlivé položky zprávy – použitý datový typ a jeho jméno. Implementačně má zpráva následující formát:

[1] https://www.base64decode.org/

```
Header header
Point32[] pts
ChannelFloat32[] chan
```

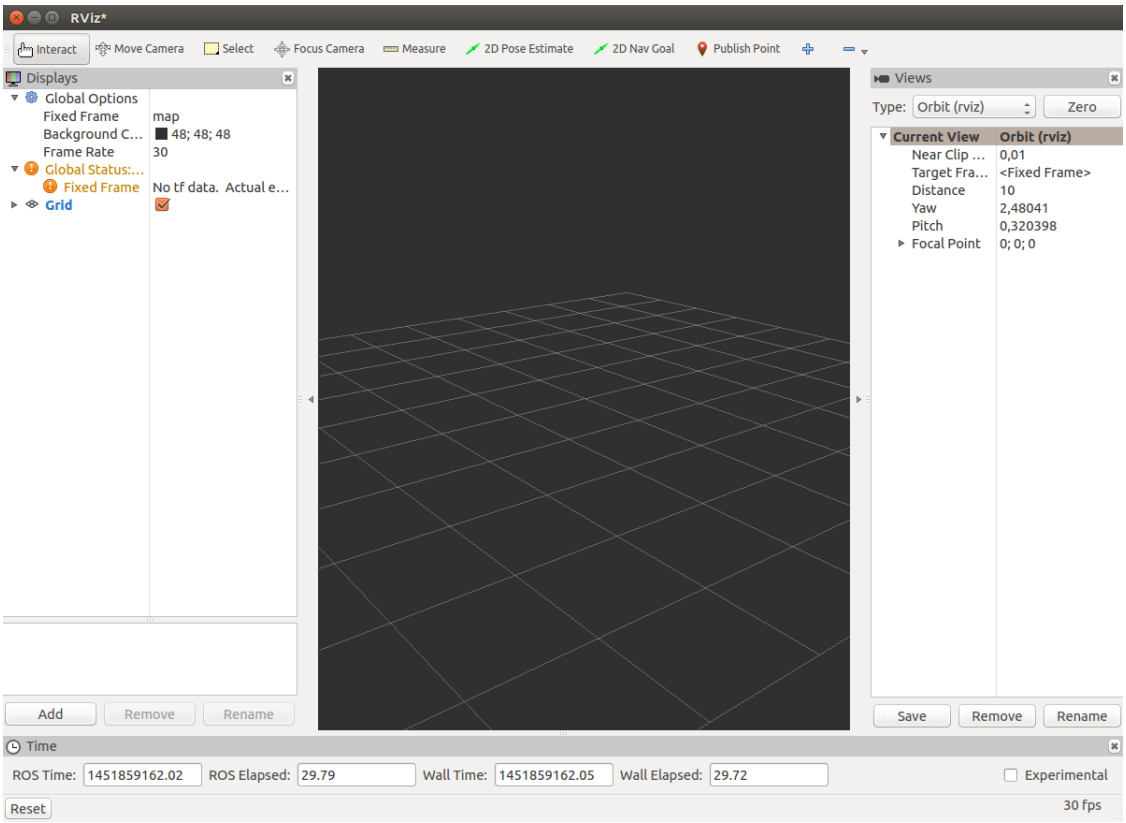
Generátory zdrojových kódů pro každý podporovaný jazyk na základě této definice vygenerují její nativní implementaci pro daný jazyk – vytvoří nativní objekt daného jazyka. ROS pak následně provádí serializaci a deserializaci objektů za běhu, jakmile jsou zprávy odesílány a přijímány mezi uzly.

4.2 Nástroje

Pro usnadnění vývoje obsahuje ROS množství podpůrných nástrojů pro správu uzlů, debugging a vizualizaci.

4.2.1 Rviz

Rviz [21] je vizualizační nástroj ROSu, který umožňuje ve zvoleném souřadném systému vizualizovat témata a transformace. Rviz lze použít pro vizualizaci obrazu z kamery, scanu z laseru, bodů, pozic v souřadném systému a dalších...



Obrázek 4.2: Ukázka nástroje Rviz.

4.2.2 Rosnode

Rosnode [9] je nástroj příkazové řádky, který zobrazuje informace o uzlech. Nástroj má následující parametry:

rosnode cleanup	Smaže registrační informace u nedosažitelných uzlů
rosnode info	Vypíše informace o uzlu
rosnode kill	Ukončí běžící uzel
rosnode list	Vypíše aktuálně běžící uzly
rosnode machine {Stroj}	Vypíše uzly běžící na zadaném stroji
rosnode machine	Vypíše aktuálně připojené stroje
rosnode ping	Otestuje konektivitu k uzlu

4.2.3 Rostopic

Rostopic [22] je nástroj příkazové řádky, který zobrazuje informace o tématech, včetně jejich publikujících a odebírajících uzlů a v nich zasílaných zprávách. Nástroj má následující parametry:

rostopic bw	Zobrazí šířku pásma využívanou tématem
rostopic echo	Vypisuje příchozí zprávy z tématu do konzole
rostopic find	Najde témata na základě typu
rostopic hz	Vypíše frekvenci zpráv v tématu
rostopic info	Vypíše informace o aktivním tématu
rostopic list	Vypíše aktivní témata
rostopic pub	Publikuje data do tématu
rostopic type	Vypíše typ tématu

4.2.4 Rosservice

Rosservice [12] je nástroj příkazové řádky pro výpis informací o servisech a umožňuje jejich volání. Nástroj má následující parametry:

rosservice call	Zavolá servis se zadanými parametry
rosservice find	Najde servisy na základě typu
rosservice info	Vypíše informace o servisu
rosservice list	Vypíše aktivní servisy
rosservice type	Vypíše typ servisu
rosservice uri	Vypíše ROSRPC adresu servisu

4.2.5 Rosbag

Rosbag [10] je nástroj příkazové řádky, který umožňuje zaznamenat vybraná témata do bag souboru a později tato data načítat a znovu publikovat. Nástroj má následující parametry:

rosbag record	Nahraje vybraná témata do bag souboru
rosbag info	Vypíše informace o bag souboru
rosbag play	Publikuje obsah jednoho nebo více bag souborů do témat
rosbag check	Zkontroluje, zda je bag soubor přehratelný na aktuálním systému nebo zda je možné na takový převést
rosbag fix	Opraví zprávy v bag souboru, aby bylo možné je přehrát na aktuálním systému
rosbag filter	Vypíše ROSRPC adresu servisu
rosbag compress	Zkomprimuje jeden nebo více bag souborů
rosbag decompress	Dekomprimuje jeden nebo více bag souborů
rosbag reindex	Přeindexuje jeden nebo více bag souborů

4.2.6 Nástroje pro překlad a správu uzlů

Pro překlad zdrojových kódů uzlů slouží nástroje rosmake [15], rosbuild [19] a catkin [6]. Rosmake jako nástroj pro překlad umožňuje překlad jednotlivých uzlů tak, aby byly splněny všechny závislosti uzlu a překlad proběhl ve správném pořadí. Rosbuild obsahuje skripty pro správu překladového systému ROSu. Catkin je nástupcem obou přechozích balíků a rozšiřuje jejich funkcionalitu. Kombinuje v sobě makra pro překladač CMake a schopnost zpracovat skripty pythonu pro rozšíření funkcionality.

Pro spuštění jednotlivých uzlů slouží nástroje rosrun [1], který umožňuje spustit jednotlivý uzel i bez znalosti jeho umístění, a roslaunch [11], který umožňuje spuštění několika uzlů – lokálních i vzdálených – přes protokol SSH a nastavování parametrů pro parameter server. Ke spuštění používá launch soubory ve formátu XML, ve kterém jsou tyto uzly a parametry specifikovány.

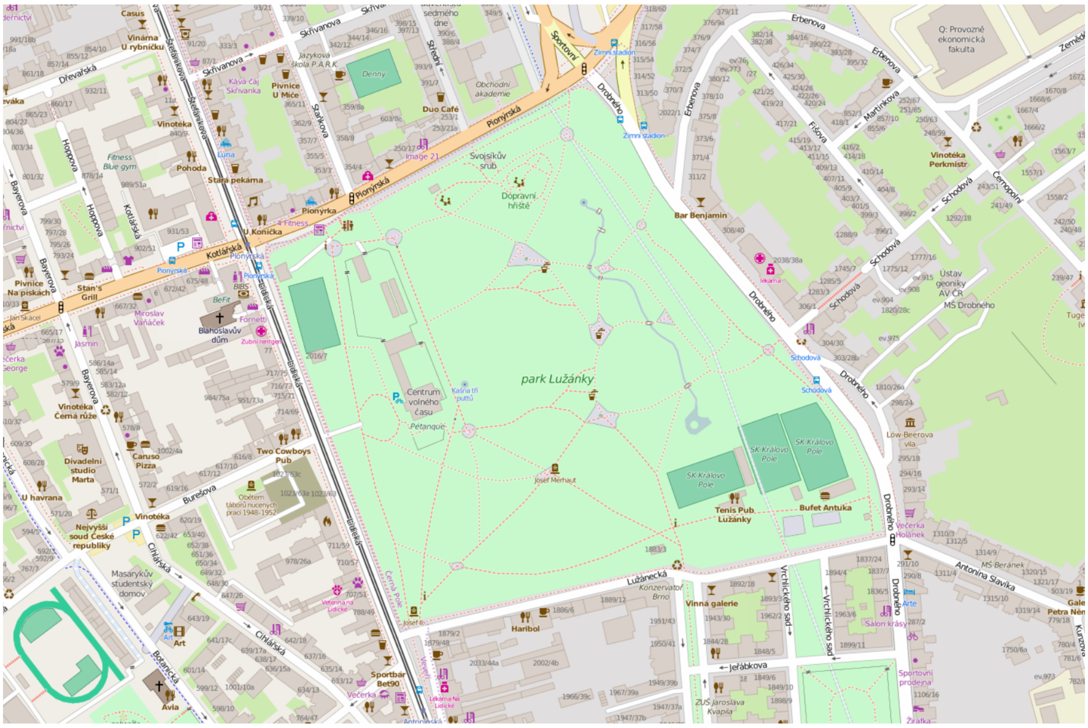
```
rosrun balík uzel
roslaunch uzly.launch
```

Kapitola 5

OpenStreet mapy

Tato kapitola je založena na [2].

OpenStreet mapy[1] jsou projektem[2] na vytvoření celosvětových mapových podkladů, které jsou volně dostupné pro každého[3] – každý může jejich obsah upravovat a nakládat s ním dle svého uvážení. Cílem projektu je vytvoření co nejpřesnější mapových podkladů na základě mapových příspěvků komunity – přispěvatelé shromažďují data ze systému GPS, na jejich základě mohou upřesňovat polohu stávajících objektů, nebo přidávat objekty nové. U objektů je uchovávána i jejich plná historie a je možné se kdykoli vracet k předchozí verzi. Mapy samotné obsahují objekty jako silnice, cesty, pěšiny, cyklostezky, budovy, vodní cesty, lesy, ale i objekty jako lavičky, telefonní budky a další. . .



Obrázek 5.1: Ukázka OpenStreet map – brněnský park Lužánky.

[1] http://www.openstreetmap.org

[2] Dokumentace k projektu je možné nalézt na adrese http://wiki.openstreetmap.org.

[3] Mapy lze exportovat na adrese http://www.openstreetmap.org

5.1 Úprava mapových podkladů

Mapové podklady v projektu OpenStreet map jsou neustále aktualizovány a doplňovány, přesto se však může stát, že některá místa v mapě jsou značně nepřesná nebo chybí úplně – v takovém případě je pak třeba mapové podklady upravit[4].

Úpravy map se provádí na základě GPS tracklogu – souboru z GPS přijímače s uloženými souřadnicemi ve formátu WGS84 a s časem jejich záznamu. V tracklogu je takto uložen set bodů, které byly zaznamenány při pohybu přijímače po cestě a zaznamenávají tak polohu cesty – jejich skutečná poloha se však může lišit až o desítky metrů v závislosti na kvalitě přijímače a síle signálu. Před nahráním je třeba tracklog překonvertovat do formátu GPX[5]. V editoru OpenStreet map, ukázaném na obrázku 5.2, je pak možné tyto body promítnout do mapy a na jejich základě upravovat polohy objektů, mazat neplatné objekty nebo přidávat objekty nové. Po dokončení úprav jsou tyto změny uloženy a během krátkého časového intervalu se promítnou v aktuálních mapových podkladech. K editaci map lze také použít značné množství externích nástrojů – například Potlatch, JOSM a Merkaartor.

Obrázek 5.2: Ukázka webového editoru OpenStreet map – brněnský park Kadetka.

5.2 Formát map

OpenStreet mapy mají vlastní datový model využívající tři základní primitiva pro popis všech objektů – uzly, cesty a vztahy mezi nimi. K dispozici je také velmi rozsáhlá možnost tagování objektů, která umožňuje jejich přesný popis. Každé z primitiv má vlastní unikátní ID, které zůstává konstantní po celou dobu existence objektu. V databázi je pak uložena každá verze každého objektu, a je tak možné na jeho základě sledovat změny objektu v čase.

[4] Úpravy se provádí například přes webové rozhraní na adrese http://www.openstreetmap.org
[5] http://www.topografix.com/gpx.asp

5.2.1 Uzly

Uzly jsou body v prostoru a jako jediné z primitiv mají informaci o své poloze. Poloha uzlu je uložena ve formátu WGS84 s přesností až na sedm desetinných míst. Uzly lze použít samostatně jako body zájmu, jako spojnici více cest nebo pro změnu směru cesty.

```
<node id="25320734" visible="true" version="3" changeset="4313202" ...
    timestamp="2010-04-03T13:51:30Z" user="hanoj" uid="4788" ...
    lat="49.2281154" lon="16.5952133"/>
```

Zde je ukázka exportovaného uzlu ve formátu OSM. Uzel obsahuje svůj jednoznačný identifikátor, informace o své viditelnosti a verzi, čas poslední změny a uživatele, který změnu provedl. Dále je zde uvedená zeměpisná šířka a délka, na které se uzel nachází.

5.2.2 Cesty

Cesty jsou reprezentovány uspořádaným seznamem uzlů. Mají mnoho různých typů a lze je využít například jako silnice, cesty, cyklostezky nebo vodní toky. V případě uzavření grafu uzlů do kruhu je také možné jich využít k ohraničení oblastí. Cesty jsou vázány na konkrétní uzly, nikoli jejich pozici, a tak změna uzlu nijak neovlivní cestu. Cesta musí obsahovat minimálně dva uzly a maximálně může obsahovat až 2000 uzlů. Jeden uzel může patřit do 0 až n cest. K protnutí cest dochází také v rámci uzlu, pokud jdou cesty přes sebe bez spojení uzlem, nejsou logicky propojeny.

```
<way id="30359940" visible="true" version="5" changeset="34141080" ...
    timestamp="2015-09-20T13:52:04Z" user="Shorty124" uid="3251760">
  <nd ref="334877282"/>
  <nd ref="334877284"/>
  <nd ref="334877286"/>
  <nd ref="3751575079"/>
  <nd ref="1771876101"/>
  <nd ref="1771876098"/>
  <nd ref="1771876094"/>
  <nd ref="1771876093"/>
  <tag k="highway" v="path"/>
  <tag k="surface" v="unpaved"/>
</way>
```

Cesta, stejně jako uzel, obsahuje svůj jednoznačný identifikátor, informace o své viditelnosti a verzi, čas poslední změny a uživatele, který změnu provedl. Dále jsou zde odkazy na identifikátory uzlů, kterými cesta prochází, a tagy pro určení typu cesty – v tomto případě se jedná o nezpevněnou cestu.

5.2.3 Vztahy

Vztah je seznamem primitiv, včetně možných vnořených vztahů. Umožňují modelovat složitější vztahy mezi objekty, které nelze uzly a cestami vytvořit. Využívají se například pro složitější, velmi dlouhé cesty nebo omezení – například zákaz odbočení. Nemusí být v mapě viditelné, ale lze je použít například i při hledání cesty mapou.

```
<relation id="1799710" visible="true" version="1" changeset="9604499" ...
    timestamp="2011-10-19T22:20:06Z" user="Washek" uid="245252">
  <member type="way" ref="133911337" role="outer"/>
  <member type="way" ref="44685378" role="outer"/>
  <tag k="name" v="L"/>
  <tag k="type" v="multipolygon"/>
</relation>
```

U definice vztahu jsou použity stejné atributy jako v případě uzlu a cesty – tedy jeho jednoznačný identifikátor, informace o jeho viditelnosti a verzi, čas poslední změny a uživateli, který změnu provedl. Tento konkrétní vztah slouží ke spojení dvou cest.

5.2.4 Tagování

Výše zmíněná primitiva dokáží fyzicky modelovat libovolný objekt, chybí jim však sémantický význam. Tento nedostatek řeší právě tagy. Tag je složen ze dvojice klíč-hodnota, přičemž obě části jsou textové řetězce o délce až 255 znaků a každé z primitiv může mít přiřazené libovolné množství tagů. Tagy lze používat jak předdefinované, tak vlastní, které je možné zpracovávat například ve vlastní aplikaci.

```
<tag k="footway" v="sidewalk"/>
<tag k="highway" v="footway"/>
```

Zde je ukázka tagů cesty pro pěšinu.

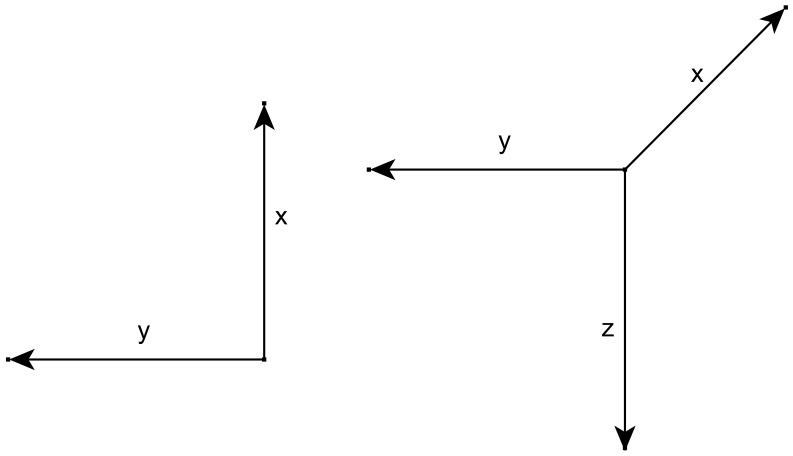
Kapitola 6

Návrh a implementace

V této kapitole je popsán návrh systému založený na teoretických znalostech z minulých kapitol a jeho implementace v ROSu. Robot Toad[1], pro který je tento systém navrhnut, má k dispozici jednotku IMU Microstrain 3DM-GX3-45, podvozek Pioneer 3-AT se snímáním odometrických dat, Kinect 360, a dvojici lidarů – Velodyne HD Lidar HDL-32E společně s lidarem Hokuyo UTM-30LX. Systém navržený v této práci využívá ke své činnosti výše zmíněné IMU a odometrická data z podvozku robota.

6.1 Zpracování dat ze senzorů

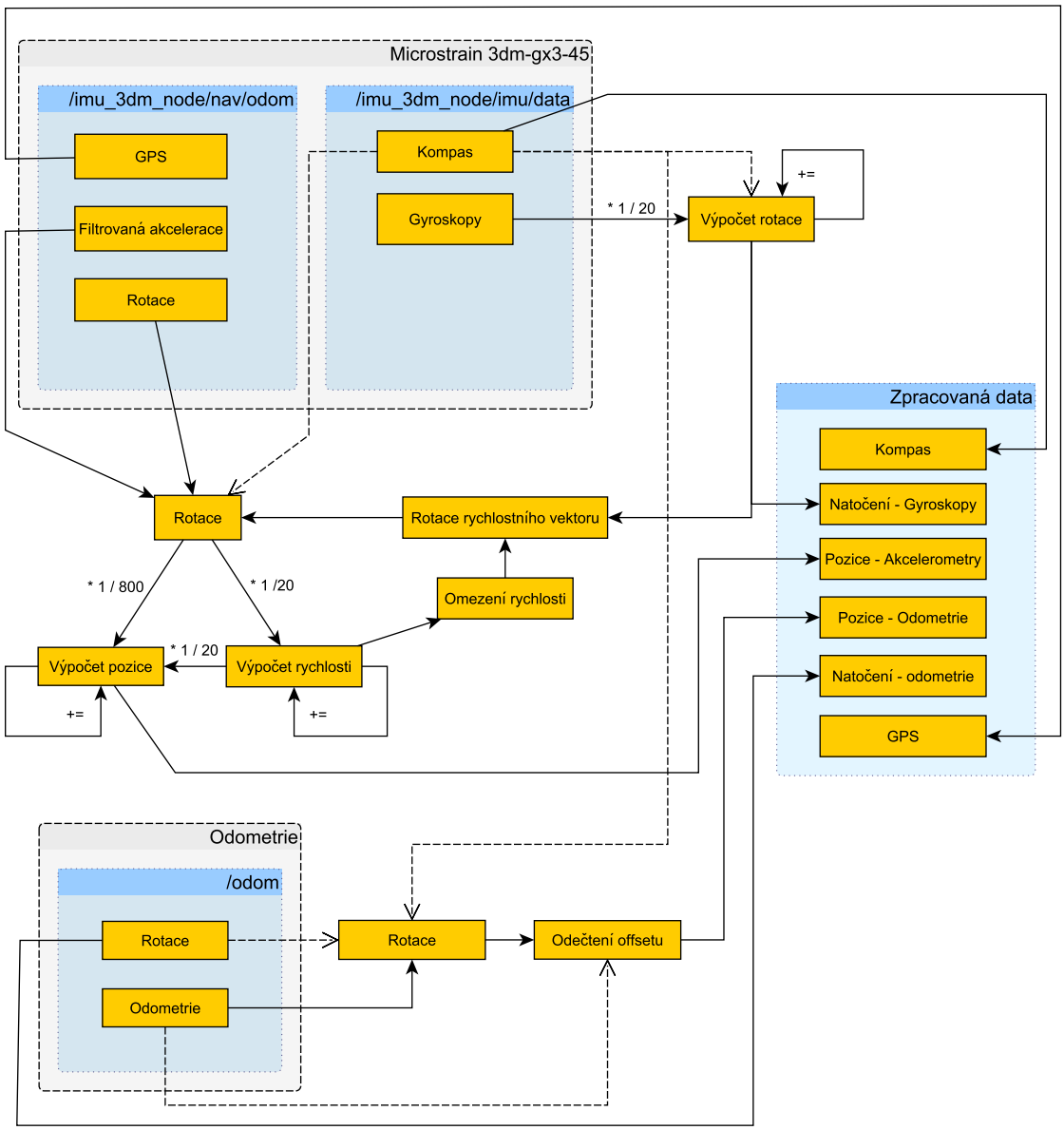
Každý z použitých senzorů disponuje ovladačem pro systém ROS, a jsou tak díky němu odstíněny jako samostatné uzly publikující snímaná data, která tento program odebírá. Každý ze senzorů i témat má jinou frekvenci zasílání zpráv, a proto je třeba řešit také synchronizaci mezi tématy i senzory. Na obrázku 6.2 je zobrazen návrh systému pro načítání dat ze senzorů, na obrázku 6.1 jsou pak souřadné systémy senzorů.



(a) (b)

Obrázek 6.1: Souřadný systém odometrie 6.1(a) a jednotky Microstrain 3DM-GX3-45 6.1(b)

[1] http://www.fit.vutbr.cz/research/groups/robo/eqp.php.cs



Obrázek 6.2: Návrh zpracování dat ze senzorů

6.1.1 Odometrie

Odometrická data jsou zasílána v tématu */odom* rychlostí 10 zpráv za vteřinu. Větev zprávy *pose* : *pose* obsahuje v sekci *position* aktuální předpokládanou polohu robota v metrech, přičemž osa x míří směrem vpřed od počátečního natočení robota, osa y pak doleva. V sekci *orintation* se nachází předpokládané natočení robota oproti jeho původní orientaci. Druhá větev zprávy *twist* : *twist* udává v sekci *linear* lineární akceleraci robota v ms^{-2} a úhlovou rychlost udávanou v radiánech za vteřinu v sekci *angular*. Zpráva dále obsahuje kovariační matice udávající, s jakou jistotou jsou tato data přesná. Příklad zprávy ROSu pro odometrickou zprávu je uveden v ukázce 6.1.

Ukázka kódu 6.1: Zpráva odometrie ROSu.

```
pose:
  pose:
    position:
      x: -8.4
      y: -102.961
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: 0.999961923064
      w: 0.00872653549837
  covariance: [0.001, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.001, 0.0, 0.0, 0.0, ...
      0.0, 0.0, 0.0, 1000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000.0, ...
      0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
      1000.0]
twist:
  twist:
    linear:
      x: 0.006
      y: 0.0
      z: 0.0
    angular:
      x: 0.0
      y: 0.0
      z: -0.0034
  covariance: [0.001, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.001, 0.0, 0.0, 0.0, ...
      0.0, 0.0, 0.0, 1000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000.0, ...
      0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
      1000.0]
```

6.1.2 IMU Microstrain 3DM-GX3-45

Microstrain 3DM-GX3-45 je jednotkou kombinující v sobě trojici akcelerometrů, gyroskopů a magnetometrů – pokaždé pro snímání ve všech osách x, y a z a inerciální GPS. Jednotka také obsahuje integrovaný Kalmanův filtr pro filtraci a spojování dat z jednotlivých senzorů. Pro příjem dat jsou zde využita dvě témata ROSu. Prvním je */imu_3dm_node/imu/data*, které obsahuje čistá data z akcelerometrů, gyroskopů a magnetometrů s připočtenou teplotní kompenzací. Ukázka těchto dat se nachází v kódu 6.2.

Data z kompasů jsou obsažena v sekci *orientation* a jsou ve formě quaternionu. Pro získání natočení robota je třeba provést přepočet tohoto quaternionu na Eulerovské úhly, čímž získáme rotaci robota okolo všech tří os. Natočení kolem osy z pak udává orientaci vůči magnetickému severu – pro získání skutečného severu je ještě třeba přičíst magnetickou deklinaci.

V sekci *angular_velocity* se nachází data z gyroskopů v radiánech za vteřinu vztažená vůči aktuální poloze senzoru. Pro získání rotace je třeba tato data vydělit frekvencí zasílání zpráv. Poslední sekce *linear_acceleration* obsahuje data z akcelerometrů jednotky.

Ukázka kódu 6.2: Zpráva dat IMU ROSu.

```
orientation:
  x: -0.0325742740231
  y: 6.39307411072e-05
  z: -0.99292337459
  w: 0.114201947347
orientation_covariance: [0.25, 0.0, 0.0, 0.0, 0.25, 0.0, 0.0, 0.0, 0.25]
angular_velocity:
  x: 0.0282230116427
  y: -0.11199798435
  z: 0.0455621816218
angular_velocity_covariance: [0.25, 0.0, 0.0, 0.0, 0.25, 0.0, 0.0, 0.0, 0.25]
linear_acceleration:
  x: -0.0221250746399
  y: 0.130992591381
  z: 0.968341529369
linear_acceleration_covariance: [0.0625, 0.0, 0.0, 0.0, 0.0625, 0.0, 0.0, ...
    0.0, 0.0625]
```

Ukázka kódu 6.3: Zpráva lokalizace a filtrovaných dat IMU ROSu.

```
pose:
  pose:
    position:
      x: 616266.205667
      y: 5454027.6817
      z: 0.0
    orientation:
      x: -5.01382693842e-05
      y: -0.0239728295282
      z: 0.61245489534
      w: 0.790141950604
  covariance: [0.11993535608053207, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
      0.12077923864126205, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
      0.21349361538887024, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
      0.03105948120355606, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0313834473490715, ...
      0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.026535050943493843]
twist:
  twist:
    linear:
      x: 0.516748845577
      y: 0.987007021904
      z: -2.18570756912
    angular:
      x: 0.061269171536
      y: 0.106291562319
      z: 0.120577916503
  covariance: [0.0625, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0625, 0.0, 0.0, 0.0, ...
      0.0, 0.0, 0.0, 0.0625, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.25, 0.0, 0.0, ...
      0.0, 0.0, 0.0, 0.0, 0.25, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.25]
```

Vzhledem k vysoké chybě těchto dat však je pro získávání dat z akcelerometrů využito druhého tématu */imu_3dm_node/nav/odom*, jehož ukázka se nachází v kódu 6.3. Data se zde nachází v sekci *twist* : *twist* : *linear*. Tato data jsou filtrována interním Kalmanovým filtrem v jednotce Microstrain 3DM-GX3-45. Naměřené hodnoty jsou orientovány vzhledem k orientačnímu quaternionu v sekci *pose* : *pose* : *orientation* určujícímu rotaci vzhledem

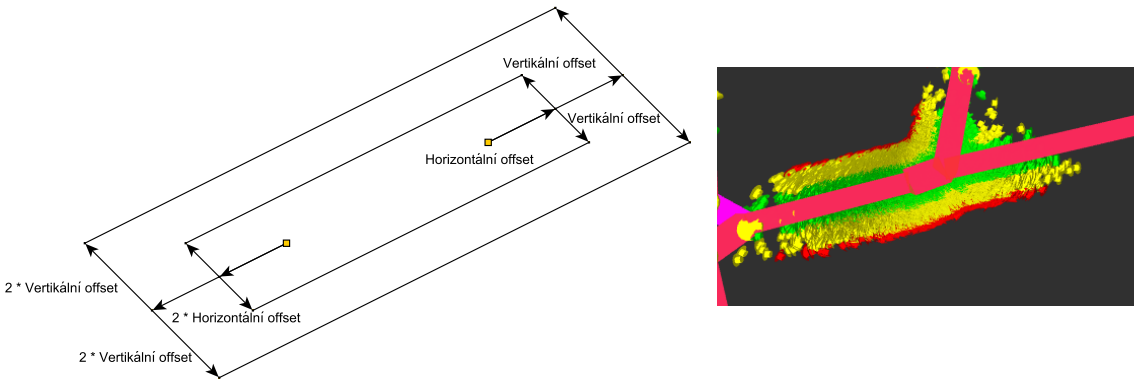
ke geografickému severu, přičemž osa x udává zrychlení robota na sever a osa y zrychlení na východ a naměřené hodnoty jsou udávány v ms^{-2}. Toto natočení však vykazuje také vysokou chybovost, a tudíž třeba provést zpětnou rotaci[2] [18] a následně provést rotaci na základě natočení udávaného kompasem. Další úpravou dat je rotace akceleračních a rychlostních vektorů na základě odchylky naměřených hodnot od natočení udávaného gyroskopy a omezení maximální povolené rychlosti.

Téma dále obsahuje v sekci *pose* : *pose* : *position* pozici robota v UTM souřadnicích která je dána fúzí všech senzorů v jednotce Microstrain 3DM-GX3-45 a je využita jako referenční pro porovnávání ujeté trajektorie a lokalizaci. Poslední sekce *twist* : *twist* : *angular* obsahuje filtrovaná data z akcelerometrů v radiánech za vteřinu vzhledem k orientačnímu quaternionu *pose* : *pose* : *orientation* a jsou nevyužita.

6.2 Zpracování mapových podkladů

Pro načtení OpenStreet map je použit balík ROSu *osm_cartography*[3], který načte mapový soubor definovaný v souboru *test_osm/tests/osm_test.launch*. Pro vizualizaci také třeba upravit offset pro statickou transformaci v souboru *osm_cartography/launch/viz_osm.launch* tak, aby odpovídal UTM souřadnicím mapy.

Při zpracování publikovaných map jsou zpracovávány pouze úseky cest – *route_segments*, které představují úsečky, ze kterých se jednotlivé cesty skládají. Pro každý ze segmentů je vytvořena třída obsahující informace o svých mezních hodnotách a se dvěma dvojicemi úseček reprezentujícími danou cestu. Tyto úsečky jsou vytvořeny na základě krajních bodů v *route_segments*[*i*] a dvojice offsetů pro šířku cesty a prodloužení cesty za okrajové body. Na obrázku 6.3(a) je znázorněna interní struktura cesty v programu, na obrázku 6.3(b) je pak zobrazena vizualizace cest v nástroji Rviz.



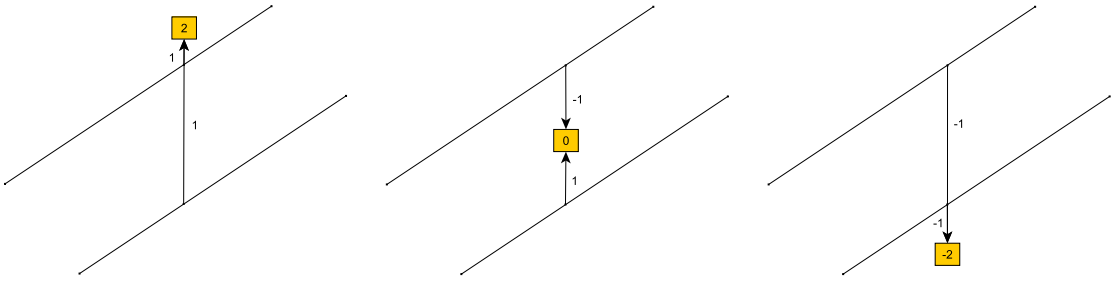
(a) (b)

Obrázek 6.3: Ukázka reprezentace úseku cesty – 6.3(a) představuje reprezentaci úseku cesty v programu, 6.3(b) znázorňuje vizualizaci cest v nástroji Rviz – zelené částice se nacházejí na cestě, žluté v jejím okolí a červené jsou mimo cestu, v další iteraci algoritmu budou vyřazeny.

[2] http://files.microstrain.com/3DM-GX3-45-Data-Communications-Protocol.pdf str. 90
[3] http://wiki.ros.org/osm_cartography

První pár úseček reprezentuje samotnou cestu a používá se pro vygenerování počátečních bodů pro Monte Carlo lokalizaci a pro kontrolu, zda se robot při pohybu nalézá na cestě. Druhý pár úseček potom reprezentuje okolí cesty, kdy je brána v úvahu určitá nepřesnost v mapových podkladech a pohybu robota – V okolí cesty dochází ve váhovém modelu Monte Carlo lokalizace k penalizaci částice, nenahlíží na ni však jako na ztracenou.

Pro zjištění, zda je částice na cestě, případně v jejím okolí, se nejprve zkontroluje, zda částice leží mezi hraničními hodnotami daného úseku cesty, poté se použije její souřadnice na pomaleji rostoucí ose a vypočte se, ve kterém bodě na rychleji rostoucí ose protne danou úsečku. Pokud je souřadnice pomaleji rostoucí osy bodu nad danou úsečkou, dostane hodnocení 1, pokud je naopak pod úsečkou, dostane hodnocení -1. Tyto hodnoty jsou následně sečteny, a pokud je výsledné hodnocení nulové, bod leží na daném úseku cesty, pokud není, opakuje se stejný postup pro dvojici úseček reprezentující okolí cesty. Na základě druhého vyhodnocení je pak rozhodnuto, zda bod leží v okolí cesty nebo se na daném segmentu cesty nenachází. Tento systém je vyobrazen na obrázku 6.4.



(a) (b) (c)

Obrázek 6.4: Systém zjištění, zda se bod nachází na cestě. Bod se nachází nad oběma úsečkami 6.4(a) – váha 2, bod se nachází mezi úsečkami 6.4(b) – váha 0, bod se nachází pod oběma úsečkami 6.4(c) – váha -2

6.3 Monte Carlo lokalizace

Metoda Monte Carlo lokalizace byla implementována v rozšířené verzi schopné řešit všechny tři lokalizační problémy. Vzhledem k velikosti mapy a relativně malému prostoru, kde se v ní může robot nacházet, však bylo nutné metodu upravit tak, aby zachovávala již vygenerované částice s dostatečnou vahou. Bez této úpravy dochází hned v prvotních iteracích algoritmu k tvorbě náhodných shluků náhodně rozložených po mapě, přičemž ve značné většině případů tyto shluky neodpovídají hledané pozici robota. Zachovávání částic tento problém efektivně řeší a shluky jsou vytvářeny až v pozdějších iteracích, kdy byly částice vyřazeny na základě váhového modelu – opustily cestu nebo jejich rotace dlouhodobě neodpovídala rotaci robota udávané kompasem.

Pro efektivní výběr částic mezi řádky 12 až 19 jsou částice seřazeny optimální řadící metodou na základě vah a každé z částic je přiřazena jako parametr suma její váhy a váhy všech částic nad ní v seřazené posloupnosti. Výběr částice s pravděpodobností odpovídající její váze nacházející se na řádku 16 pak probíhá metodou vygenerováním náhodného čísla v intervalu <0, particle[N].sumWeigth> a jeho nalezení v sumách vah částic metodou půlení intervalu.

Na obrázku 6.5 je vyobrazen návrh systému Monte Carlo lokalizace, jeho napojení na senzory a následně na proces shlukování a odhad pozice robota. Obrázky 6.6 a 6.7 zobrazují příklad průběhu lokalizace na testovací trase 2 pro 10000 částic.

Algoritmus 6: AUGMENTED_MCL_WITH_COPY

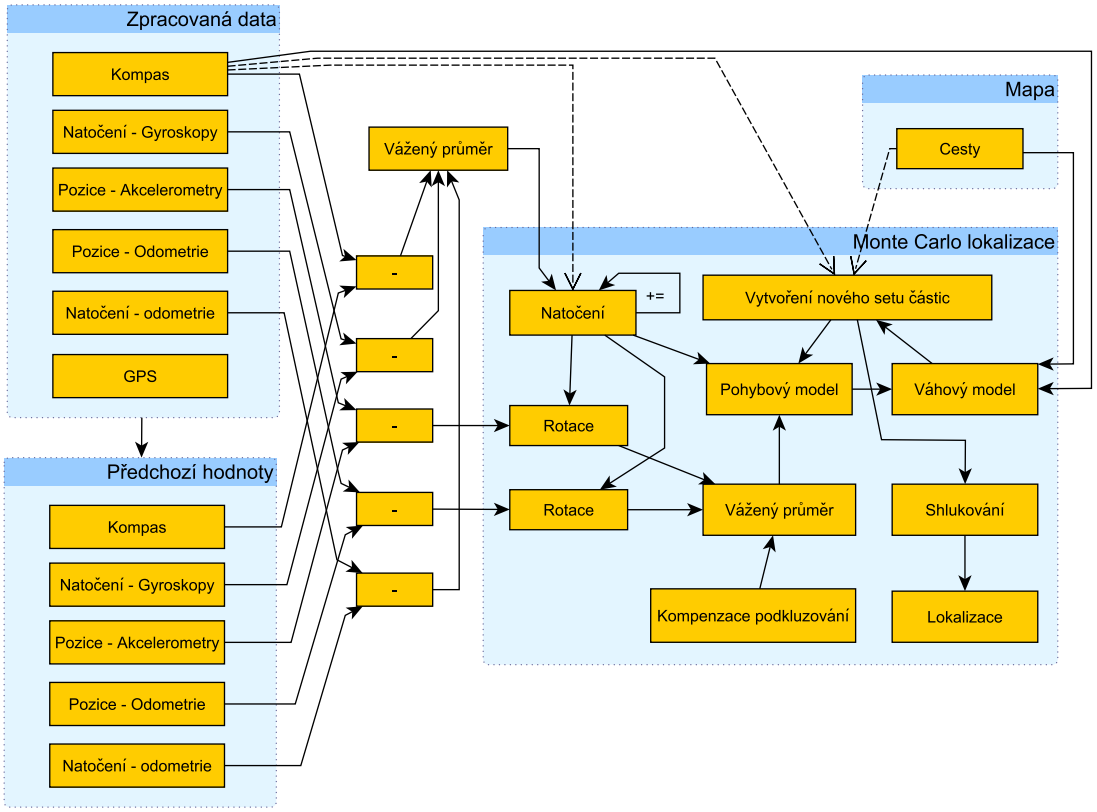
Input: $(\chi_{t-1}, u_t, z_t, m)$
Output: χ_t

1: static w_{slow}, w_{fast}
2: $\overline{\chi_t} = \chi_t = \emptyset$
3: **for** $m = 1$ to M **do**
4: $x_t^{[m]} = sample\_motion\_model(u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = measurement\_model(z_t, x_t^{[m]}, m)$
6: $\overline{\chi_t} = \overline{\chi_t} + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: $w_{avg} = w_{avg} + \frac{1}{M} \cdot w_t^{[m]}$
8: **end for**
9: $w_{slow} = w_{slow} + \alpha_{slow} \cdot (w_{avg} - w_{slow})$
10: $w_{fast} = w_{fast} + \alpha_{fast} \cdot (w_{avg} - w_{fast})$
11: $particles\_copied$ = add particles with $weigth > copyWeigth$ to χ_t
12: **for** $m = particles\_copied$ to M **do**
13: **if** $Rand(0,1) < max(0.0, 1.0 - \frac{w_{fast}}{w_{slow}})$ **then**
14: add random pose to χ_t
15: **else**
16: draw m with probability $\approx w_t^{[m]}$
17: add $x_t^{[m]}$ to χ_t
18: **end if**
19: **end for**
20: **return** χ_t

6.3.1 Pohybový a váhový model

Pohybový model byl implementován na základě teoretického modelu popsaného v kapitole 3.3.2. Model využívá pro odhad natočení vážený průměr změny rotace udávaný kompasem, gyroskopem a odometrií. Tato rotace je také aplikována na korekci směru pohybu udávaného odometrií i akcelerometry a následně je z těchto dvou pohybů také spočítán vážený průměr. Jeho součástí je také kompenzace podkluzování podvozku, která je reprezentována nulovou hodnotou a svou vahou. Tento sloučený pohybový vektor a rotace jsou následně použity jako vstup pro pohybový model.

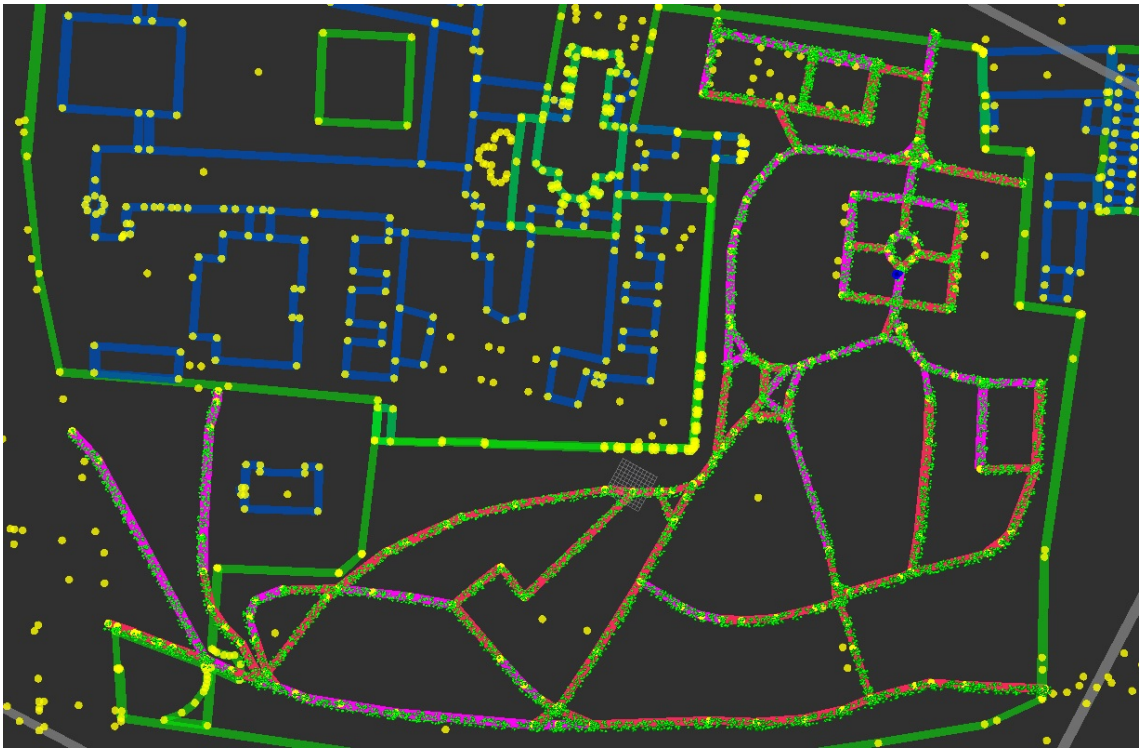
Váhový model využívá ke své činnosti úsečky úseků cest popsané výše a aktuální natočení robota udávaného kompasem. Pokud se částice nachází na cestě, dochází k navýšení váhy částice, pokud se nachází v okolí cesty, je váha snížena, a pokud částice opustí cestu zcela, je v další iteraci algoritmu vyřazena. V případě kompasu je sledována odchylka natočení částice vůči natočení kompasu – při stejném či podobném natočení váha částice roste, po překročení definovaného prahu pro odchylku váha částice klesá. Váhový model ke své činnosti tudíž potřebuje přesné mapové podklady a prostředí, kde nebude docházet k rušení kompasu okolními vlivy.



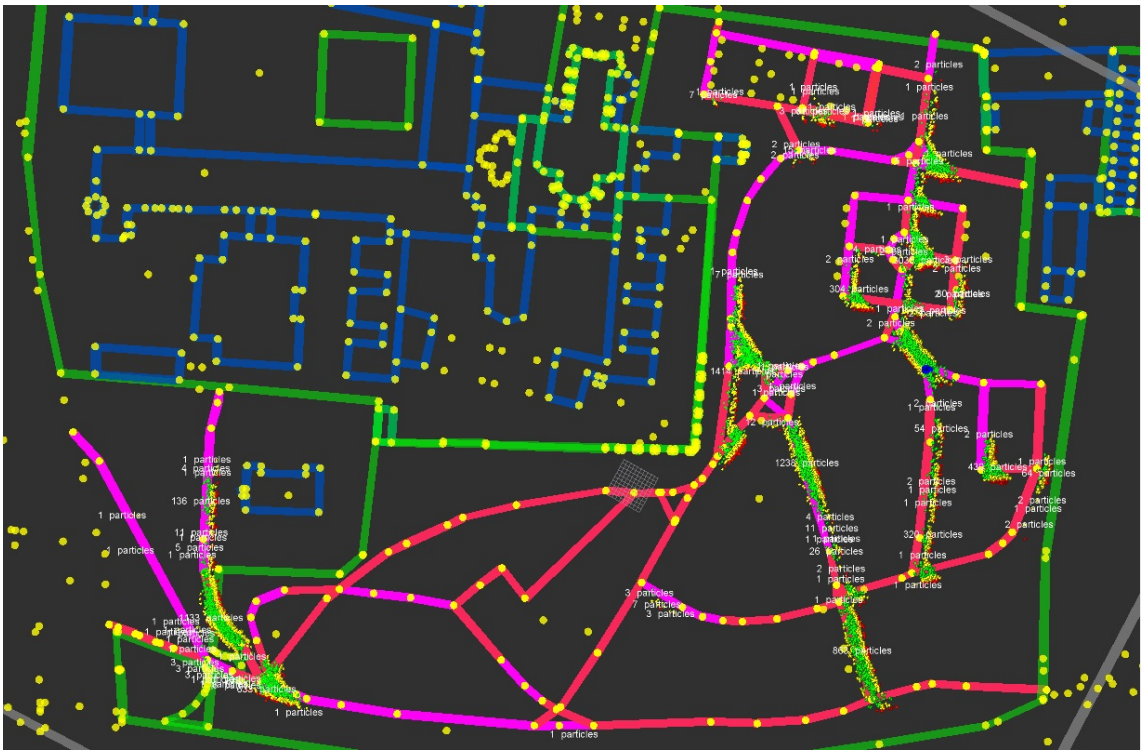
Obrázek 6.5: Návrh Monte Carlo lokalizace

6.3.2 Shlukování a lokalizace robota

Pro odhad přibližné polohy robota na základě polohy částic Monte Carlo lokalizace je použito shlukování. Shlukovací proces začíná po určité době běhu programu, kdy počet původně vygenerovaných částic klesne pod 30%, aby se předešlo vytvoření počátečního shluku zasahujícího celou mapu. Pro shlukování bylo z důvodů velkého množství částic a potřeby rychlého zpracování zvoleno mřížkové shlukování s kontrolou čtyřokolí. Nejprve jsou po aplikaci pohybového modelu vypočteny hranice mřížky ze souřadnic částic na mapě a je vytvořena mřížka zasahující všechny částice. Následně dochází k přiřazením každé částice do patřičného pole mřížky na základě souřadnic a do jeho interní struktury jsou zaznamenány její souřadnice, a počet částic v poli se zvýší o 1. Na konci tak buňky mřížky obsahují počet obsažených částic a sumy jejich souřadnic. V druhém kroku dochází k postupnému procházení mřížky, a pokud je nalezen čtverec, který má nenulový počet částic a nenáleží žádnému shluku, je přiřazen do nového shluku a je prohledáno jeho čtyřokolí. Pokud je v něm nalezena další buňka, která obsahuje také nenulový počet částic a nenáleží do žádného shluku, je do tohoto nového shluku přiřazena a její čtyřokolí je přidáno k prohledání. Tvorba shluku končí, když jsou všechna pole ve frontě prohledána a v okolí se nenachází žádné vyhovující pole. Shlukovací proces pak končí po dosažení posledního čtverce v mřížce. Pokud některý ze shluků má větší počet částic než je stanovený práh, je průměrná hodnota souřadnic částic tohoto shluku prohlášena za odhadovanou pozici robota.

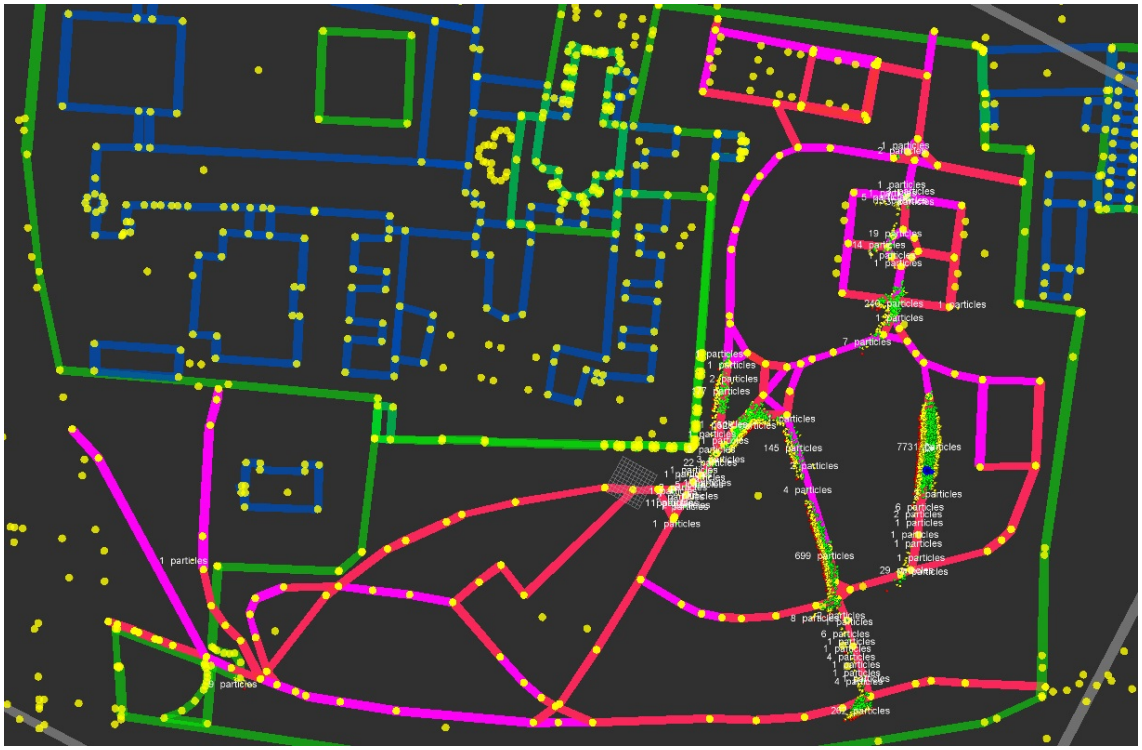


(a)

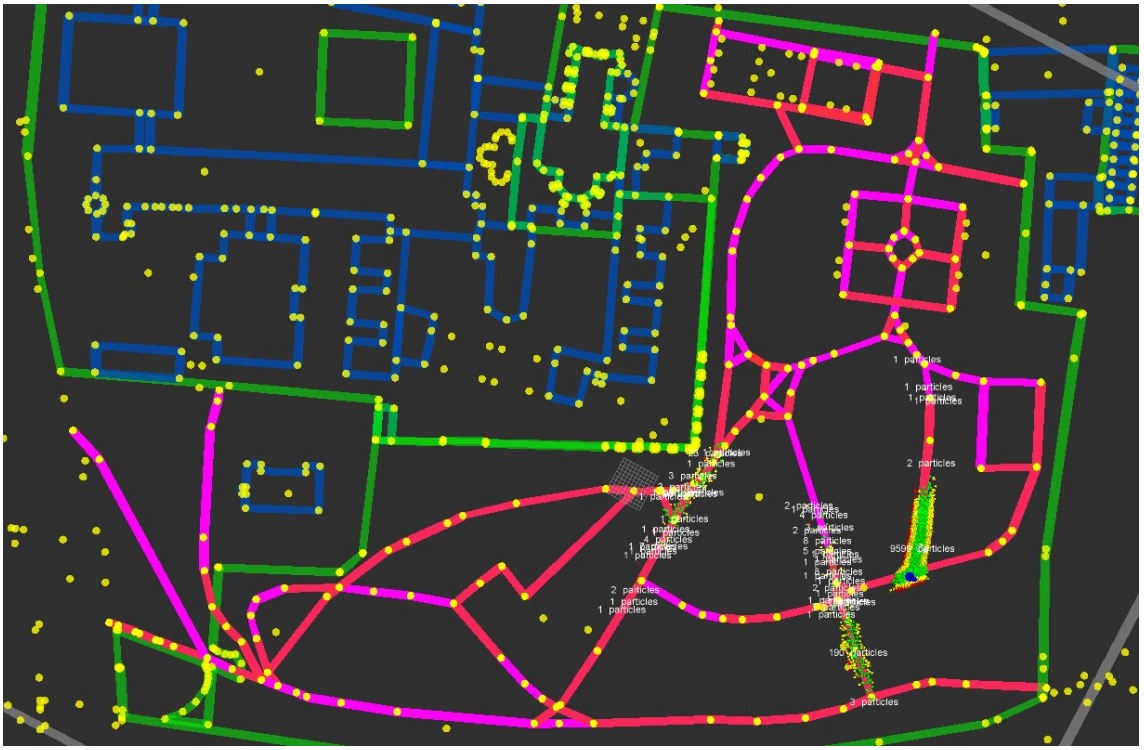


(b)

Obrázek 6.6: Ukázka činnosti Monte Carlo lokalizace v nástroji Rviz v čase 0 sekund – počáteční rovnoměrné vygenerování bodů 6.6(a) a 40 sekund – tvorba shluků 6.6(b)



(a)



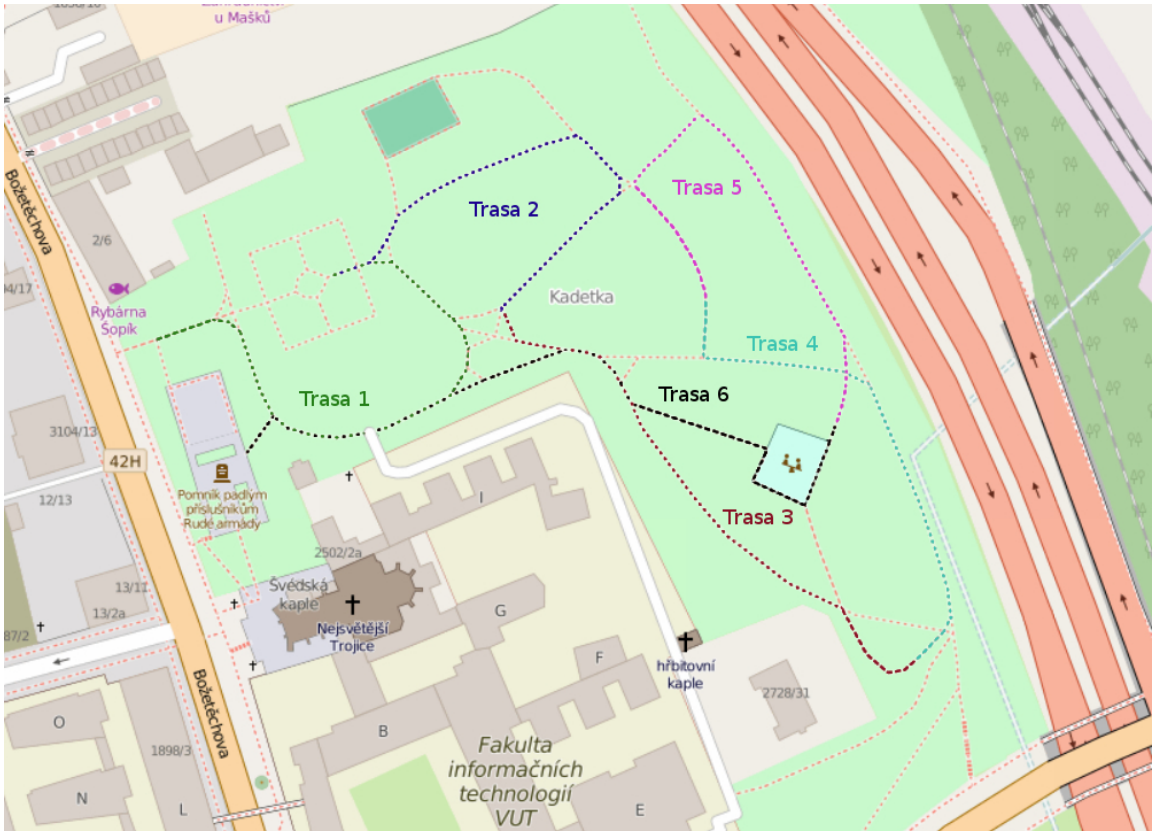
(b)

Obrázek 6.7: Ukázka činnosti Monte Carlo lokalizace v nástroji Rviz v čase 80 sekund – nalezení pozice robota 6.7(a) a 120 sekund – udržování pozice, posilování hlavního shluku 6.7(b)

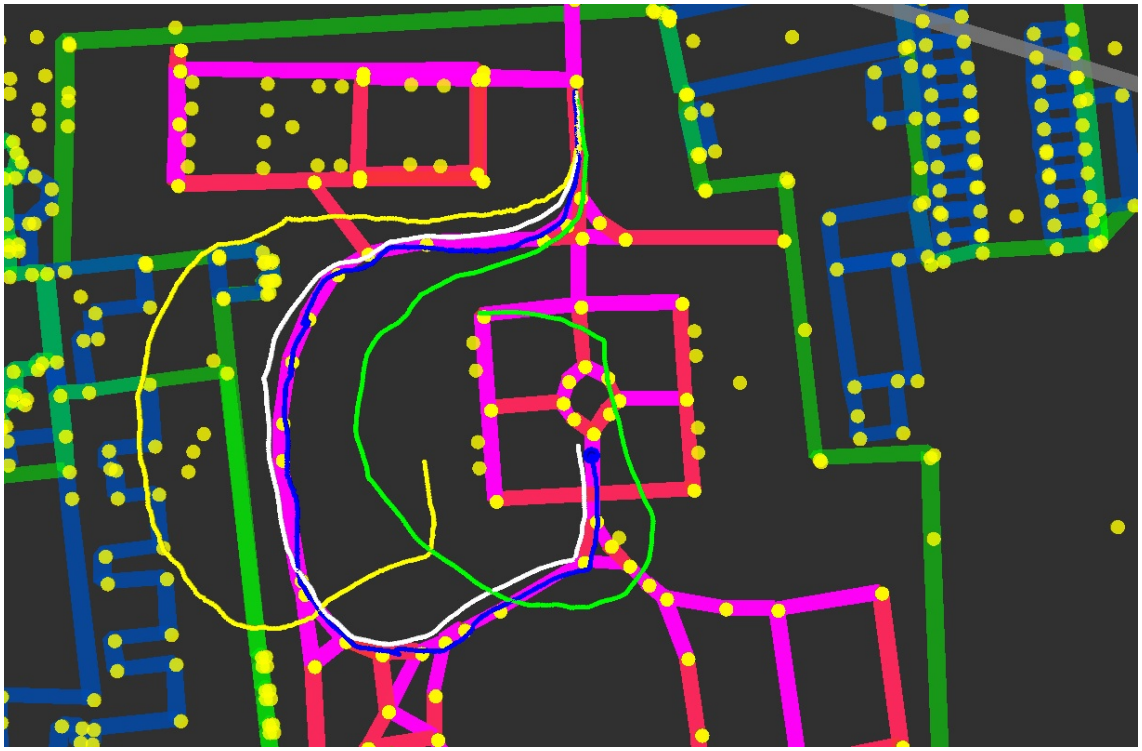
Kapitola 7

Testování a dosažené výsledky

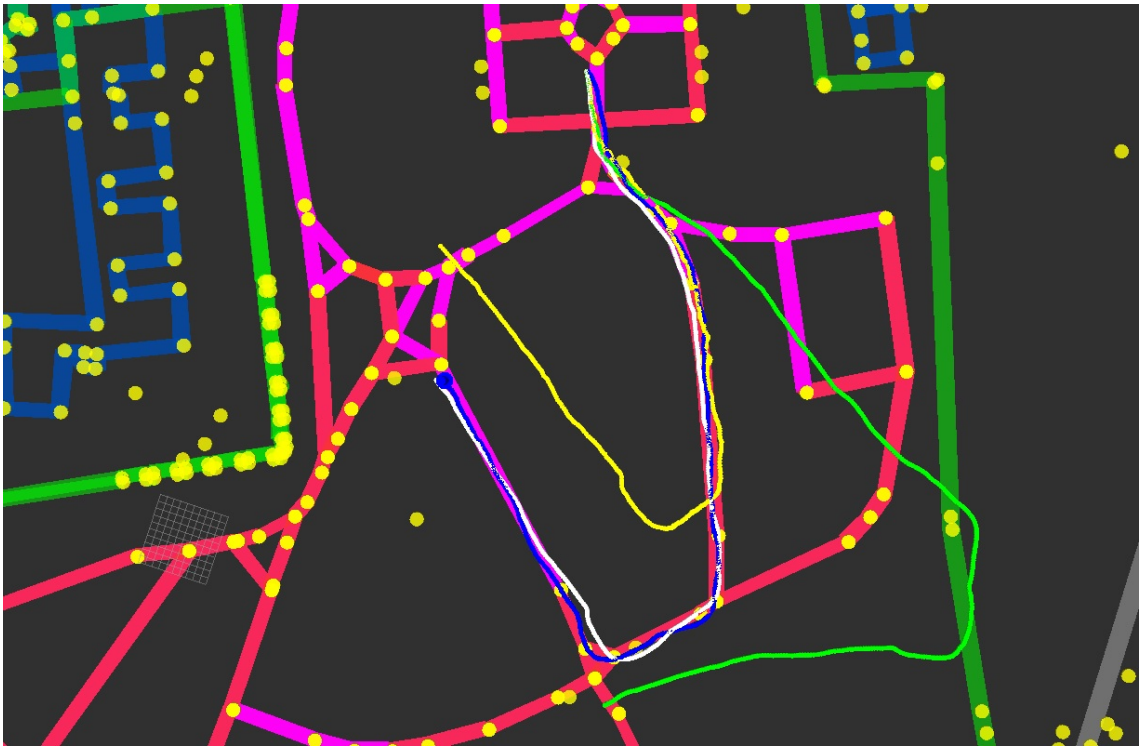
Testování probíhalo v rámci brněnského parku Kadetka. V prvotních fázích testování byly zjištěny značné nesrovnalosti v mapových podkladech – nepřesně umístěné nebo úplně chybějící cesty v rámci parku, a proto došlo k jejich přeměření pomocí GPS přijímače umístěného v modulu Microstrain 3dm gx3 45 a aktualizaci mapových podkladů na základě naměřených GPS stop. Pro testování bylo zvoleno šest tras vyobrazených na obrázku 7.1 a samotné testování bylo rozděleno na tři části – rekonstrukce ujeté trajektorie na základě dat ze senzorů, testování Monte Carlo lokalizace a testování problému uneseného robota.



Obrázek 7.1: Testovací trasy v brněnském parku Kadetka.

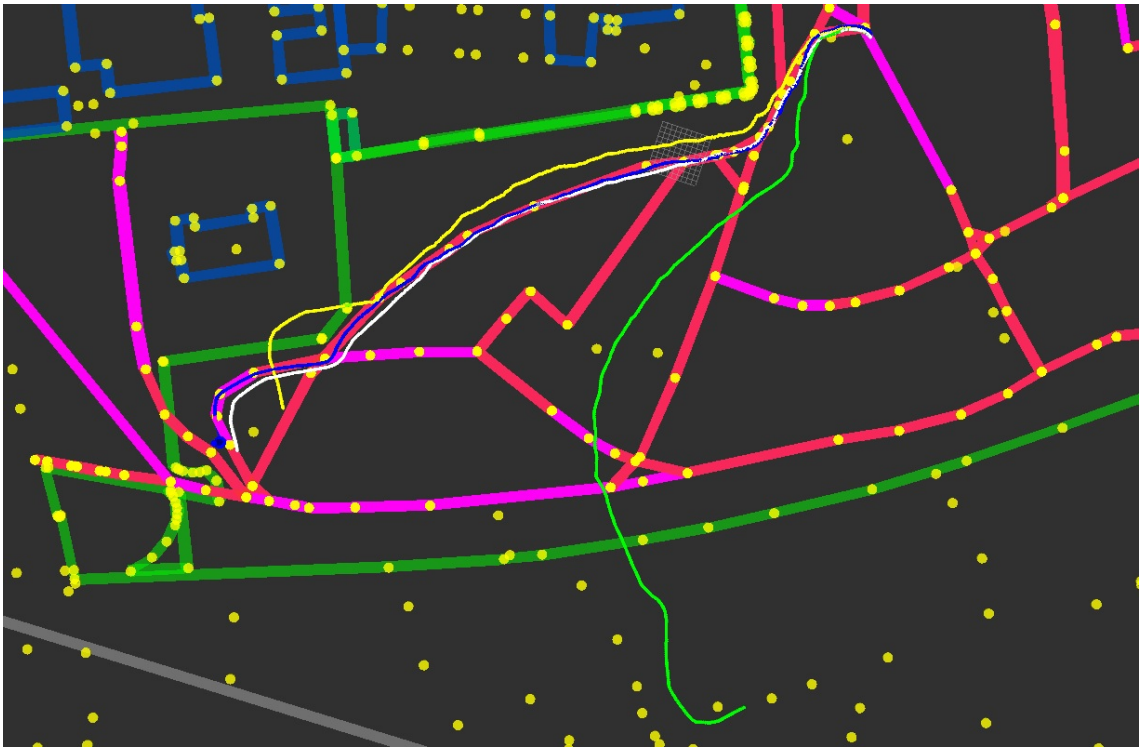


(a)



(b)

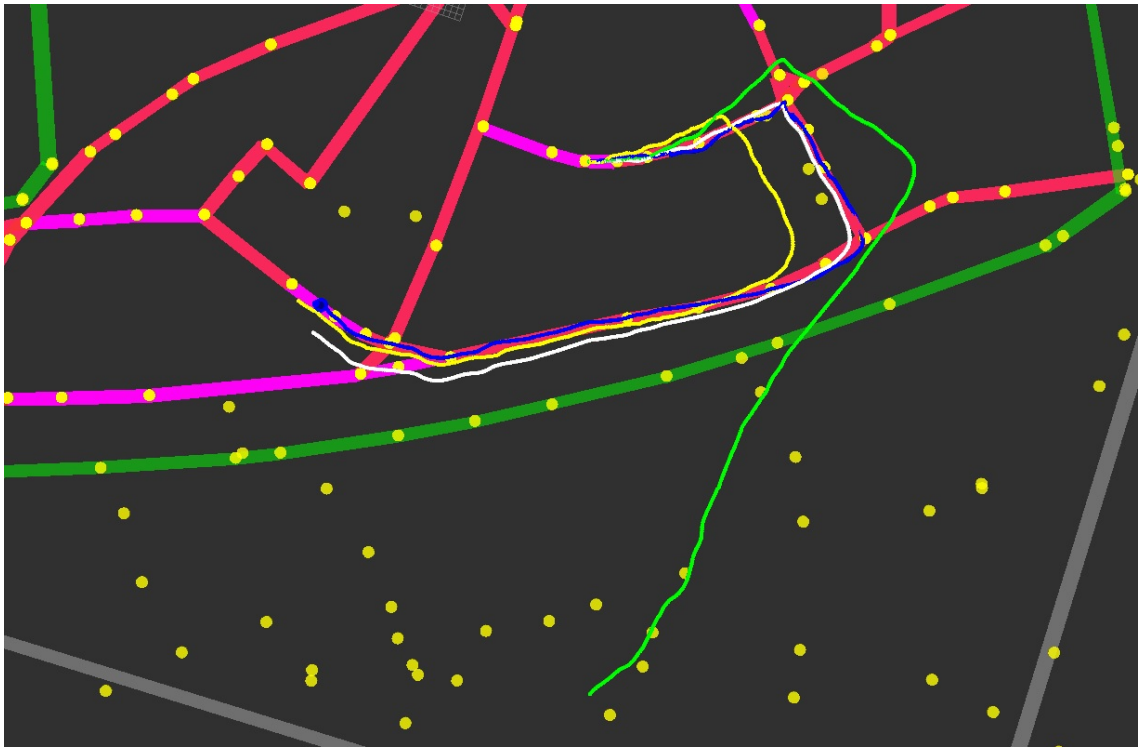
Obrázek 7.2: Odhadované trajektorie robota na základě dat ze senzorů pro trasu 1 7.2(a) a trasu 2 7.2(b). Zelená – odometrie, žlutá – akcelerometry a gyroskopy, bílá – kombinace všech senzorů, modrá – referenční trasa podle GPS.



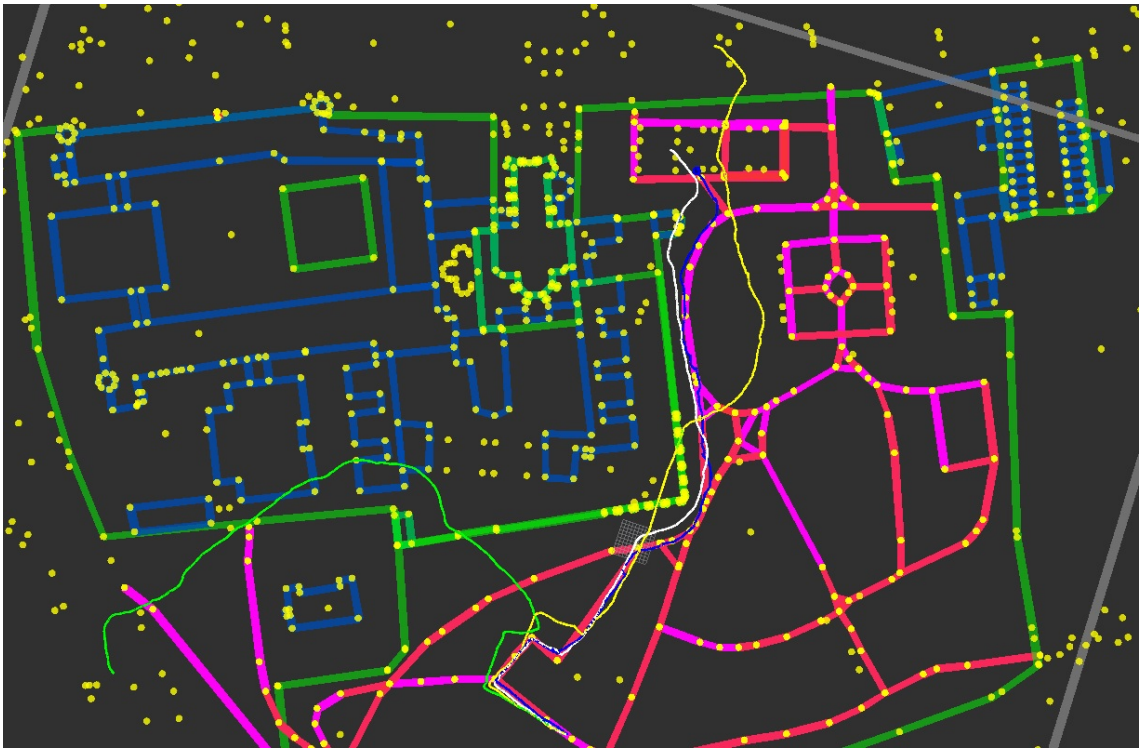
(a)

(b)

Obrázek 7.3: Odhadované trajektorie robota na základě dat ze senzorů pro trasu 3 7.3(a) a trasu 4 7.3(b). Zelená – odometrie, žlutá – akcelerometry a gyroskopy, bílá – kombinace všech senzorů, modrá – referenční trasa podle GPS.



(a)



(b)

Obrázek 7.4: Odhadované trajektorie robota na základě dat ze senzorů pro trasu 5 7.4(a) a trasu 6 7.4(b). Zelená – odometrie, žlutá – akcelerometry a gyroskopy, bílá – kombinace všech senzorů, modrá – referenční trasa podle GPS.

7.1 Rekonstrukce trajektorie robota

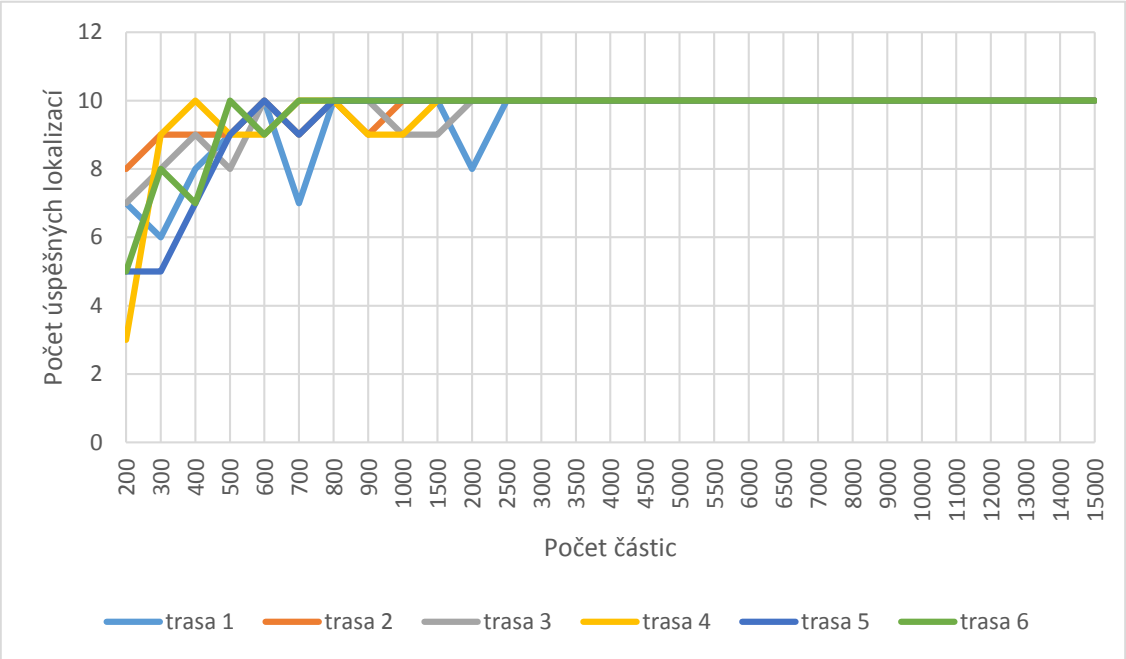
V této sekci se nachází porovnání odhadu trajektorie z jednotlivých senzorů, výsledné trajektorie po spojení dat ze všech senzorů a jejich porovnání z trasou zaznamenanou GPS jednotkou jako referenční trasou. Na obrázcích 7.2, 7.3 a 7.4 jsou vyobrazeny trajektorie pro jednotlivé ujeté trasy.

Jak je z obrázků patrné, odometrie je ve všech případech schopna se značnou přesností určit ujetou vzdálenost, ale má značnou chybu ve směru pohybu. Spojení dat z akcelerometrů a gyroskopů je schopné do značné míry kopírovat směr pohybu robota, ale vykazuje vysokou chybovost v případě ujeté vzdálenosti. Nejlepší výsledky se pak podařilo dosáhnout spojení všech senzorů pomocí váženého průměru.

Pro spojení dat byly u určení natočení robota použity váhy 55% pro kompas, 45% pro gyroskopy a 0% pro odometrii. V případě měření ujeté vzdálenosti byly použity váhy 83% pro odometrii, 10% pro akcelerometry a 7% pro kompenzaci podkluzování kol podvozku. Tyto váhy byly experimentálně zjištěny na základě množství konfigurací a výsledná trajektorie získaná těmito vahami kopíruje s vysokou přesností trajektorii zaznamenanou GPS.

7.2 Testování Monte Carlo lokalizace

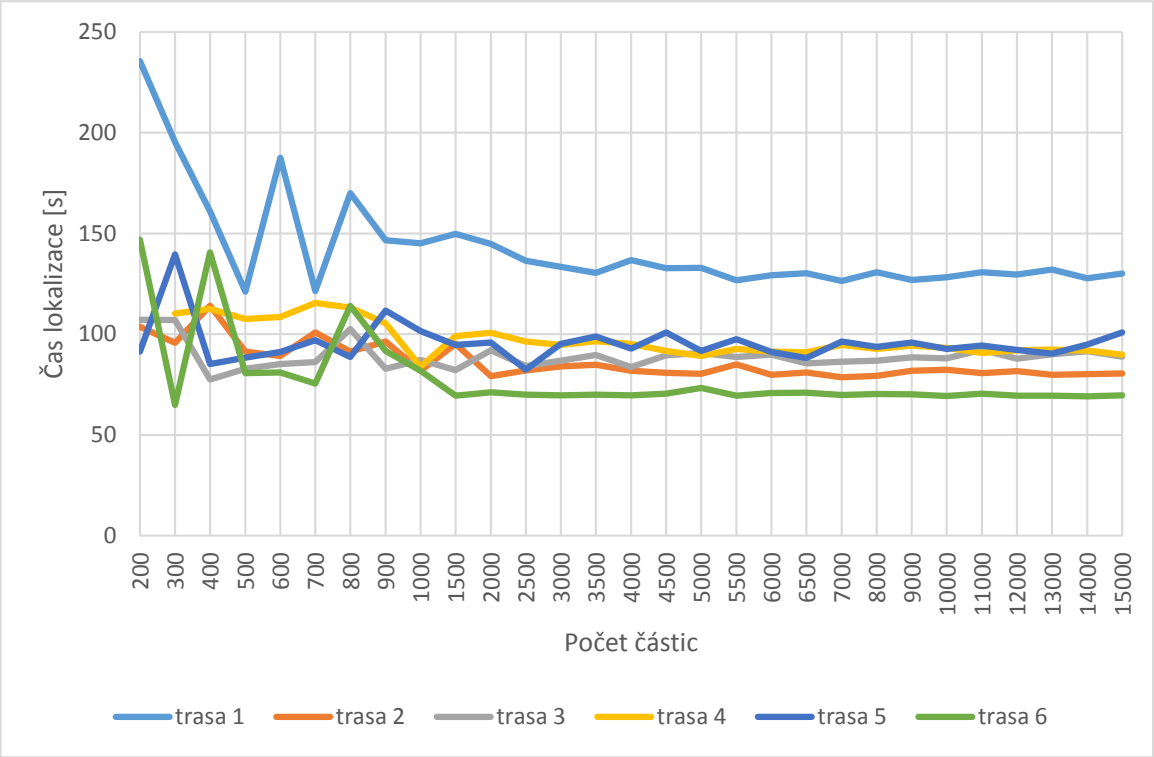
V případě Monte Carlo lokalizace byl testován vliv množství použitých částic na úspěšnost lokalizace a schopnost udržení pozice, jakmile k lokalizaci dojde. Testování probíhalo pro počty částic od 200 do 15000, vahami pro pohybový model $\alpha_1 = 0,03$, $\alpha_2 = 0,03$, $\alpha_3 = 1.2$, $\alpha_4 = 0,15$ a parametry Monte Carlo lokalizace $\alpha_{fast} = 0,8$ a $\alpha_{slow} = 0,1$, přičemž každý z testů byl proveden desetkrát pro omezení vlivu náhodnosti při běhu programu a do grafů byly vyneseny průměrné výsledky.



Obrázek 7.5: Úspěšnost lokalizace robota vzhledem k počtu částic.

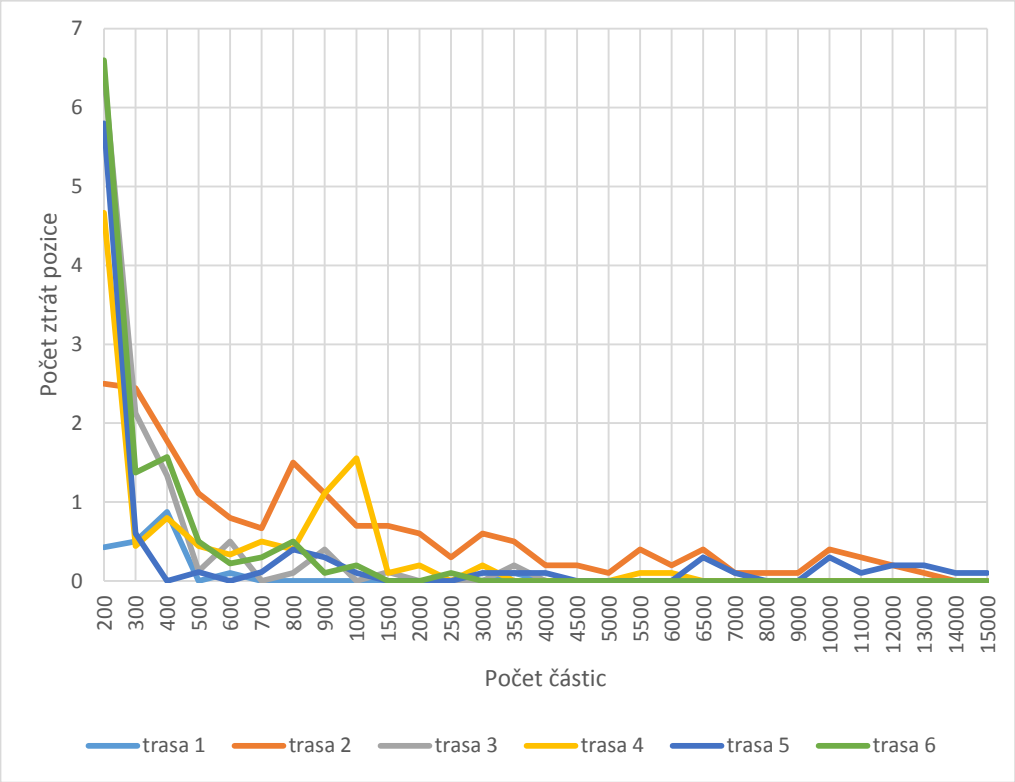
Graf 7.5 zobrazuje počet úspěšných lokalizací z deseti běhů programu pro každou z hodnot. Jak je z výsledků patrné, robot se s vysokou pravděpodobností dokáže lokalizovat i s malým množstvím částic – 500, od 2500 částic pak došlo k úspěšné lokalizaci již ve všech případech.

Druhou sledovanou veličinou byl vliv počtu částic na čas lokalizace robota vyobrazený v grafu 7.6. V grafu je vynesen průměrný čas úspěšných lokalizací z deseti měření. Při nízkých počtech částic zde hraje velkou roli vygenerování původního setu i odchylky během pohybu. V některý případech se částice trefí do správné pozice, a umožňují tak lokalizovat robota ve velmi krátkém čase, v opačných případech trvá nalezení robota dlouho nebo nedojde k jeho lokalizaci vůbec. Od 900 částic se již vyhledávací proces stabilizuje a optimálních hodnot opět dosahuje kolem 2500 částic, kdy pak již nedochází k výraznému zlepšení času lokalizace.

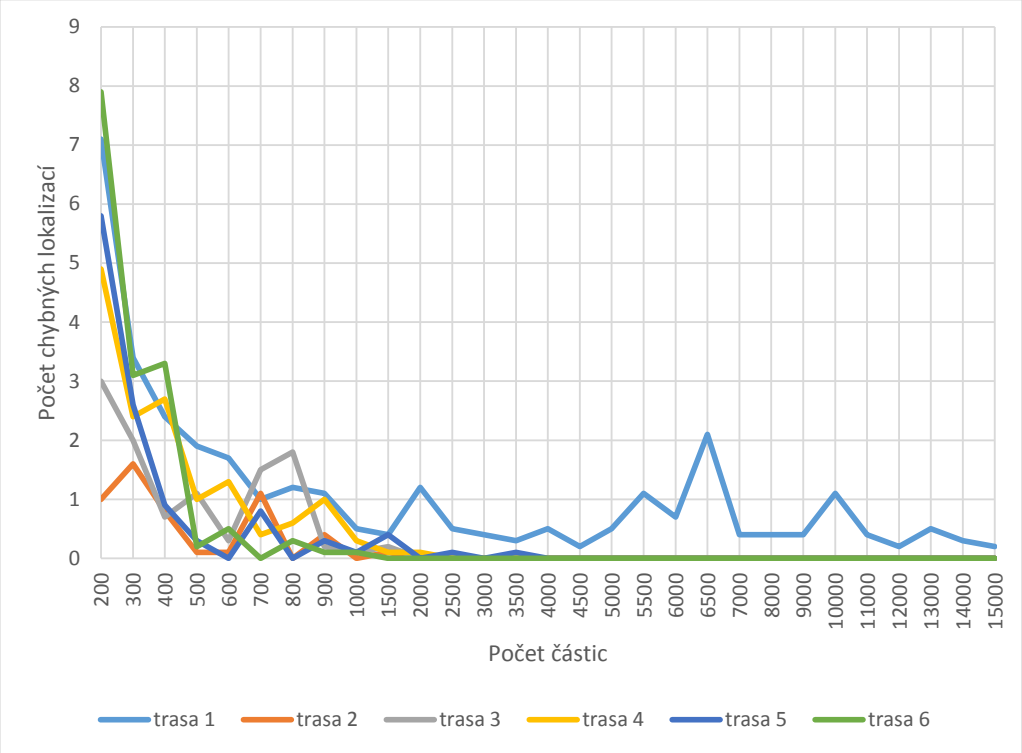


Obrázek 7.6: Graf průměrného času lokalizace na testovacích trasách.

Třetí sledovanou veličinou bylo sledování pozice robota v závislosti na počtu částic. Jak je z grafu 7.7 patrné, algoritmus má pro malé počty částic problém udržet sledovanou pozici a během běhu dochází k mnoha ztrátám sledované pozice. Optimálních hodnot pro většinu sledovaných tras zde dosahují počty částic 1500 až 2500, přičemž přidání dalších částic ještě dále zlepšuje stabilitu sledování pozice robota. Vetší počet ztrát pozice na testovací trase 2 je způsoben ostrou zatáčkou v čase 160 vteřin, kdy dojde ke značné redukci hlavního shluku a ztrátě pozice trvající v řádu vteřin.



Obrázek 7.7: Graf průměrného počtu ztrát pozice lokalizovaného robota.



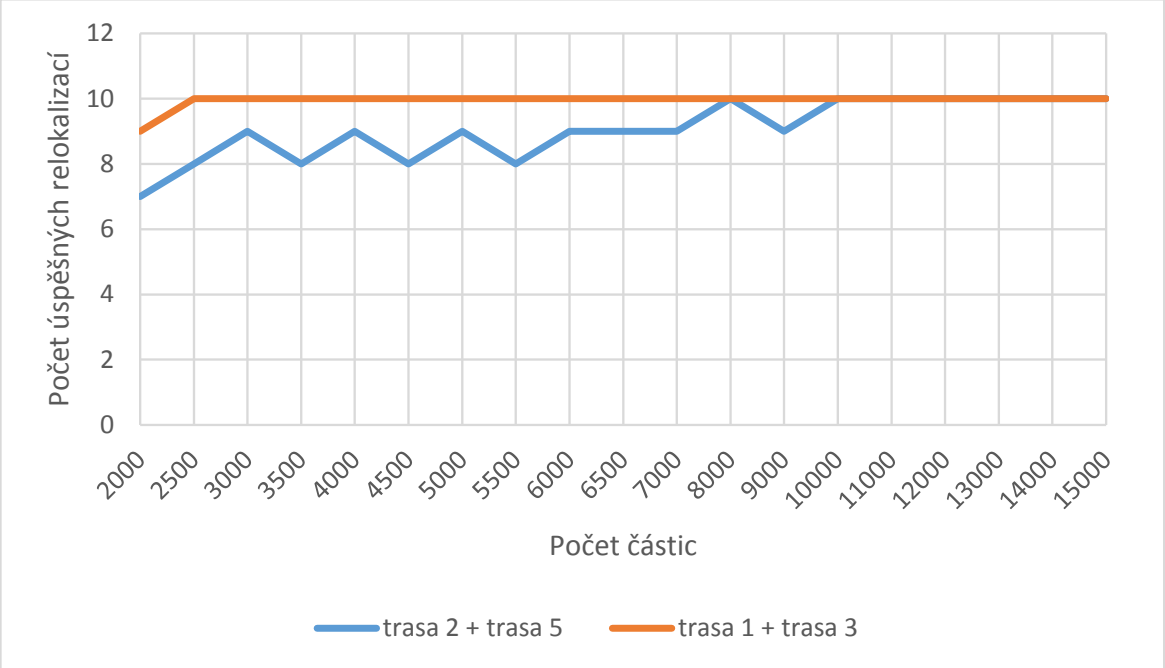
Obrázek 7.8: Graf průměrného počtu chybných lokalizací robota.

Poslední sledovanou veličinou byl vliv počtu částic na chybnou lokalizaci robota. Z hodnot vynesených v grafu 7.8 je opět patrné, že příliš malé množství částic vede k velkému množství chybně detekovaných pozic robota. Pro všechny testovací trasy s výjimkou trasy 1 se zde optimální hodnota částic pohybuje okolo 2000, kdy již nedocházelo k falešným lokalizacím. V případě trasy 1 pak dochází k falešným lokalizacím i při větším počtu částic kvůli k existenci dvou velmi podobných úseků, které se v mapě nachází.

7.3 Problém uneseného robota

V případě problému uneseného robota byl stejně jako v předchozím případě sledován vliv počtu částic u Monte Carlo lokalizace na úspěšnost opětovného nalezení robota a průměrný čas tohoto nalezení z deseti testů pro omezení vlivu náhodnosti. Pro testování byly zvoleny dvě dvojice tras, a sice trasa 2 následovaná trasou 5, druhou pak byla trasa 1 následovaná trasou 3. Dosažené výsledky byly vyneseny do grafů 7.9 a 7.10.

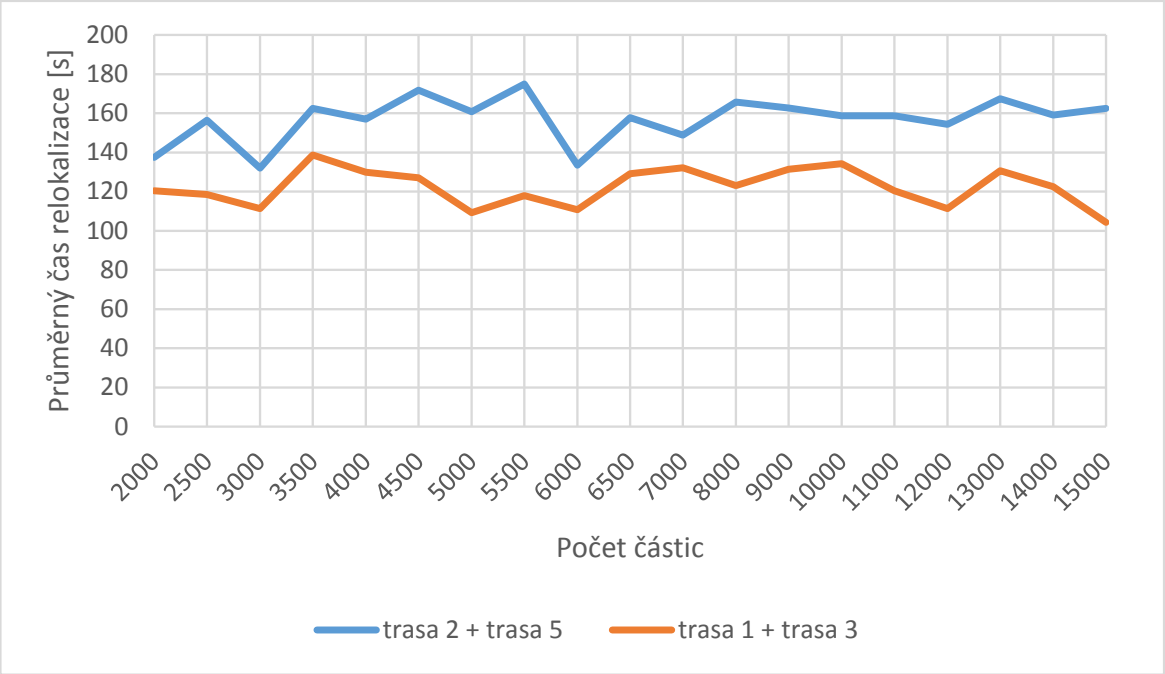
Testování probíhalo pro počty částic od 2000 do 15000, které v předchozím případě podávaly dobré výsledky pro prvotní lokalizaci. Váhy pro pohybový model byly zvoleny stejné jako v předchozím případě, tedy $\alpha_1 = 0,03$, $\alpha_2 = 0,03$, $\alpha_3 = 1.2$, $\alpha_4 = 0,15$ a parametry Monte Carlo lokalizace $\alpha_{fast} = 0,8$ a $\alpha_{slow} = 0,1$.



Obrázek 7.9: Graf počtu úspěšných nalezení uneseného robota.

Jak je z výsledků patrné, v případě problému uneseného robota je vhodné použít větší počet částic než pro lokalizaci a sledování pozice. K jistému znovunalezení pozice robota bylo zapotřebí 10000 a více částic v rámci první dvojice tras, což je násobně více než v předchozím případě. V případě druhé testované dvojice postačovalo pro lokalizaci 2500 jako v předchozím případě.

Z grafu 7.10 pak vyplývá, že počet částic nemá příliš velký vliv na čas relokalizace. V případě nízkých počtů částic hraje jako v předchozím případě velkou roli náhodnost, zda dojde k vygenerování náhodné částice na správném místě – v takovém případě pak dochází k rychlejší lokalizaci než v případě velkého počtu částic.



Obrázek 7.10: Průměrný čas znovunalezení uneseného robota.

Kapitola 8

Závěr

V rámci diplomové práce byly nastudovány jednotlivé senzory vhodné pro lokalizaci robota, jejich vlastnosti a vhodnost a využitelnost pro daný účel. Dále pak byly nastudovány lokalizační algoritmy, middleware ROS a OpenStreet mapy, pro které byl tento systém navržen a implementován.

Pro lokalizaci robota se používá rozšířené Monte Carlo lokalizace s generováním náhodných částic, odometrického pohybového modelu a vlastního váhového modelu využívajícího segmenty cest a odchylku natočení robota vůči kompasu. Pro odhad výsledné pozice je použito shlukování.

Navržený systém byl otestován na šesti testovacích trasách v rámci brněnského parku Kadetka, přičemž byla sledována rekonstrukce trajektorie robota a její porovnání s referenční GPS trajektorií. Dále pak vliv počtu částic na úspěšnost lokalizace, dobu lokalizace, počet falešných lokalizací, počet ztrát pozice při sledování lokalizovaného robota a dále úspěšnost a doba relokalizace v problému uneseného robota.

V případě rekonstrukce ujeté trajektorie podávala na uraženou vzdálenost výrazně lepší výsledky odometrie. Akcelerometry projevují značnou chybovost na odhad uražené vzdálenosti. V případě odhadu úhlu zatočení podávaly akcelerometry i kompas velmi dobré výsledky, v případě gyroskopů však s přibývajícím časem dochází ke kumulaci drobné chyby.

Pro lokalizaci a sledování pozice podával nejlepší výsledky počet částic 2500, tedy přibližně 1.31 částice na každý metr cesty v mapě. Další navyšování počtu částic vedlo již jen k drobným zlepšením za cenu ztráty výpočetního výkonu. V případě problému uneseného robota je vhodné použít částic více – stabilních výsledků zde dosahuje počet částic 10000, tedy přibližně 5.25 částice na každý metr cesty v mapě.

Jako další rozšíření této práce by bylo možné použít dalších senzorů – určení úhlu zatočení s pomocí detekce bodů v obrazu kamery a jejich sledování nebo odhad odometrie za pomoci sledování objektů lidarem. Metodu Monte Carlo lokalizace by bylo možné rozšířit o dynamický počet částic, kdy se jejich počet snižuje při úspěšné lokalizaci a naopak navyšuje při ztrátě pozice, a to na základě váhy částic.

Literatura

[1] AustinHendrix: *Rosbash* [Online]. 13.6.2015 [cit. 3.1.2016].
URL http://wiki.ros.org/rosbash#rosrun

[2] Bennett, J.: *OpenStreetMap: Be your own Cartographer*. Packt Publishing Ltd., 32 Lincoln Road, Olton, Birmingham, B27 6PA, UK., první vydání, 2010, ISBN 978-1-847197-50-4, 234 s.

[3] Carter, J.; Schmid, K.; Waters, K.; aj.: *Lidar 101: An Introduction to Lidar Technology, Data, and Applications* [Online]. Technická zpráva, National Oceanic and Atmospheric Administration (NOAA) Coastal Services Center, 2012 [cit. 3.1.2016].
URL https://coast.noaa.gov/digitalcoast/_/pdf/lidar101.pdf

[4] DirkThomas: *Roscore* [Online]. 31.12.2013 [cit. 3.1.2016].
URL http://wiki.ros.org/roscore

[5] DirkThomas: *Parameter Server* [Online]. 7.8.2013 [cit. 3.1.2016].
URL http://wiki.ros.org/Parameter%20Server

[6] EdVenator: *Catkin* [Online]. 30.7.2015 [cit. 3.1.2016].
URL http://wiki.ros.org/catkin

[7] Groves, P. D.: *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. Artech House, Boston, MA, druhé vydání, 2013, ISBN 978-1-60807-005-3, 800 s.

[8] IsaacSaito: *Rosout* [Online]. 10.10.2015 [cit. 3.1.2016].
URL http://wiki.ros.org/rosout

[9] IsaacSaito: *Rosnode* [Online]. 26.4.2013 [cit. 3.1.2016].
URL http://wiki.ros.org/rosnode

[10] JochenSprickerhof: *Rosbag* [Online]. 16.6.2015 [cit. 3.1.2016].
URL http://wiki.ros.org/rosbag

[11] Kamiccolo: *Roslaunch* [Online]. 12.5.2015 [cit. 3.1.2016].
URL http://wiki.ros.org/roslaunch

[12] KenConley: *Rosservice* [Online]. 15.7.2011 [cit. 3.1.2016].
URL http://wiki.ros.org/rosservicee

[13] KenConley: *Master* [Online]. 3.2.2012 [cit. 3.1.2016].
URL http://wiki.ros.org/Master

[14] Langle, R. B.: Getting your bearings: The magnetic compass and GPS. *GPS World*, ročník 14, č. 9, září 2003 [cit. 3.1.2016].
URL http://www2.unb.ca/gge/Resources/gpsworld.september03.pdf

[15] NickLamprianidis: *Rosmake* [Online]. 17.11.2012 [cit. 3.1.2016].
URL http://wiki.ros.org/rosmake

[16] Olson, E.: A primer on odometry and motor control. 2004.
URL http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-186-mobile-autonomous-systems-laboratory-january-iap-2005/study-materials/odomtutorial.pdf

[17] Quigley, M.; Conley, K.; Gerkey, B.; aj.: ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, ročník 3, 2009, str. 5.
URL https://www.willowgarage.com/sites/default/files/icraoss09-ROS.pdf

[18] Slabaugh, G. G.: Computing Euler angles from a rotation matrix.
URL http://www.staff.city.ac.uk/~sbbh653/publications/euler.pdf

[19] ThibaultKruse: *Rosbuild* [Online]. 13.11.2012 [cit. 3.1.2016].
URL http://wiki.ros.org/rosbuild

[20] Thrun, S.; Burgard, W.; Fox, D.: *Probabilistic robotics*. The MIT Press, první vydání, 2005, ISBN 978-0262201629, 672 s.

[21] TimOberhauser: *Rviz* [Online]. 14.8.2015 [cit. 3.1.2016].
URL http://wiki.ros.org/rviz/UserGuide

[22] TomMoore: *Rostopic* [Online]. 23.4.2013 [cit. 3.1.2016].
URL http://wiki.ros.org/rostopic

[23] TullyFoote: *Verbosity Levels* [Online]. 26.7.2010 [cit. 3.1.2016].
URL http://wiki.ros.org/Verbosity%20Levels

[24] Zennaro, S.: *Evaluation of Microsoft Kinect 360 and Microsoft Kinect One for robotics and computer vision applications* . Diplomová práce, Universita degli studi di Padova, Via 8 Febbraio 1848, 2, 35122 Padova PD, Itálie, 2014.

[25] Zhang, Z.: Microsoft kinect sensor and its effect. *MultiMedia, IEEE*, ročník 19, č. 2, 2012: s. 4–10.