VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

APLIKACE SLAM ALGORITMŮ PRO VOZIDLO S ČTYŘMI ŘÍZENÝMI KOLY

APPLICATION OF SLAM ALGORITHMS FOR 4WS VEHICLE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. JAN NAJMAN
AUTHOR

VEDOUCÍ PRÁCE — DOC. ING. ROBERT GREPL, PH.D.
SUPERVISOR

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav mechaniky těles, mechatroniky a biomechaniky
Akademický rok: 2014/2015

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Jan Najman

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Mechatronika (3906T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Aplikace SLAM algoritmů pro vozidlo s čtyřmi řízenými koly

v anglickém jazyce:

Application of SLAM algorithms for 4WS vehicle

Stručná charakteristika problematiky úkolu:

Tato práce se bude zabývat implementací SLAM algoritmů na konkrétní experimentální vozidlo se čtyřmi řízenými a poháněnými koly vyvíjené v Mechatronické laboratoři na ÚMTMB, FSI VUT v Brně (Mechlab).

Cíle diplomové práce:

1) Výběr a testování vlastností dostupných sensorů vhodných pro SLAM (odometrie, laserový skener, LEDDAR, Xtion a další).
2) Implementace SLAM algoritmů při použití různých variant kombinace snímačů v prostředí MATLAB. Porovnání variant z hlediska přesnosti lokalizace a mapování a výpočetní náročnosti.
3) Výběr nejvhodnější varianty, implementace pro řešení v reálném čase s využitím palubního počítače robotu, experimentální ověření v reálném indoor prostředí.

Seznam odborné literatury:

[1] Corke, P.: Robotics, Vision and Control: Fundamental Algorithms in MATLAB (Springer Tracts in Advanced Robotics), Springer, 2013
[2] Siegwart, R.: Introduction to Autonomous Mobile Robots (Intelligent Robotics and Autonomous Agents series), 2004
[3] http://wiki.ros.org/

Vedoucí diplomové práce: doc. Ing. Robert Grepl, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2014/2015.

V Brně, dne 21.11.2014

L.S.

prof. Ing. Jindřich Petruška, CSc. Ředitel ústavu	prof. RNDr. Miroslav Doupovec, CSc., dr. h. c. Děkan fakulty

Abstrakt

Tato práce se zabývá aplikací SLAM algoritmů na experimentální čtyřkolové vozidlo Car4. V první části práce je uveden princip fungování SLAM včetně popisu rozšířeného Kalmanova filtru, který je jednou z jeho hlavních součástí. Dále je zde uveden stručný seznam dostupných softwarových nástrojů k řešení této problematiky v prostředí programu MATLAB a přehled snímačů použitých v této práci.

Ve druhé části je uvedena metodika a výsledky testování jednotlivých snímačů a jejich kombinací pro výpočet odometrie a snímání okolního prostoru. Dále je zde uveden postup aplikace SLAM algoritmů na vozidlo Car4 s použitím vybraných snímačů a výsledky testování celého systému v praxi.

Klíčová slova

SLAM, lokalizace, mapování, laserový skener, Car4, Hokuyo, Xtion, IMU, RPLIDAR, kalmanův filtr, odometrie

Abstract

This paper deals with the application of SLAM algorithms on experimental four wheel vehicle Car4. The first part shows the basic functioning of SLAM including a description of the extended Kalman filter, which is one of its main components. Then there is a brief list of software tools available to solve this problem in the environment of MATLAB and an overview of sensors used in this work.

The second part presents methodology and results of the testing of individual sensors and their combinations to calculate odometry and scan the surrounding space. It also shows the process of applying SLAM algorithms on Car4 vehicle using the selected sensors and the results of testing of the entire system in practice.

Key words

SLAM, localization, mapping, laser scanner, Car4, Hokuyo, Xtion, IMU, RPLIDAR, Kalman filter, odometry

Bibliografická citace

NAJMAN, J. *Aplikace SLAM algoritmů pro vozidlo s čtyřmi řízenými koly*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2015. 75 s. Vedoucí diplomové práce doc. Ing. Robert Grepl, Ph.D..

Čestné prohlášení

Prohlašuji, že jsem diplomovou práci na téma "Aplikace SLAM algoritmů pro vozidlo s čtyřmi řízenými koly" vypracoval samostatně s použitím odborné literatury a pramenů uvedených v seznamu, který tvoří přílohu této práce.

V Brně dne 29. 5. 2015

.....

Jan Najman

Poděkování

Rád bych poděkoval svému vedoucímu práce doc. Ing. Robertu Greplovi, Ph.D, za jeho vedení, konzultace a cenné rady při tvorbě této práce. Dále bych chtěl poděkovat kolegům z laboratoře Mechlab za jejich pomoc a vytvoření příjemného pracovního prostředí a v neposlední řadě také své rodině a přátelům za podporu během tvorby této práce.

Obsah

1 Úvod

V posledních letech můžeme pozorovat velký rozvoj v oblasti mobilní robotiky, jehož výsledky vidíme třeba i v našem každodenním životě. Robotické vysavače již nejsou v našich domácnostech žádnou převratnou novinkou, moderní automobily dokážou automaticky popojíždět v dopravních zácpách či sledovat jízdní pruhy a většina velkých automobilových společností a výzkumných organizací má v současné době funkční prototyp plně autonomního vozidla a intenzivně pracuje na zdokonalování a vývoji pro jeho fungování v běžném provozu.

Je tedy zřejmé, že tato problematika je stále velmi aktuální a nejedná se pouze o okrajovou oblast bez velkého praktického využití.

Obr. 1.1: Autonomní vozidlo od společnosti Google [1]

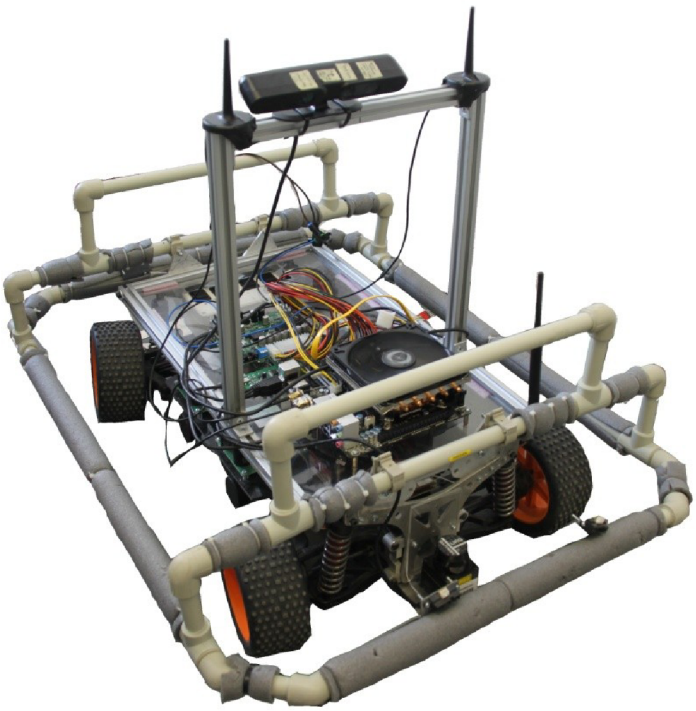
V Mechatronické laboratoři na ÚMTMB, FSI VUT v Brně (Mechlab) existuje již od roku 2010 experimentální vozidlo Car4, které slouží zejména studentům pro jejich práci v oblasti konstrukce, řízení a modelování čtyřkolových vozidel. Přestože se jedná pouze o zmenšený model, mohou být výsledky výzkumu využitelné při vývoji reálných automobilů.

Aby bylo možné využívat Car4 i pro experimenty právě z oblasti autonomního pohybu, bylo rozhodnuto o jeho rozšíření o další senzory pro snímání okolí, jako jsou například laserové skenery, prostorový snímač Xtion nebo dvojice kamer pro stereovizi.

Vzhledem k tomu, že v rámci projektu Car4 se jedná o zcela novou kapitolu, zabývá se tato práce mimo jiné také podstatou fungování jednotlivých senzorů, které bylo třeba zprovoznit a otestovat jejich specifické vlastnosti a omezení.

Druhou velkou kapitolou je aplikace SLAM algoritmů, které by měly vozidlu sloužit pro jeho orientaci v neznámém prostředí. Také zde bylo nutné kromě samotné realizace řešení zmínit i některé základní pojmy a provést výzkum v oblasti dostupných softwarových nástrojů.

Tato diplomová práce, společně s předcházející bakalářskou prací [2] by tedy měly sloužit jako výchozí bod pro další vývoj v oblasti poloautonomního a do budoucna i autonomního pohybu vozidla.



Obr. 1.2: Podoba experimentálního vozidla Car4 na počátku této práce

2 Rešeršní studie

V této první části práce se budeme zabývat obecnými principy SLAM algoritmů a dostupnými softwarovými i hardwarovými nástroji.

2.1 Teorie SLAM [3] [4]

Zkratka SLAM v překladu znamená Simultánní Lokalizace A Mapování. Jedná se o poměrně rozsáhlou oblast, která zpravidla řeší problém pohybu (často autonomního) mobilních robotů v částečně nebo zcela neznámém prostředí pomocí postupné tvorby mapy prostředí a současné lokalizace za použití téže mapy.

Rozsáhlost problematiky spočívá ve velkém množství použitelných senzorů, konstrukcí robotů a dalšího hardwaru, a dále v množství různých algoritmů, řešících jednotlivé kroky celého problému.

V této podkapitole se pokusíme o alespoň velmi stručný a zjednodušený přehled této oblasti se zaměřením na postupy použité v rámci této práce. Je vhodné připomenout, že zde zmíněné metody mají mnoho alternativ a nejsou zdaleka jediným možným přístupem k problematice SLAM.

2.1.1 Hardware a senzory využívané metodami SLAM

SLAM algoritmy jsou zpravidla aplikovány na nějakém druhu mobilního vozidla či robotu. Už samotný výběr typu a konstrukce robota má velký vliv na celý proces. Často se totiž využívají kinematické a dynamické modely pro predikci pohybu těchto robotů, dále s tímto souvisí také volba snímačů a použití specifických algoritmů.

Velmi často se, stejně jako v případě této práce, používá kolových vozidel, pohybujících se v prostředí, jež můžeme při určité míře zjednodušení, považovat za jednoúrovňové - 2D prostředí (např. jedno patro budovy).

S volbou typu konstrukce souvisí také pojem odometrie, což je odhad změny polohy vozidla v závislosti na čase, a to na základě údajů z jeho senzorů pohybu. V případě řešení celé úlohy ve dvourozměrném prostoru jsou typickým výstupem souřadnice polohy vozidla x a y (v kartézském souřadném systému) a také jeho aktuální natočení θ.

Způsobů získání odometrie je samozřejmě opět mnoho, mezi nejjednodušší patří například odhad ujeté vzdálenosti na základě údajů ze senzorů pootočení kol (rotační enkodéry) a použití jednoduchého kinematického modelu pohybu vozidla. Hlavním problémem odometrie bývá ovšem poměrně rychlá ztráta přesnosti, jelikož dochází k postupné akumulaci chyb měření, takže po čase dojde k úplnému odchýlení skutečné a odhadované polohy vozidla v prostoru.

A právě z důvodu nepřesnosti odometrie jsou vozidla dále vybavena minimálně jedním senzorem, který je schopný snímat své okolí (např. vzdálenosti od překážek). Typicky se pro tento účel využívá laserových skenerů, které jsou schopné provést několik desítek až stovek měření svého okolí během zlomku vteřiny a to s poměrně vysokou přesností.

S pomocí těchto údajů, je poté vozidlo schopné korigovat svou odhadovanou polohu. Tyto dva procesy - predikce polohy pomocí odometrie a následná korekce pomocí snímače okolí tvoří jádro SLAM procesu.

Konkrétní senzory použité při realizaci této práce jsou dále detailněji popsány v kapitole 2.3.

2.1.2 Přehled procesu SLAM

Nyní provedeme stručný popis celého procesu SLAM a jeho hlavních částí, z nichž většina byla aplikována i v rámci této práce.

Zjednodušeně probíhá SLAM podle následujícího vzorce:

1. vzít nepřesný odhad aktuální pozice robota (odometrie)
2. provést měření okolního prostředí (laserový skener)
3. na základě měření vyhledat v okolním prostředí orientační body (anglicky features/landmarks)
4. porovnat aktuální orientační body s dříve pozorovanými, které jsou uloženy v paměti
5. na základě srovnání aktuálně pozorovaných a uložených orientačních bodů vypočítat změnu polohy robota a aktualizovat původní odhad (bod 1.)
6. uložení nově pozorovaných orientačních bodů do paměti

Všechny tyto body proběhnou v rámci jednoho výpočetního kroku, a celý proces se cyklicky opakuje.

Stejně jako odometrie, ani pozorování orientačních bodů není bezchybné a tak je celý proces fúzí dvou nepřesných měření, jejímž výsledkem by měla být lepší celková přesnost odhadu skutečné polohy. Tuto fúzi obou měření zajišťuje algoritmus EKF (Extended Kalman Filter), který bude popsán níže v kapitole 2.1.4.

Nyní se na některé z těchto bodů SLAM algoritmu podíváme podrobněji.

2.1.3 Orientační body, metody jejich extrakce a asociace

Orientačním bodem rozumíme výrazný a jednoduše odlišitelný objekt v datech naměřených laserovým skenerem (nebo jiným snímačem okolí). Aby orientační body plnily správně svou funkci, měly by mít následující vlastnosti:

- Měly by být dostatečně unikátní, aby je nebylo možné jednoduše zaměnit s jinými orientačními body
- Měly by být opakovatelně pozorovatelné z různých úhlů a vzdáleností
- Mělo by se jich v daném prostředí nacházet dostatečné množství
- Měly by být statické

Všechny tyto vlastnosti jsou důležité pro plynulé, efektivní fungování celého procesu SLAM a také pro zamezení falešných detekcí nebo vzájemné záměny orientačních bodů.

Z tohoto důvodu je třeba věnovat zvláštní pozornost výběru typu orientačních bodů a metodě jejich extrakce z naměřených dat, zejména s ohledem na prostředí, v jakém se bude robot pohybovat. Je totiž zřejmé, že v místnosti plné lidí, prázdných kancelářských chodbách nebo například v průmyslové hale se spoustou ocelových sloupů, se budou nabízet rozdílné druhy vhodných orientačních bodů.

Pro bližší představu zde nyní zmíníme tři konkrétní příklady, z nichž poslední zmíněný je využit i v této práci:

1. **Majáky**

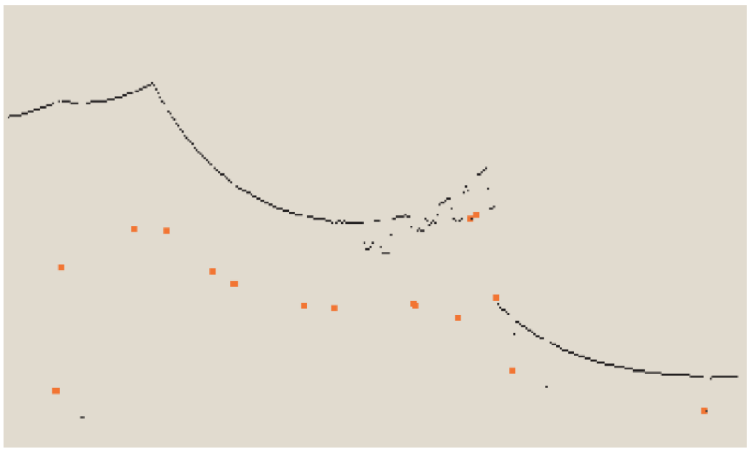
Umělé majáky jsou jedním z nejjednodušších a nejspolehlivějších typů orientačních bodů a je mnoho možností jak maják a jeho detekci realizovat. Může jít například o aktivní vysílač pracující na rádiovém či optickém principu [5] nebo o pasivní objekt, který silně odráží dopadající záření (např. reflexní páska odrážející laserový paprsek)

Výhodou je poměrně jednoduchá a přesná detekce, možnost umístění majáků v prostředí tak aby co nejlépe pokrývaly celý prostor a velmi malá pravděpodobnost falešných detekcí či záměny dvou majáků (při jejich vhodném rozmístění). Naopak nevýhodou je právě nutnost "modifikace" prostředí, což do určité míry omezuje univerzálnost tohoto řešení.

Obr. 2.1: Příklad majáku pracujícího jako infračervený vysílač [5]

2. Extrakce extrémů

Vstupem pro tuto metodu jsou data z laserového scanneru a na rozdíl od předchozího případu již nevyžaduje žádnou další elektroniku či pomůcky. Orientačním bodem se v tomto případě rozumí výrazné změny ve vzdálenostech naměřených snímačem (v rámci jednoho skenu). Tato metoda pak nejčastěji detekuje objekty jako jsou nohy stolu či různé hrany nábytku apod. Je vhodná do velmi členitého prostředí bohatého na podobné extrémy a naopak selhává v prostorách bez výrazných změn (např. dlouhé přímé chodby).

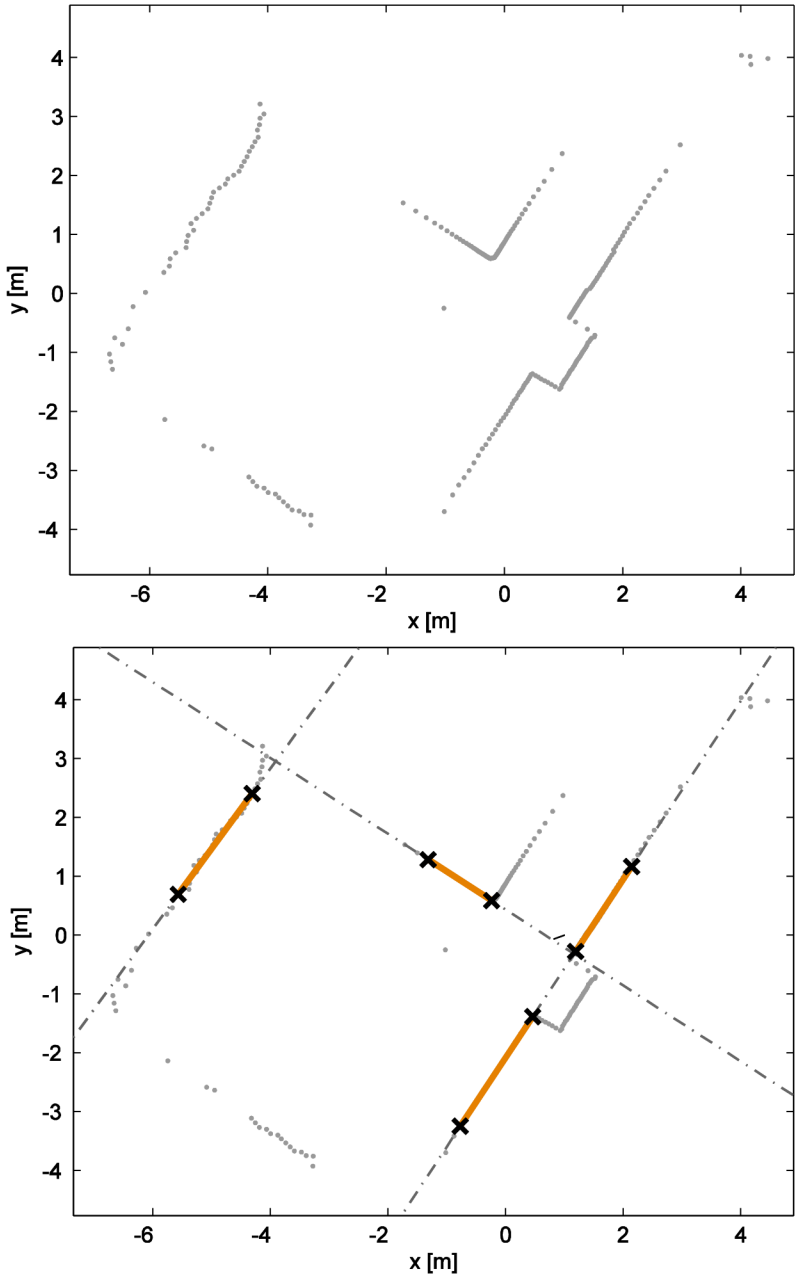


Obr. 2.2: Příklad extrakce extrémů. Oranžové tečky představují nohy stolu detekované jako orientační body [3]

3. Extrakce přímek

Tato metoda je do jisté míry opakem extrakce extrémů. Taktéž sice využívá pouze laserového skeneru, ovšem jako orientační body nepoužívá extrémy, ale naopak hledá v naměřených datech přímky. Algoritmů jak tyto přímky nalézt je více, nicméně jádro procesu spočívá v minimalizaci kolmé vzdálenosti bodů od prokládané přímky. Po nalezení přímek také často dochází ke sloučení duplicitních detekcí. Možností reprezentace takto nalezených orientačních bodů je také více. Často jsou reprezentovány například jako souřadnice *x, y* bodu přímky, ležícího nejblíže nějakému fixnímu bodu v prostoru (nejčastěji počátek souřadného systému mapy).

V případě extrakce orientačních bodů z dat laserového skeneru velmi závisí na nastavení parametrů detekčního algoritmu. Při hledání přímek je nutné nastavit například minimální délku detekovaných úseků, maximální odchylku bodů od prokládané přímky apod. Toto nastavení je často nutné ladit experimentálně, na základě výchozího odhadu a následného testování v reálném či simulovaném prostředí.



Obr. 2.3: Příklad extrakce přímek z dat laserového skeneru. Oranžové úseky představují nejlepší aproximace skupiny bodů přímkou. Ne všechny potenciální přímky jsou identifikovány jako orientační body díky nastavení určitého prahu citlivosti detekce.

Asociace dat

Aby mohl celý proces korekce polohy na základě orientačních bodů fungovat, je třeba v každém kroku správně přiřadit aktuálně pozorované orientační body k dříve pozorovaným, které jsou uloženy v paměti. Tento krok je další klíčovou součástí SLAM algoritmů. V případě chybného přiřazení může totiž dojít ke kolapsu celého systému, protože robot bude přesvědčen, že se nachází jinde, než ve skutečnosti je.

Je tedy důležité, aby při přiřazování existoval rozhodovací proces, který rozhodne, zda se jedná o opakované pozorování, které přiřadí správnému bodu v paměti, nebo zda jde o nový orientační bod. Níže je uveden příklad takovéhoto rozhodovacího procesu, jak je uveden v [3] (str. 27):

1. *při obdržení nového laserového skenu použít metodu extrakce orientačních bodů k nalezení všech viditelných orientačních bodů*
2. *přiřazení každého nalezeného orientačního bodu k nebližšímu orientačnímu bodu v databázi, který jsme pozorovali více než N-krát*
3. *předání všech takto přiřazených dvojic (aktuálně pozorovaný orientační bod, orientační bod v databázi) přes validační bránu*
    a. *pokud dvojice projde validační bránou, musí jít o ten samý orientační bod, který jsme pozorovali dříve, takže zvýšíme v databázi číslo, kolikrát jsme jej pozorovali o 1*
    b. *pokud dvojice neprojde validační bránou, přidáme orientační bod do databáze a nastavíme číslo, kolikrát jsme jej pozorovali na 1*

(pozn.: validační brána znamená, zda pozorovaný bod leží v předpokládané oblasti nejistoty získané pomocí EKF algoritmu. Více o validačních bránách a celé problematice asociace dat například v [6])

2.1.4 EKF

EKF (v překladu "rozšířený Kalmanův filtr") je algoritmus zodpovědný za aktualizaci původního odhadu polohy robota získaného odmetrií na základě pozorovaných orientačních bodů. Také vypočítává a uchovává v paměti údaje o míře nepřesnosti polohy robota a orientačních bodů.

Vzhledem k tomu, že odometrie ani měření laserového skeneru nejsou přesné, je v rámci EKF k naměřeným datům přidán určitý náhodný prvek, šum, s Gaussovým rozdělením pravděpodobnosti díky kterému je lépe modelováno skutečné chování senzorů. Celý algoritmus tedy nepočítá přesnou polohu vozidla a orientačních bodů ale spíše oblast jejich pravděpodobného výskytu. Čím jsou snímače, kinematické modely a další součásti algoritmu přesnější, tím jsou tyto oblasti menší. V případě dokonalých snímačů a modelů by se pak smrskly na body.

Cílem této kapitoly není podrobně vysvětlit celou problematiku EKF, která již byla v literatuře mnohokrát velmi dobře popsána (například [3] [7] [8] [9]). Důvodem zařazení této kapitoly je spíše ujasnění názvosloví a označení zejména pro jednotlivé matice užívané v EKF v rámci této práce a také zaměření na určitá specifika algoritmu při použití v rámci SLAM. Mnoho zde uvedených matic je uvedeno již ve výsledné formě po provedení určitých úprav.

Stavový vektor *x*

Uchovává odhadovanou polohu a natočení robota x_r, y_r a θ_r a dále polohu všech dosud pozorovaných orientačních bodů x_1, y_1 až x_n, y_n.

$$\begin{bmatrix} x_r \\ y_r \\ \theta_r \\ x_1 \\ y_1 \\ \vdots \\ x_n \\ y_n \end{bmatrix}$$

Obr. 2.4: Stavový vektor ***x***

Vektor pozorovaných orientačních bodů *z*

Reprezentuje orientační body pozorované robotem. Pro každý tento bod obsahuje jeho vzdálenost *r* a orientaci *b* vzhledem k aktuální pozici robota. Orientací se zde rozumí úhel mezi polohou bodu a směrem, kterým je robot natočen.

$$\begin{bmatrix} r_1 \\ b_1 \\ r_2 \\ b_2 \\ \vdots \\ r_n \\ b_n \end{bmatrix}$$

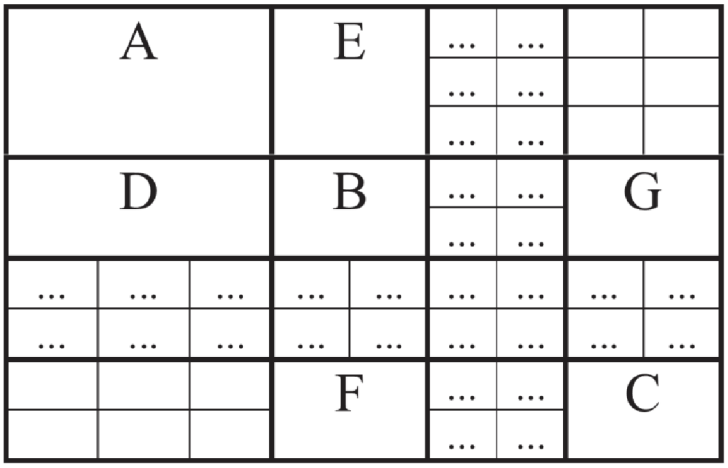
Obr. 2.5: Vektor orientačních bodů ***z***

Kovarianční matice *P*

Obsahuje kovarianci [10] pozice robota, polohy jednotlivých orientačních bodů a také vzájemné kovariance mezi nimi. Kovariance v podstatě popisuje nejistotu těchto poloh (robota i orientačních bodů). Matici můžeme pomyslně rozdělit na několik menších buněk (viz. Obr. 2.6). Označení buněk (**A** až **G**) nemá žádnou souvislost s názvy některých matic uvedených dále v této kapitole!

Buňka **A** (rozměr 3x3) obsahuje kovarianci pozice robota, další buňky na diagonále **B** až **C** obsahují kovariance pozic orientačních bodů (rozměr 2x2). Buňky mimo diagonálu popisují kovarianci mezi těmito prvky - buňky v prvních třech řádcích a sloupcích (mimo **A**) jsou kovariance mezi pozicí robota a jednotlivými orientačními

body (např. **D** je kovariance mezi pozicí robota a prvním orientačním bodem), ostatní buňky jsou kovariance mezi orientačními body navzájem (např. **F** je kovariance mezi prvním a posledním orientačním bodem). Matice ***P*** je symetrická, takže **E** je transpozicí **D**, **G** je transpozicí **F**, atd.



Obr. 2.6: Kovarianční matice ***P*** [3]

Jakobián modelu predikce *A*

Tato matice je použita k predikci polohy robota - jakým způsobem ujetá vzdálenost a pootočení (získané odometrií) změní jeho výslednou polohu a orientaci. Na

Obr. 2.7 je uvedena zjednodušená forma matice kde Δx a Δy jsou údaje posunutí robota v souřadnicích x a y.

1	0	$-\Delta y$
0	1	Δx
0	0	1

Obr. 2.7: Jakobián modelu predikce ***A***

Jakobián modelu měření *H*

Podobně jako v případě jako Jakobiánu ***A***, matice ***H*** udává, jak se změní vzdálenost a orientace všech orientačních bodů vůči predikované poloze robota, při změně jeho polohy a natočení.

Celá forma matice pro n orientačních bodů je na Obr. 2.8, kde x_r, y_r jsou souřadnice predikované polohy robota, x_i, y_i jsou souřadnice jednotlivých orientačních bodů a r_i jsou jejich vzdálenosti od robota, přičemž $i = 1, \ldots, n$.

$\frac{x_r - x_1}{r_1}$	$\frac{y_r - y_1}{r_1}$	0	$\frac{x_1 - x_r}{r_1}$	$\frac{y_1 - y_r}{r_1}$	0	0	$\cdots$	0	0
$\frac{y_1 - y_r}{r_1^2}$	$\frac{x_r - x_1}{r_1^2}$	-1	$\frac{y_r - y_1}{r_1^2}$	$\frac{x_1 - x_r}{r_1^2}$	0	0	$\cdots$	0	0
$\frac{x_r - x_2}{r_2}$	$\frac{y_r - y_2}{r_2}$	0	0	0	$\frac{x_2 - x_r}{r_2}$	$\frac{y_2 - y_r}{r_2}$	$\cdots$	0	0
$\frac{y_2 - y_r}{r_2^2}$	$\frac{x_r - x_2}{r_2^2}$	-1	0	0	$\frac{y_r - y_2}{r_2^2}$	$\frac{x_2 - x_r}{r_2^2}$	$\cdots$	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\frac{x_r - x_n}{r_n}$	$\frac{y_r - y_n}{r_n}$	0	0	0	0	0	$\cdots$	$\frac{x_n - x_r}{r_n}$	$\frac{y_n - y_r}{r_n}$
$\frac{y_n - y_r}{r_n^2}$	$\frac{x_r - x_n}{r_n^2}$	-1	0	0	0	0	$\cdots$	$\frac{y_r - y_n}{r_n^2}$	$\frac{x_n - x_r}{r_n^2}$

Obr. 2.8: Jakobián modelu měření **H** [8]

Kovarianční matice *Q* a *R*

Jak již bylo řečeno na začátku této podkapitoly, EKF pracuje s nejistotami měření jak odometrie tak laserového skeneru, které reprezentují právě tyto matice ***Q*** a ***R***.

Matice **Q** popisuje náhodnou chybu měření odometrie. Níže uvedená forma matice (Obr. 2.9) je opravdu velmi jednoduchá a lze provést její různé modifikace, aby lépe modelovala chybu odometrie (viz. například [11]).

Získáme ji vynásobením údajů o změně polohy a natočení s konstantou q, která udává míru nepřesnosti s Gaussovským rozložením a nulovou střední hodnotou (q odpovídá rozptylu). Velikost této hodnoty závisí na naší znalosti toho, jak přesná je obecně celá odometrie (zjištěno například experimentálně) a často bývá ještě korigována na základě praktických testů celého SLAM procesu.

$q\Delta x^2$	$q\Delta x\Delta y$	$q\Delta x\Delta\theta$
$q\Delta y\Delta x$	$q\Delta y^2$	$q\Delta y\Delta\theta$
$q\Delta\theta\Delta x$	$q\Delta\theta\Delta y$	$q\Delta\theta^2$

Obr. 2.9: Kovarianční matice ***Q***

Matice ***R*** popisuje nepřesnost měření laserového skeneru, u nějž taktéž předpokládáme Gaussovské rozložení, pro každý pozorovaný orientační bod. Tvar matice pro n těchto bodů je na Obr. 2.10, kde r_i a b_i jsou naměřené vzdálenosti a orientace jednotlivých orientačních bodů, přičemž $i = 1, \ldots, n$. Konstanty c a d pak opět udávají míru nepřesnosti měření, obdobně jako v případě matice ***Q***. Velikost těchto konstant opět závisí na naší znalosti přesnosti výstupních dat laserového skeneru.

r_1c	0	0	0	⋯	0	0
0	b_1d	0	0	⋯	0	0
0	0	r_2c	0	⋯	0	0
0	0	0	b_2d	⋯	0	0
⋮	⋮	⋮	⋮	⋱	⋮	⋮
0	0	0	0	⋯	r_nc	0
0	0	0	0	⋯	0	b_nd

Obr. 2.10: Kovarianční matice **R**

Proces predikce a korekce

Nyní již můžeme přistoupit k popisu celého procesu predikce a korekce EKF.

1) Predikce stavového vektoru $\boldsymbol{x}$:

$$\boldsymbol{x}(k+1|k) = \boldsymbol{x}(k|k) + \boldsymbol{u}(k+1) \qquad 2.1$$

kde $\boldsymbol{x}(k+1|k)$ znamená predikci stavového vektoru pro krok $k+1$ na základě stavového vektoru $\boldsymbol{x}(k|k)$ v kroku k a vstupu z měření odometrie pro krok $k+1$

2) Predikce oblastí kovarianční matice $\boldsymbol{P}$ vztahující se k pozici robota:

$$\boldsymbol{P_{rr}}(k+1|k) = \boldsymbol{A}\boldsymbol{P_{rr}}(k|k)\boldsymbol{A}^T + \boldsymbol{Q} \qquad 2.2$$

kde $\boldsymbol{P_{rr}}$ odpovídá submatici pro kovarianci pozice robota (na Obr. 2.6 značená jako buňka **A**). Matice $\boldsymbol{A}$ a $\boldsymbol{Q}$ jsou také aktualizovány na základě vstupů z odometrie. Značení kroků k analogicky jako v rovnici 2.1

$$\boldsymbol{P_{ri}}(k+1|k) = \boldsymbol{A}\boldsymbol{P_{ri}}(k|k) \qquad 2.3$$

kde $\boldsymbol{P_{ri}}$ odpovídá submaticím pro kovariance mezi pozicí robota a jednotlivými orientačními body (na Obr. 2.6 značené jako buňky **D, E**, …). Tuto operaci provádíme pro všechny orientační body (tedy celé první tři řádky/sloupce matice $\boldsymbol{P}$).

3) Výpočet inovační matice $\boldsymbol{S}$ na základě vektoru orientačních bodů $\boldsymbol{z}$ pozorovaných v kroku $k+1$:

$$\boldsymbol{S} = \boldsymbol{H}\boldsymbol{P}(k+1|k)\boldsymbol{H}^T + \boldsymbol{R} \qquad 2.4$$

kde matice $\boldsymbol{H}$ a $\boldsymbol{R}$ jsou aktualizovány na základě vektoru $\boldsymbol{z}(k+1)$.

4) Výpočet Kalmanova zesílení $\boldsymbol{K}$ na základě nové inovační matice $\boldsymbol{S}$:

$$\boldsymbol{K} = \boldsymbol{P}(k+1|k)\boldsymbol{H}\boldsymbol{S}^{-1} \tag{2.5}$$

5) Korekce stavového vektoru $\boldsymbol{x}$ na základě pozorovaných orientačních bodů:

$$\boldsymbol{x}(k+1|k+1) = \boldsymbol{x}(k+1|k) + \boldsymbol{K}\big(\boldsymbol{z}(k+1) - \boldsymbol{H}\boldsymbol{x}(k+1|k)\big) \tag{2.6}$$

6) Korekce kovarianční matice $\boldsymbol{P}$ na základě pozorovaných orientačních bodů:

$$\boldsymbol{P}(k+1|k+1) = \boldsymbol{P}(k+1|k) - \boldsymbol{K}\boldsymbol{H}\boldsymbol{P}(k+1|k) \tag{2.7}$$

Tímto je celý jeden krok cyklu ukončen.

2.2 Dostupné softwarové nástroje

I přesto, že předchozí kapitola je pouze velmi stručným shrnutím celé problematiky SLAM, je zřejmé, že se jedná o dosti rozsáhlou oblast, jejíž problémy lze řešit na různých úrovních složitosti. Vzhledem k charakteru této práce bylo tedy rozhodnuto o využití již existujících softwarových nástrojů a řešení pro SLAM, které by bylo možné dále modifikovat a rozšířit pro naše potřeby. Bylo by samozřejmě možné vytvořit celý algoritmus a všechny jeho nezbytné součásti od začátku, což by ovšem zabralo velkou část celé práce na úkor dalších cílů.

2.2.1 Volba prostředí MATLAB

Jako hlavní programovací prostředí byl pro tuto práci vybrán MATLAB od firmy MathWorks. Důvody této volby jsou zejména tyto:

- široké spektrum funkcionalit, které je možné rozšiřovat pomocí tzv. toolboxů
- efektivní práce s maticemi (MATLAB = matrix laboratory)
- jednoduchý programovací jazyk
- grafické prostředí Simulink podporující automatické generování kódu pro vestavěné mikroprocesory použité na vozidle Car4
- možnost rychlého a jednouchého grafického zobrazení dat
- kvalitní dokumentace a podpora, velká uživatelská komunita

Oproti jiným programovacím jazykům (například C++) bývá MATLAB zpravidla pomalejší, což je ovšem vyváženo rychlejším vývojem celé aplikace. Další možností je například provést vývoj a ladění aplikace v MATLABu a poté ji přepsat do C++, který ji bude vykonávat rychleji.

2.2.2 Speciální software

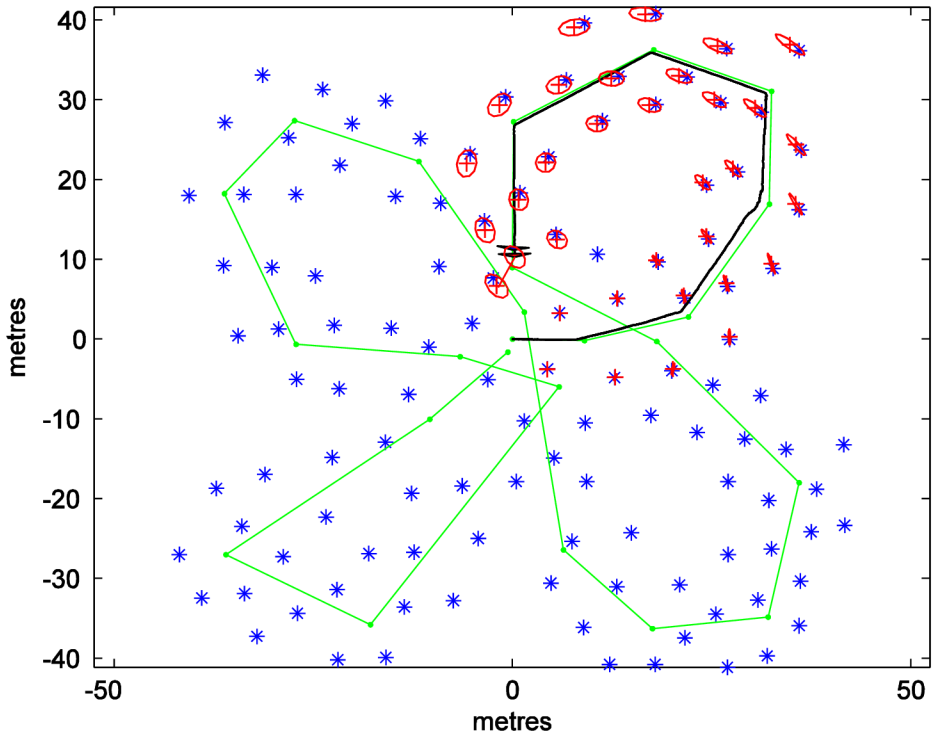
V této kapitole uvedeme některé ze softwarových nástrojů, jež byly více či méně prozkoumány v rámci hledání vhodných řešení pro aplikaci SLAM v této práci. Pro všechny níže uvedené případy platí, že jde o kód psaný přímo v MATLABu nebo o funkce psané v jiném jazyce zkompilované tak, aby je bylo možné v MATLABu použít.

EKF-SLAM Simulator [12]

Tento simulátor je primárně určený pro testování různých mapovacích algoritmů. Pracuje na principu klasického EKF a je možné v něm vytvořit simulované prostředí, jehož vstupem jsou souřadnice orientačních bodů a trajektorie, kterou má robot projet. Tyto údaje je možné zadat buď manuálně v podobě dvou matic, nebo můžeme využít nástroje, ve kterém prostřednictvím grafického uživatelského rozhraní (GUI) zadáme jednotlivé body kliknutím na požadované místo na mapě.

Dále je možné nastavit množství parametrů jako rychlost a rozměry vozidla, míru nepřesnosti odometrie a měření polohy orientačních bodů, citlivost validační brány pro jejich asociaci, maximální rádius jejich detekce a mnoho dalších.

Po spuštění simulace je možné na mapě sledovat pohyb robota, jeho odchylku od požadované trajektorie nebo velikost elips reprezentujících oblast spolehlivosti kovariančních matic pro polohu robota a orientačních bodů (viz. Obr. 2.11).

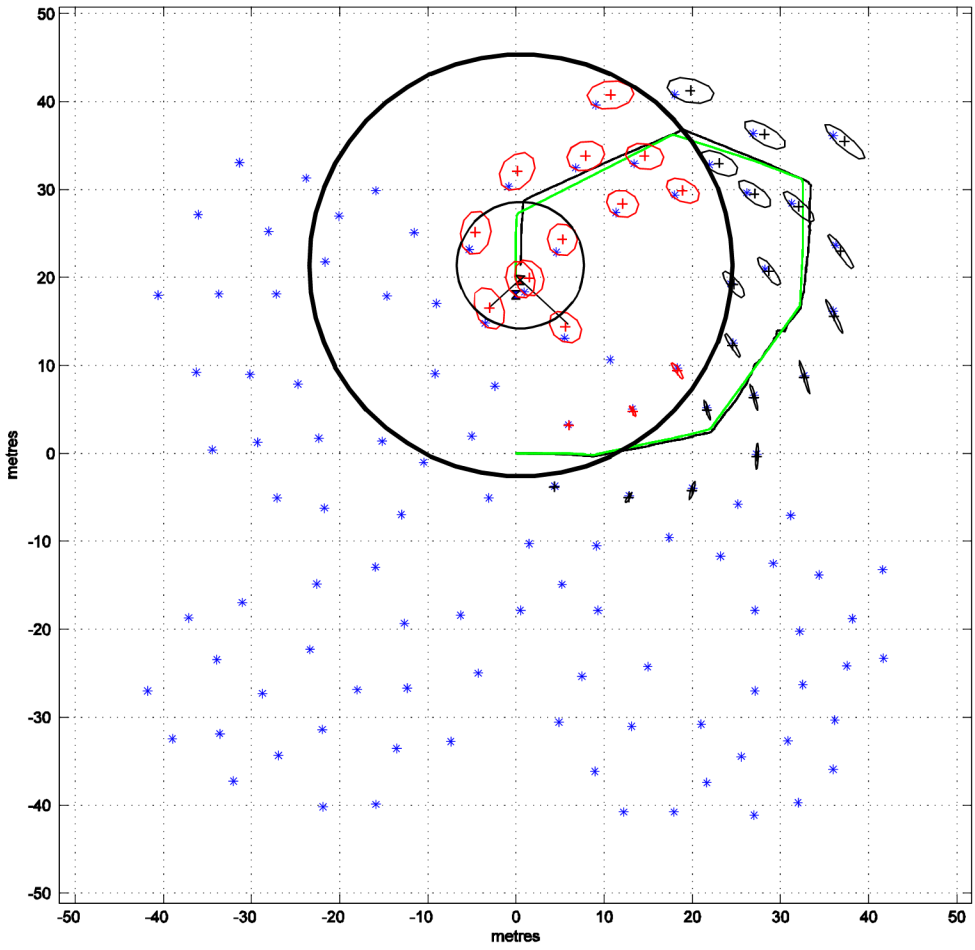


Obr. 2.11: EKF-SLAM Simulator (modrá - orientační body, zelená - žádaná trajektorie, černá - skutečná trajektorie, červená - oblasti spolehlivosti kovariančních matic pro jednotlivé orientační body a polohu robota)

CEKF-SLAM Simulator [13]

Tento program do značné míry využívá předchozí EKF-SLAM Simulator, ovšem rozšiřuje jej o použití tzv. Komprimovaného EKF SLAM algoritmu (anglicky Compressed Extended Kalman Filter-based SLAM). Tato modifikace se snaží řešit problém, kdy s rostoucím počtem detekovaných orientačních bodů roste velikost kovariančních matic a s tím i výpočetní náročnost celého algoritmu.

Princip CEKF algoritmu spočívá v rozdělení matic na aktivní a pasivní část, z nichž pouze aktivní je v každém kroku výpočtu aktualizována. Aktivní část je určována podle toho, v jakém sektoru mapy se robot právě nachází. Algoritmus tedy vždy pracuje pouze s orientačními body ve svém blízkém okolí a pouze při přechodu z jednoho sektoru do druhého provede kompletní aktualizaci všech bodů. Tím dochází ke značnému zrychlení celého výpočtu - v případě srovnání s EKF-SLAM byl CEKF-SLAM rychlejší až o 100% při použití shodných simulačních parametrů a počtem orientačních bodů 117.

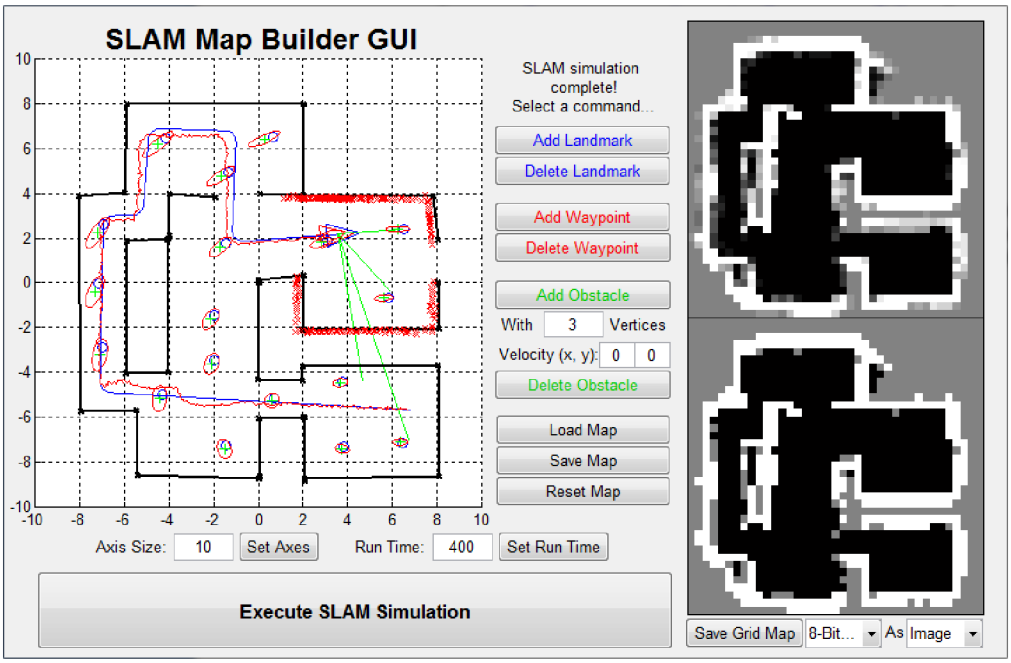


Obr. 2.12: CEKF-SLAM Simulator (modrá - orientační body, zelená - žádaná trajektorie, černá linka- skutečná trajektorie, černý kruh (velký) - aktuální aktivní oblast, červená/černá elipsa - oblasti spolehlivosti kovariančních matic pro jednotlivé aktivní/pasivní orientační body a polohu robota)

SLAM Map Builder GUI [14]

SLAM Map Builder je propracovanější alternativou předchozích dvou simulátorů. V rámci jeho uživatelského rozhraní lze kromě orientačních bodů a trajektorie, nadefinovat také vzhled prostředí pomocí překážek různých tvarů, reprezentujících například zdi místnosti. Program navíc simuluje i výstupy z měření laserového skeneru, pomocí nichž pak vytváří mapu okolí v podobě mřížky obsazenosti (Obr. 2.13 - vpravo).

Stejně jako v předchozích dvou případech, i zde lze provést nastavení mnoha parametrů algoritmu.

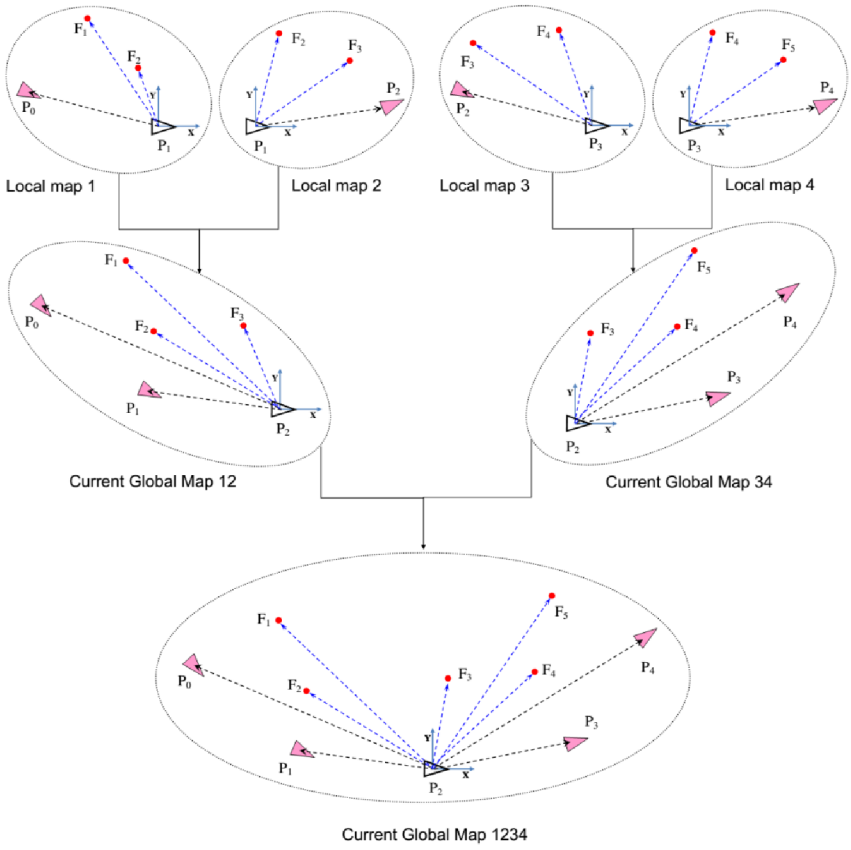


Obr. 2.13: SLAM Map Builder GUI

Linear SLAM [15]

Dalším nástrojem je Linear SLAM, který se zaměřuje na problém vytváření velkých map pomocí skládání lokálních submap. Program již neřeší vytváření těchto submap a je nutné mu je dodat jako vstupní data. Pro demonstrační účely obsahuje Linear SLAM i několik sad testovacích submap.

Proces skládání map je sekvencí řešení lineárního problému nejmenších čtverců. Toto skládání může probíhat buď sekvenčně, nebo metodou "rozděl a panuj". Při sekvenčním skládání rychlost algoritmu s rostoucí velikostí mapy (a počtem orientačních bodů) rapidně klesá. Metoda "rozděl a panuj" funguje na odlišném principu, kdy vždy dvě vstupní submapy jsou spojeny do submapy vyšší úrovně dokud není vytvořena finální mapa (nejvyšší úroveň). Princip tohoto postupu je patrný z Obr. 2.14. Tím se snižuje výpočetní náročnost oproti běžnému sekvenčnímu skládání, kdy velikost aktuální globální mapy, která má být spojena s následující lokální submapou, neustále roste.



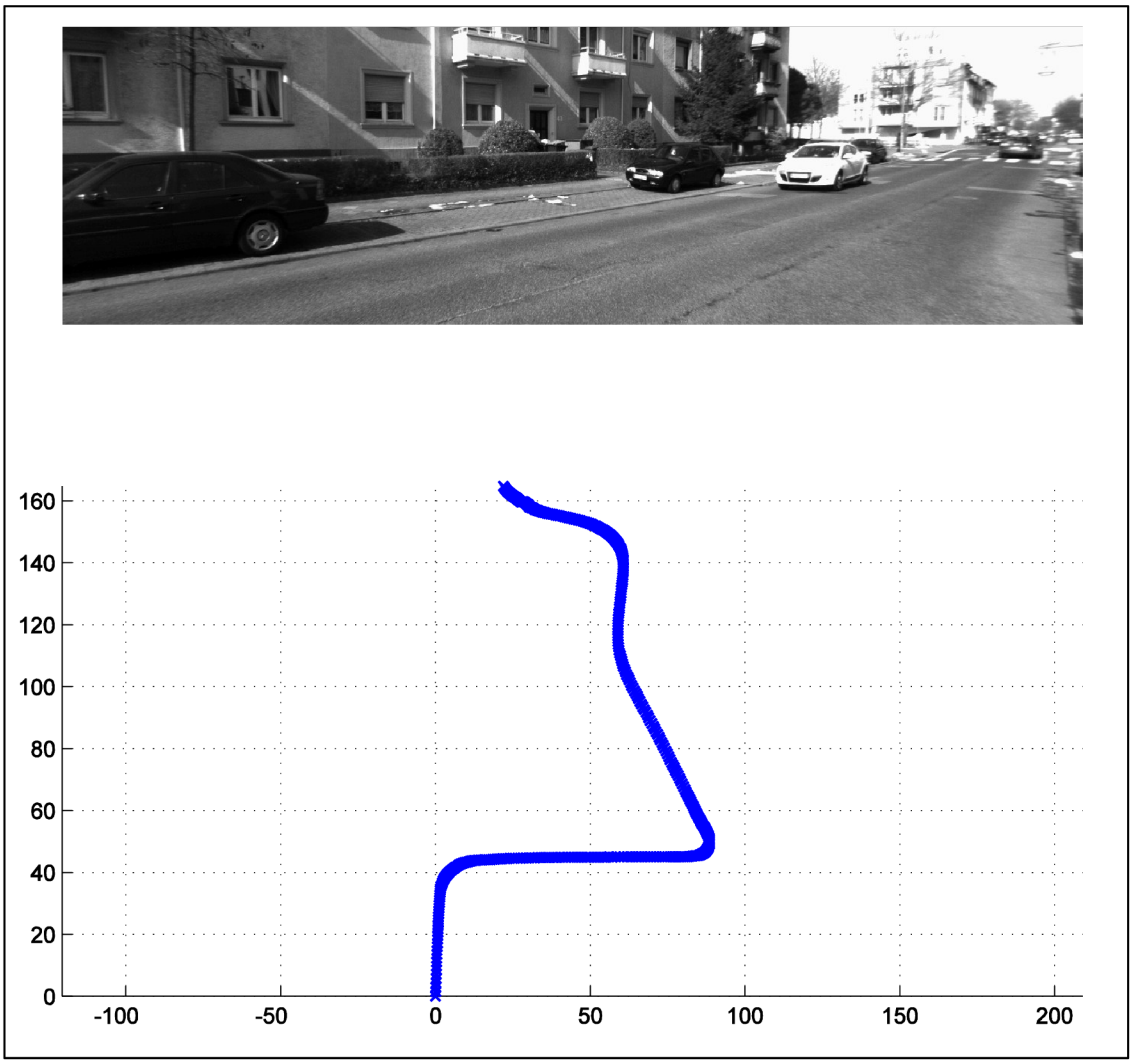
Obr. 2.14: Princip metody "rozděl a panuj" použité v programu Linear SLAM [16]

LIBVISO2 (Library for Visual Odometry 2) [17]

Na rozdíl od všech výše zmíněných nástrojů, se program LIBVISO2 nezabývá řešením problematiky SLAM. Jde o alternativu ke klasickému měření odometrie, která využívá dat z digitální kamery pro výpočet její změny polohy a orientace (více v kapitole 2.3.1). LIBVISO2 obsahuje funkce jak pro práci s jednou kamerou, tak pro zpracování stereovize (snímání scény dvěma kamerami současně). Vstupem je série jednotlivých monochromatických obrazů z kamery.

V rámci této práce byla otestována pouze verze pro práci s jednou kamerou. U té je vzhledem k menšímu počtu vstupních informací oproti stereovizi nutné, aby byla kamera během pohybu umístěna ve známé a fixní výšce nad zemí. Dále je také nutné znát některé základní parametry jako například ohnisková vzdálenost.

Příklad výstupu z tohoto nástroje pro demonstrační data z jízdy po městě je na Obr. 2.15.



Obr. 2.15: Příklad ujeté dráhy vozidla vypočítané pomocí nástroje LIBVISO2 pro vizuální odometrii

2.2.3 CAS-toolbox [18]

CAS-toolbox je posledním ze zkoumaných softwarových řešení SLAM a vzhledem k tomu, že byl nakonec vybrán jako hlavní program pro praktickou realizaci na vozidle Car4, je mu také věnována vlastní kapitola.

Jedná se komplexní nástroj, jehož cílem je co největší univerzálnost a uživateli nabízí možnost přizpůsobení jednotlivých částí pro konkrétní aplikaci. Díky tomu, že jde o objektově orientovaný design, je možné velmi jednoduše provést například změnu třídy robota (různé konstrukce a modely), nebo metodu extrakce orientačních bodů.

Hlavní výhody tohoto toolboxu jsou:

- import vstupních dat v různých formátech (při zachování určité základní formy)
- práce s větším počtem senzorů
- jednoduchá změna metody extrakce orientačních bodů či naprogramování vlastního řešení
- jednoduchá změna modelu robota či naprogramování vlastního řešení
- sada funkcí pro snadné vykreslování objektů jako mapa, vozidlo, značka počátku a orientace souřadného systému, elipsa nejistoty, atd.

Nyní se podrobněji zaměříme na některé hlavní části CAS-toolboxu:

Import dat a časování

Jak již bylo zmíněno, toolbox je dosti flexibilní co do formátu vstupních dat ze senzorů. Importovaný soubor by měl být v textovém formátu a každé jedno měření by mělo být na novém řádku, opatřené na začátku názvem senzoru a časovou značkou (anglicky timestamp). Poté už mohou následovat data oddělená mezerou s libovolnou strukturou. Styl jakým se data čtou je definován v nastavení každého konkrétního experimentu. Data z více senzorů mohou být zapsána v jednom společném souboru nebo samostatně.

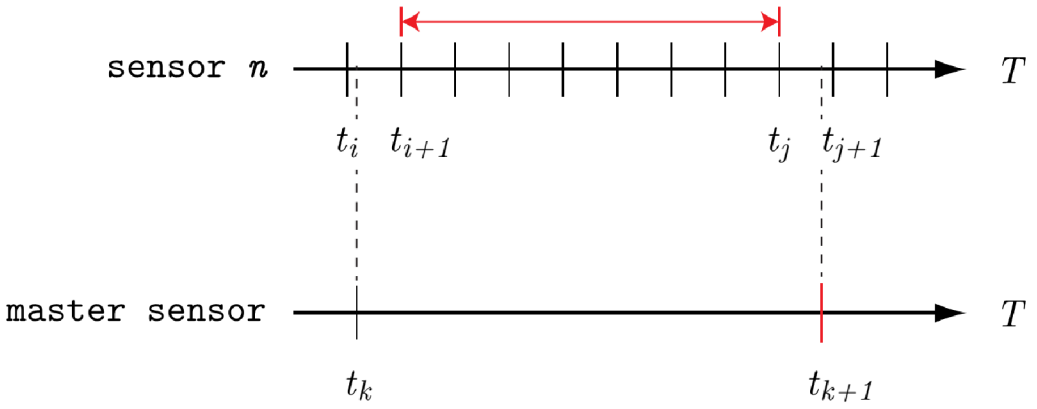
Příklad formátování souboru pro data z enkodérů (E) a laserového skeneru (S):

```
E 2290878 0.000000 0.000000
E 2291977 -0.000460 0.000460
E 2293077 -0.000460 0.000460
...
```

```
S 2425853 348 6.175731 -0.295176 9.913137 -0.760344 9.877998 ...
S 2472265 356 6.051506 -0.337071 5.053622 -0.394853 5.230943 ...
S 2519745 364 8.147476 -0.012257 8.093701 -0.157563 8.070460 ...
...
```

Dalším problémem který je nutné vyřešit při práci s různými senzory je časování celého algoritmu. To je řešeno volbou hlavního snímače, který poté udává "tempo" celého algoritmu. Zpravidla jím bývá laserový skener, ale není to podmínkou. Výpočet poté probíhá tak, že se v každém kroku načte jedno nové měření hlavního senzoru a dále všechna měření z ostatních senzorů, která proběhla v době od posledního kroku (tedy od předchozího měření hlavního senzoru). Z tohoto důvodu je tedy nutné mít všechna měření opatřená časovou značkou.

Tento princip je ilustrován na Obr. 2.16 kde v čase měření t_{k+1} hlavního senzoru (master sensor) jsou čtena všechna měření podřízeného senzoru (sensor n) v intervalu od t_{i+1} do t_j.



Obr. 2.16: Řešení čtení dat ze senzorů s různým časováním [19]

Nastavení experimentu

Pro každý experiment je potřeba kromě dodání vstupních dat, provést také nastavení parametrů určujících chování celého výpočetního algoritmu. Toto je realizováno pomocí skriptu, který do struktury "params" zapíše všechny potřebné údaje:

- Seznam a parametry všech senzorů včetně definice formátu dat ve vstupním souboru
- Definice hlavního senzoru
- Nastavení třídy robota a jeho specifických parametrů
- Nastavení metody extrakce orientačních bodů a jejich parametrů

Díky tomuto systému lze k jednomu experimentu uchovávat více souborů s různým nastavením, například pro jednodušší srovnání vlivu jednotlivých parametrů na chování algoritmu nebo pro jednodušší rekonstrukci dříve prováděných experimentů.

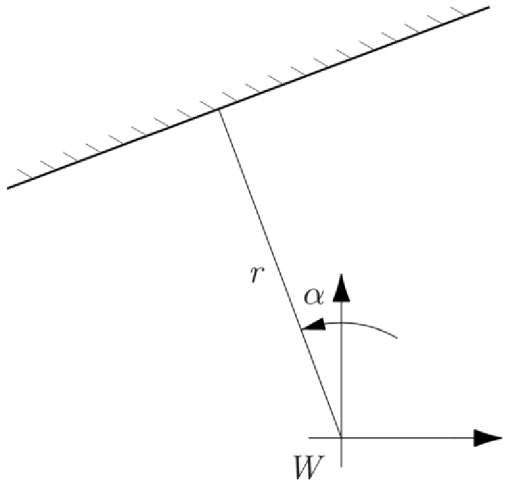
Zabudované metody extrakce orientačních bodů

Toolbox po instalaci obsahuje dvě připravené metody pro extrakci orientačních bodů. Jedná se o metodu extrakce majáků a extrakce přímek (viz. kapitola 2.1.3).

Extrakce majáků je poměrně přímočará funkce s minimem parametrů. Nastavit lze zejména minimální vzdálenost mezi majáky a minimální počet pozorování nutných k uznání jejich platnosti (eliminace chybných měření).

Extrakce přímek už je složitější funkcí a má také více možností nastavení. Metoda využívá postupu, kdy okno o konstantní velikosti prochází všemi měřeními laserového skeneru (z jednoho kroku) a pokouší o proložení dat přímkou. Poté záleží na parametrech metody, zda je přímka skutečně přijata jako orientační bod či nikoli. Mezi tyto parametry patří minimální délka prokládaného segmentu nebo limit přesnosti proložení. Metoda také umí provést fůzi několikanásobných detekcí stejné přímky do jediného orientačního bodu.

Tyto orientační body jsou poté reprezentovány jako nekonečné přímky s parametry α a r, kde r udává délku kolmice na přímku z počátku globálního souřadného systému a α je úhel který s ním tato kolmice svírá (viz. Obr. 2.17)



Obr. 2.17: Znázornění parametrů přímky jako orientačního bodu

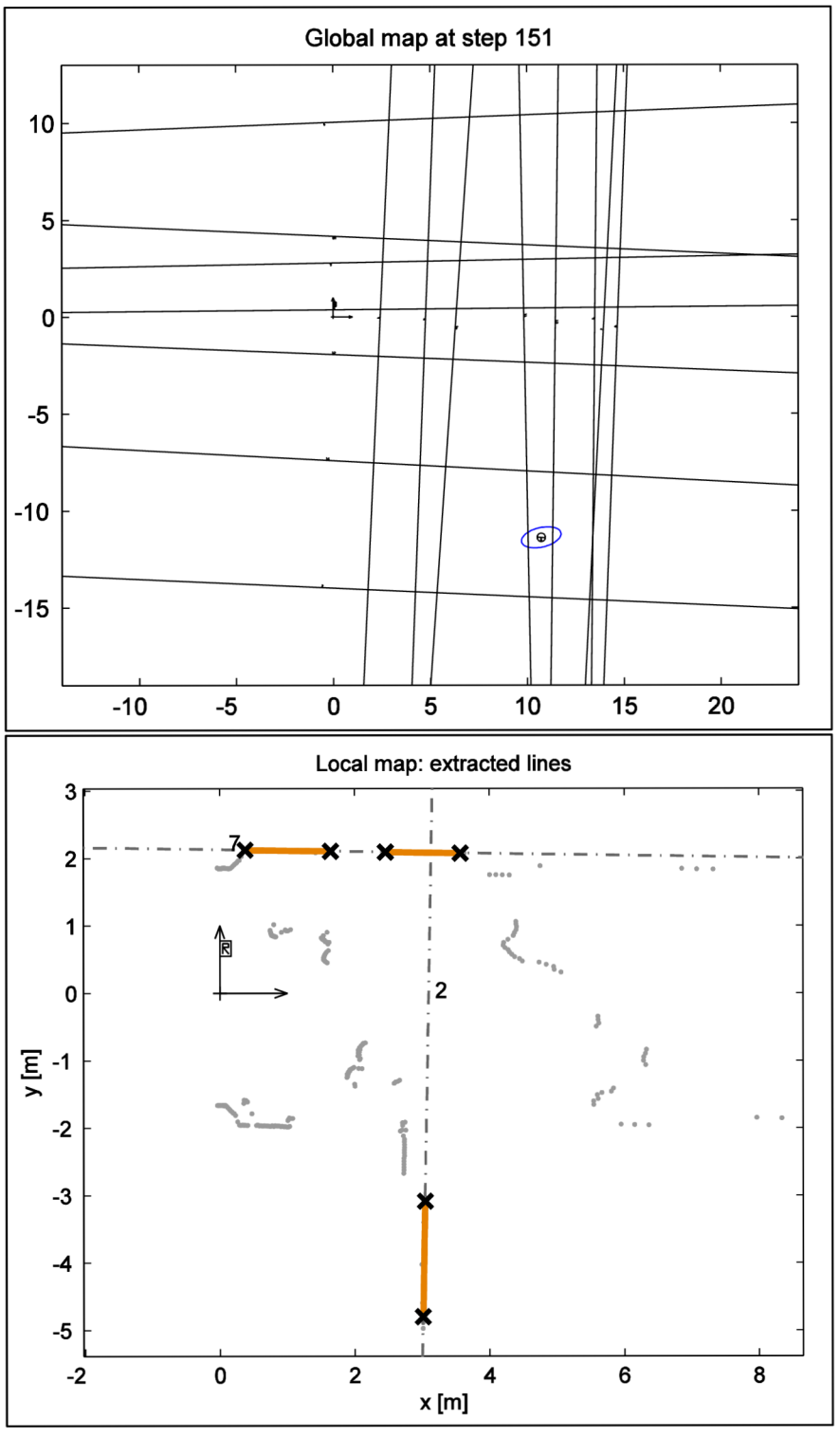
Zabudovaný model robota

Kromě extrakčních metod obsahuje toolbox také jeden model (třídu) robota. Jedná se o dvoukolového diferenciálně řízeného robota. Tento model bohužel nelze přímo aplikovat na vozidlo Car4 a z tohoto důvodu bude nutné vytvořit nový model, jehož součástí je i celý algoritmus predikce polohy robota na základě odometrie.

Proto se touto částí programu nebudeme v rešerši dále zabývat. Postup vytvoření nového modelu pro vozidlo Car4 je uveden v praktické části v kapitole 5.1.

Funkce SLAM

Jednou z hlavních funkcí toolboxu je funkce "slam.m", která zastřešuje celý proces. Stará se o čtení dat, volání metod, vykreslování mapy a další, vše na základě parametrů vygenerovaných při nastavení celého experimentu. Algoritmus této funkce je v podstatě jednoduchý, dochází při něm k postupnému provádění jednotlivých kroků EKF tak, jak byly popsány v kapitole 2.1.4. Jejím výstupem je pak globální mapa s vyznačenými orientačními body a aktuální polohou vozidla. Je také možné nechat si v každém kroku vykreslit aktuální mapu plus znázornění lokální mapy včetně dat z laserového skeneru (či jiného snímače). Příklad takovýchto dvou map z jednoho kroku je na Obr. 2.18.



Obr. 2.18: Příklad globální (nahoře) a lokální (dole) mapy při použití přímek jako orientačních bodů.

2.3 Přehled použitých snímačů

Pro realizaci této práce bylo vybráno několik snímačů, které pracují na rozdílných principech. Tyto snímače jsme rozdělili do dvou základních kategorií, popsaných v následujících dvou kapitolách 2.3.1 a 2.3.2.

2.3.1 Senzory pro odometrii

Jak již bylo popsáno v kapitole 2.1, odometrie používá data ze senzorů pohybu k odhadu změny polohy vozidla v závislosti na čase. V případě této práce jsou k tomuto účelu využity následující snímače:

- Enkodéry
- Inerciální měricí jednotka (IMU)
- Kamera

Testováním přesnosti odometrie při použití výše uvedených snímačů a jejich kombinací a také problémem získávání dat a komunikace se zabývá kapitola 4. Níže je uveden stručný popis a princip fungování jednotlivých zařízení:

Enkodéry

Vozidlo Car4 je vybaveno čtyřmi rotačními enkodéry, které jsou součástí hnacích motorů Transmotec PD4266 [20]. Enkodéry jsou dvoukanálové s rozlišením 38 tiků na otáčku pro každý kanál. Díky převodu 14:1 (motor:kolo) dostaneme ve výsledku rozlišení 1064 tiků na jednu otáčku kola vozidla. Při známé velikosti průměru kol, bychom měli jednoduchým přepočtem získat ujetou vzdálenost:

$$s = \frac{n}{1064} \cdot \pi \cdot D \qquad 2.8$$

kde n je počet tiků za daný časový úsek, D je průměr kola vozidla a s je vzdálenost, kterou toto kolo urazilo.

Tento způsob měření ujeté vzdálenosti je teoreticky velmi přesný, zejména při vysokém rozlišení enkodéru, ovšem v praxi této přesnosti nikdy dosáhnout nemůžeme, zejména z důvodu možného prokluzování kol, deformacím pneumatiky vozidla, vůlím v převodech apod. Přesnost měření ujeté vzdálenosti pomocí enkodérů tedy velmi závisí na konstrukci vozidla a stylu jízdy.

Senzory mají digitální výstup a jsou připojeny na řídící jednotky vozidla Car4, které tyto signály dále zpracovávají a případně přeposílají dále.

Obr. 2.19: Motor PD4266 od firmy Transmotec se zabudovaným enkodérem (pod plastovým krytem v zadní části) [20]

Inerciální měřicí jednotka (IMU)

Další z použitých senzorů, jenž je možné použít pro odometrii je tzv. IMU jednotka (Inercial Measurment Unit). Tato zařízení v sobě integrují akcelerometry, gyroskopy a případně i magnetometry, jenž měří zrychlení, úhlovou rychlost a velikost magnetického pole ve třech osách. Fúzí dat z těchto senzorů je možné plně určit orientaci a rychlost pohybu vozidla v prostoru.

V našem případě používáme senzor MPU-9150 od firmy Invensense [21]. Jedná se o velmi malé zařízení s minimální spotřebou, takže je vhodné i do velmi malých a energeticky nenáročných aplikací. Komunikace probíhá přes rozhraní I^2C. Senzor má mnoho pokročilých funkcí jako například nastavení filtrace signálu, omezení citlivosti, nebo i fúze dat pomocí zabudovaného mikroprocesoru. V základním módu z něj lze vyčítat pro každou ze tří os (X, Y, Z) zejména tyto údaje:

- Zrychlení v rozsahu až ±16 [g]
- Úhlová rychlost v rozsahu až ±2000 [°/s]
- Síla magnetického pole v rozsahu až ±1200 [μT]

Rychlost vzorkování jednotlivých veličin rámci čipu, 8 kHz (1 kHz při filtraci signálu) a rychlost rozhraní I^2C až 400 kHz jsou více než dostatečné, jelikož v rámci této práce, kdy se údaje ještě dále zpracovávaly, nebylo potřeba rychlosti vyšší než přibližně 50 – 100 Hz.

Přístupů jak zpracovávat data z IMU pro získání odometrie je více, a vzájemnou korekcí signálů, kalibrací a pokročilými algoritmy pro jejich fúzi, je možné dosáhnout vysoké přesnosti. Vzhledem ke spíše přehledovému charakteru této práce v dané oblasti, jsme zvolili jednodušší přístup čtení jednotlivých signálů (zrychlení a úhlová rychlost) a jejich následné integrace v časové oblasti pro získání změny polohy a natočení:

$$\Delta s = \iint_{t_1}^{t_2} a\, dt \qquad 2.9$$

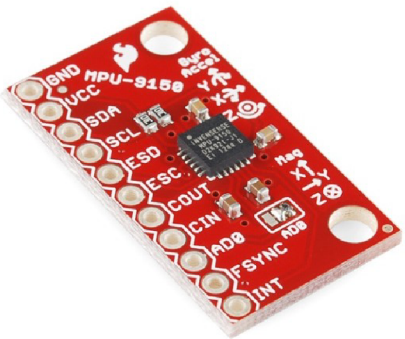
$$\Delta\theta = \int_{t_1}^{t_2} \omega\, dt \qquad 2.10$$

kde Δs a $\Delta\theta$ je změna polohy resp. natočení za daný časový úsek (t_1, t_2), a je zrychlení a ω úhlová rychlost měřené IMU jednotkou. Tyto vztahy platí pro jednu osu měření (X, Y, Z).

I v tomto případě jsou odometrická data zatížena chybou, která plyne z nepřesností výstupních dat IMU jednotky. Pro získání co nejlepších výsledků, musíme řešit zejména tyto tři problémy:

- **Offset měřené veličiny**
  Výstupní hodnoty zpravidla nejsou v klidové poloze snímače nulové. Je tedy třeba určit offset - odchylku měřené veličiny od nuly, když je snímač v klidu (nepohybuje se a nepůsobí na něj žádná síla, kromě gravitační).
- **Drift měřené veličiny**
  Výstupní hodnoty s časem pomalu rostou/klesají, i když je snímač v klidu. Je třeba zjistit velikost "driftu" - tedy jak rychle se výstupní hodnota odchyluje od počáteční hodnoty, když je snímač v klidu.
- **Šum měření**
  Ve výstupních datech je běžně přítomen šum. Je tedy nutné zjistit jeho velikost, případně jeho závislost na dalších faktorech.

Všechny tyto problémy přispívají k výsledné chybě odometrie a proto je nutné je co nejlépe analyzovat a na základě toho eliminovat jejich dopady. Situace je v této konkrétní aplikaci o to kritičtější, že při integrování signálů (zejména u akcelerometru) se veškeré chyby projeví mnohem výrazněji.



Obr. 2.20: Destička se senzorem MPU-9150 [22]

Kamera (vizuální odometrie)

Využití kamery pro získání údajů o změně polohy a natočení spadá do oblasti tzv. vizuální odometrie. Při ní dochází ke zpracování sekvence obrazů z digitální kamery, hledání společných znaků mezi dvěma obrazy a výpočtu posunutí a natočení vozidla na základě transformace nalezených společných znaků.

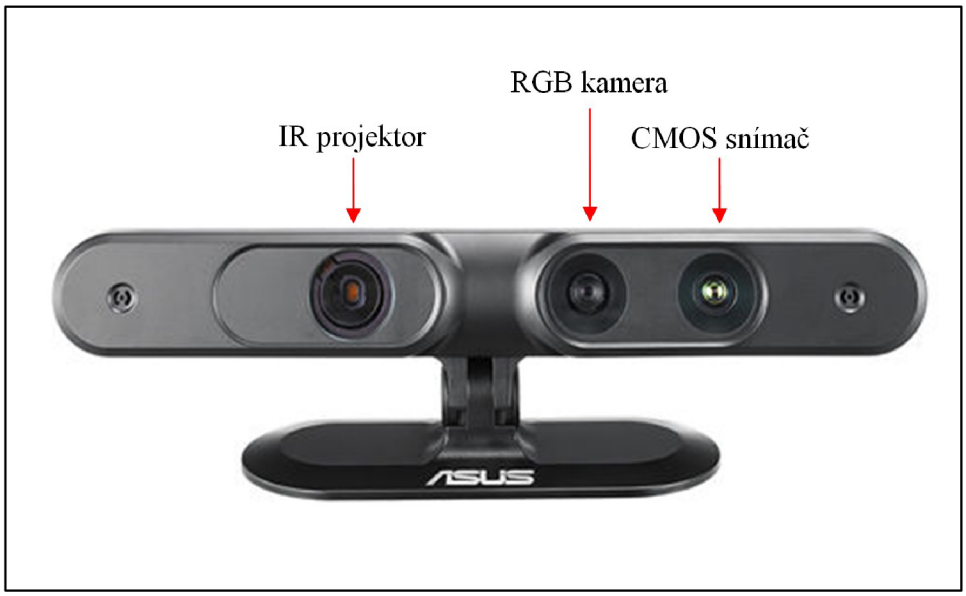
K tomu je zapotřebí znát vlastnosti použité kamery (např. ohnisková vzdálenost) a případně provést její kalibraci, pro eliminaci optických vad.

Tento způsob získání odometrie je oproti předchozím přístupům výhodnější například v tom, že není závislý na konstrukci samotného vozidla (výhodné u vozidel se složitým kinematickým/dynamickým modelem) nebo netrpí chybami měření jako offset nebo drift u IMU jednotky.

Na druhou stranu je vizuální odometrie díky nutnosti zpracovávat obraz náročnější na výpočetní výkon, je třeba korigovat optické vady kamery a v neposlední řadě je tato metoda závislá na okolních podmínkách jako je osvětlení a jeho rychlé změny nebo třeba dostatečná "pestrost" scény pro nalezení co největšího počtu společných znaků v obrazech.

V případě této práce, bylo pro experimenty s vizuální odometrií využito kamery zabudované ve snímači Asus Xtion PRO LIVE, což je zařízení téměř totožné se známějším produktem Kinect od firmy Microsoft. Dalším funkcím zařízení Xtion se věnuje kapitola 2.3.2.

Kamera v zařízení Xtion snímá barevný obraz (RGB) v rozlišení 640x480 obrazových bodů s maximální frekvencí 30 snímků za sekundu. Zařízení je určeno primárně k připojení k počítači či notebooku a komunikuje přes USB rozhraní.



Obr. 2.21: Snímač Asus Xtion PRO LIVE [23]

2.3.2 Senzory pro snímání okolí

Senzory pro snímání okolního prostředí jsou druhou kategorií snímačů, použitých v této práci. Na rozdíl od senzorů pro odometrii se však nezabývají pohybem samotného vozidla, ale detekcí překážek v zorném poli a dosahu snímače (viz. kapitola 2.1).

Pro potřeby této práce byly vybrány tyto snímače:

- Leddar M16
- Asus Xtion PRO LIVE
- Hokuyo URG-04LX-UG01
- Robopeak RPLIDAR

Tato kapitola se bude opět zabývat pouze stručným popisem těchto snímačů, komunikací a testováním přesnosti se zabývá kapitola 4.

Leddar M16

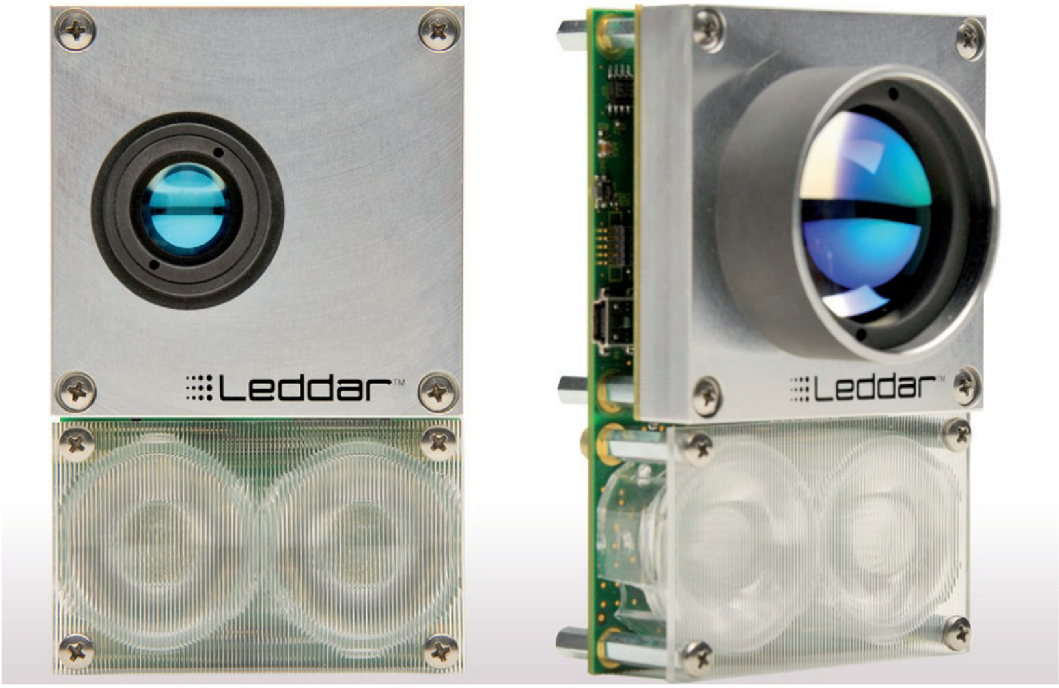
Leddar M16 (model LED-MOD-45-10) od firmy LeddarTech je multisegmentový snímač vzdálenosti, který měří vzdálenost od překážek pomocí výkonných infračervených diod sloužících jako vysílače a 16 kanálového fotodetektoru. Měření vzdálenosti probíhá na principu "time-of-flight" [24].

Výsledkem je 16 měření v jednotlivých segmentech, z nichž každý představuje výseč o šířce 2.8° a výšce 7.5° (celková šířka zorného pole 45°). Výhodou dosah senzoru až 100 m, rychlost měření až 50 Hz, přesnost do 5 cm nebo také absence pohyblivých částí [25].

Zařízení má také k dispozici několik různých rozhraní - USB, UART, RS-485 a CAN.

Funkcionalita Leddaru byla zběžně otestována za pomoci dodávané aplikace od výrobce a byla také vytvořena jednoduchá funkce v MATLABu pro měření dat. Nicméně vzhledem k faktu, že senzor měří vzdálenost pouze v 16 segmentech, což je pro lokalizaci a mapování málo, bylo usouzeno, že není zcela vhodný pro použití v rámci této práce. Jeho potenciál by bylo možné využít v aplikacích vyžadujících rychlou reakci na bezprostřední nebezpečí, jako například vyhýbání se překážkám, zabránění kolize ve vyšších rychlostech nebo velmi hrubá analýza okolního prostoru.

Tímto zařízením se tedy nebude práce v dalších kapitolách zabývat.



Obr. 2.22: Senzor Leddar M16 od společnosti LeddarTech [25]

Asus Xtion PRO LIVE

Tento snímač byl použit a otestován v rámci bakalářské práce [2], na kterou tato práce částečně navazuje. Jeho využití pro aplikaci ve SLAM algoritmech se nabízí díky jeho schopnosti vytvářet prostorový obraz scény před sebou v zorném poli 58° horizontálně a 45° vertikálně. Tento obraz je na výstupu reprezentován maticí bodů 640x480x3 která pro každý obrazový bod (rozlišení 640x480) udává jeho polohu v kartézském souřadném systému (osy x, y a z).

Takto velké množství informací by mohlo být užitečné pro přesnou a komplexní analýzu snímané scény. Ovšem na to aby tato problematika mohla být probrána pouze jako podkapitola celé práce, se bohužel jedná o příliš rozsáhlou oblast. Z tohoto důvodu a z důvodu zachování formátu dat, kompatibilního s dalšími senzory v této kapitole, je z celého snímaného obrazu vybrán pouze jeden horizontální řádek reprezentující rovinu ležící v optické ose senzoru. Matice dat se tedy redukuje na velikost 640x1x3, přičemž souřadnice v ose y je nulová (viz. [2], obr. 3.3). I přes tuto poměrně razantní redukci získáme dostatečné množství dat pro další zpracování.

Stejně jako u RGB kamery v kapitole 2.3.1 je maximální rychlost snímání scény 30 Hz, přičemž data z hloubkové mapy a kamery jsou aktualizována současně. Dosah snímače se pohybuje v rozmezí od 60 do 350 cm, kde je chyba měření do 4 cm. Na větších vzdálenostech už přesnost exponenciálně klesá (viz. [2], kapitola 4.2.1). Jak již bylo zmíněno v předchozí kapitole, zařízení komunikuje přes USB rozhraní.

Hokuyo URG-04LX-UG01

Snímač Hokuyo byl, stejně jako v předchozím případě, použit již v předchozí práci v rámci níž byly také otestovány jeho vlastnosti [2]. Jedná se o laserový skener, který při každém jednom skenu provádí až 682 měření vzdálenosti v jedné rovině v zorném poli 240° (úhlové rozlišení 0.35°). Snímač pracuje na principu metody fázového posunu [24], díky čemuž vykazuje velmi nízkou chybu měření do 2 cm v celém rozsahu měření, tj. od 8 do 400 cm. Frekvence skenování je u Hokuya až 10 Hz a komunikace probíhá přes USB rozhraní.

Obr. 2.23: Laserový skener Hokuyo URG-04LX-UG01

Robopeak RPLIDAR

RPLIDAR je druhým laserovým skenerem použitým v této práci, avšak od senzoru Hokuyo se liší v mnoha ohledech.

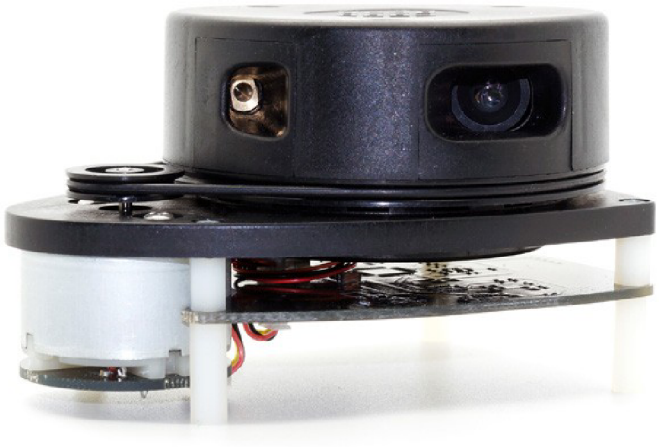
Prvním je už samotná konstrukce zařízení, jehož hlavní části tvoří rotující hlava skeneru ve tvaru válce, ve kterém je umístěn vysílač a přijímač infračerveného laserového paprsku a také většina elektroniky, dále hnací motor a nakonec modul zajišťující napájení a komunikační rozhraní. Zařízení je větší než Hokuyo a celkově působí méně profesionálním dojmem. Při rotaci hlavy skeneru je také znát mírné její mírné vyosení což by ovšem podle dosavadního pozorování nemělo mít vliv na měření.

Díky své konstrukci má RPLIDAR zorné pole celých 360° přičemž úhlové rozlišení není konstantní a jeho velikost je $\leq 1°$ [26]. Jeden sken typicky zahrnuje cca 360 měření. Na rozdíl od skeneru Hokuyo, nemá RPLIDAR pevně stanovené úhly jednotlivých měření, takže spolu s informací o vzdálenosti posílá senzor také informaci o aktuálním natočení paprsku.

Frekvence skenování je až 5,5 Hz s možností konfigurace až na 10 Hz (netestováno).

Také princip měření vzdálenosti je jiný než u Hokuya, protože RPLIDAR využívá geometrickou metodu triangulace [24]. Díky použití této metody nemá zařízení v celém svém rozsahu, od 15 do 600 cm, konstantní chybu měření. Ta roste spolu s měřenou vzdáleností a má nezanedbatelnou velikost [27]. Pro další použití zařízení je tedy třeba zjistit, zda je tato chyba měření kompenzována již v zařízení a pokud ne, bude nutné změřit její charakteristiku a provést kalibraci.

Komunikace se snímačem je možná pomocí rozhraní UART/USB.



Obr. 2.24: Laserový skener RPLIDAR od firmy Robopeak [28]

3 Formulace problému a cíle řešení

3.1 Aktuální stav vozidla Car4

Jak bylo zmíněno v úvodní kapitole, experimentální vozidlo Car4 vzniklo v roce 2010 v rámci několika diplomových a bakalářských prací v realizovaných v laboratoři Mechlab. Od té doby prošlo několika revizemi elektroniky i mechaniky a bylo doplněno o některé senzory a výpočetní hardware (palubní PC)

Takovýto byl stav vozidla na počátku a v průběhu realizace této práce:

- Každé ze 4 kol má vlastní nezávislý pohon a servořízení, v praxi je ovšem hnaná pouze přední náprava z důvodu revize části výkonové elektroniky
- O řízení motorů, servopohonů a případně i jiné výpočty se starají 3 řídící jednotky (dsPIC33FJ128MC804) - jedna hlavní a dvě podřízené
- K dispozici je palubní PC s operačním systémem Windows 7
- Vozidlo je možné ovládat pomocí dálkového ovládání nebo přes rozhraní UART

Řízení pohybu probíhá posíláním příkazů "rychlost" a "natočení", přičemž "rychlost" odpovídá momentu hnacích motorů a "natočení" odpovídá poloze okamžitého středu otáčení vozidla, na základě něhož se počítá natočení kol podle Ackermannovy kinematiky (viz. [29], kapitola 3.4.2).

Vzhledem k tomu, že implementace algoritmů SLAM má být realizována v prostředí MATLAB, bylo rozhodnuto o využití palubního PC, aby bylo možné provádět výpočty v reálném čase s hardwarem umístěným přímo na vozidle. Podrobnou konfiguraci PC je možné nalézt v [2].

SLAM algoritmy bývají také často spojovány s určitou formou autonomního pohybu robota. Nicméně v rámci této práce bude řešena pouze problematika lokalizace a mapování, přičemž vozidlo Car4 bude řízeno člověkem s pomocí dálkového ovládání.

3.2 Formulace cílů

Na základě zadání práce, poznatků z rešeršní studie a s ohledem na dostupné prostředky a časovou náročnost jednotlivých problémů, byly zvoleny tyto cíle a rozpracovány nezbytné kroky k jejich dosažení:

Cíle práce:

- Vybrat z dostupných snímačů variantu, která by poskytovala co nejlepší vstupní data pro SLAM
- Vybrat vhodné softwarové řešení a použít jej pro aplikaci SLAM na vozidle Car4
- Provést testování celého navrženého systému v reálném indoor prostředí včetně zpracování dat v reálném čase
- Na základě výsledků testování provést případnou optimalizaci tak aby bylo dosaženo co nejvyšší přesnosti lokalizace vozidla v neznámém prostředí

Nezbytné kroky, které je třeba provést:

- zprovoznit všechny dostupné senzory, jejich komunikaci s palubním PC a zpracování surových dat do podoby použitelné ve SLAM algoritmech
- vyřešit přenos dat mezi palubním PC a hlavní řídící jednotkou Car4
- otestovat chování senzorů, zejména jejich specifické vlastnosti a omezení
- provést nezbytné úpravy kódu u vybraného softwarového řešení SLAM s ohledem na použití na vozidle Car4
- provést úpravy pro běh programu v reálném čase

4 Srovnání a výběr senzorů

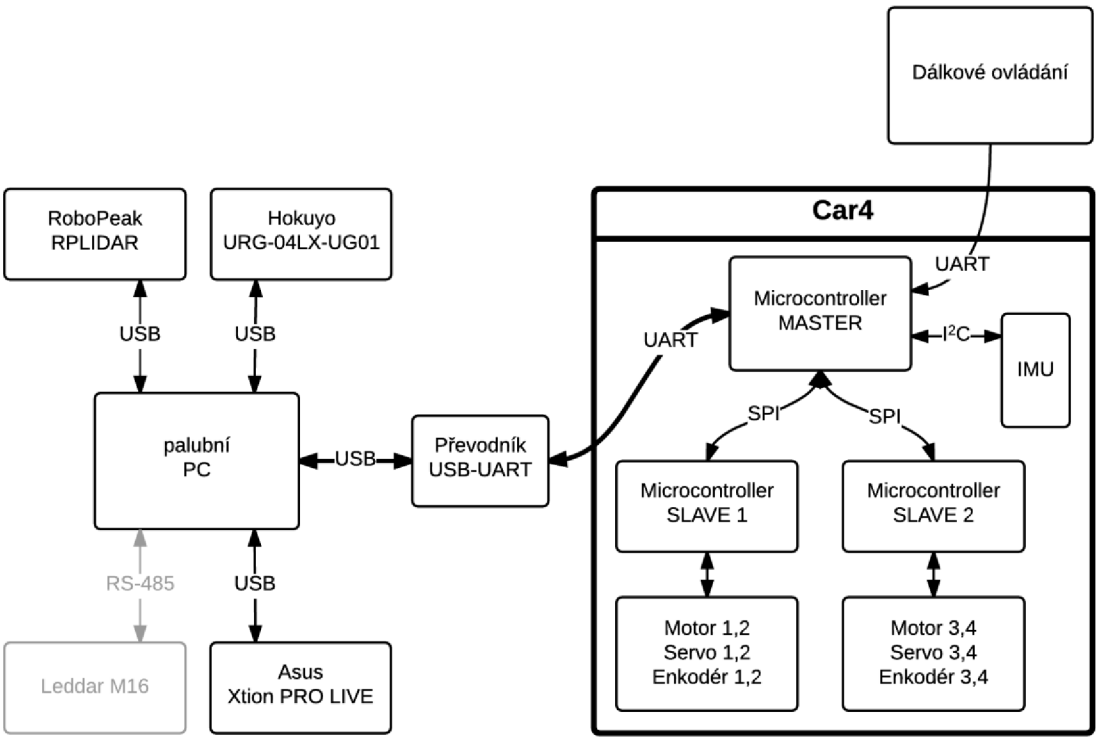
4.1 Komunikace a základní zpracování dat ze senzorů

4.1.1 Struktura a komunikace „senzory-vozidlo-PC“

Díky omezeným možnostem rozhraní u palubního PC bylo nutné vytvořit komunikační kanál mezi ním a hlavní řídící jednotkou vozidla Car4, která zpracovávala data z jednotlivých enkodérů a také k ní byla připojena IMU jednotka.

Schéma znázorňující všechna propojení je na Obr. 4.1. Je třeba zmínit, že zařízení RoboPeak, Hokuyo a převodník USB-UART se byla v PC reprezentována jako virtuální sériový COM port.

Komunikace s jednotlivými jednotkami je popsána v následující kapitole 4.1.2. Přenos dat mezi palubním PC a hlavní řídící jednotkou, probíhá tak, že PC pošle žádost o data v podobě zprávy "RTS" a řídící jednotka v odpovědi zašle aktuální měření z IMU, enkodérů a také údaj o natočení kol (viz. Obr. 4.4)



Obr. 4.1: Schéma komunikace v rámci vozidla Car4

4.1.2 Jednotlivé senzory – komunikace a základní zpracování dat

U většiny senzorů bylo v první řadě nutné napsat komunikační protokol buď přímo v MATLABu nebo generováním kódu pro hlavní mikroprocesor, které probíhalo v prostředí Simulink s využitím toolboxu MPLAB® Device Blocks for Simulink®.

Níže jsou uvedena řešení komunikace a případného zpracování dat ze všech použitých snímačů:

Asus Xtion PRO LIVE

V tomto případě bylo využito knihovny C++ funkcí OpenNI dodávané spolu se zařízením, které byly zkompilovány pro použití v MATLABU. Toto zařízení již bylo použito v předchozí práci [2] kde je také uveden podrobnější popis použití těchto funkcí.

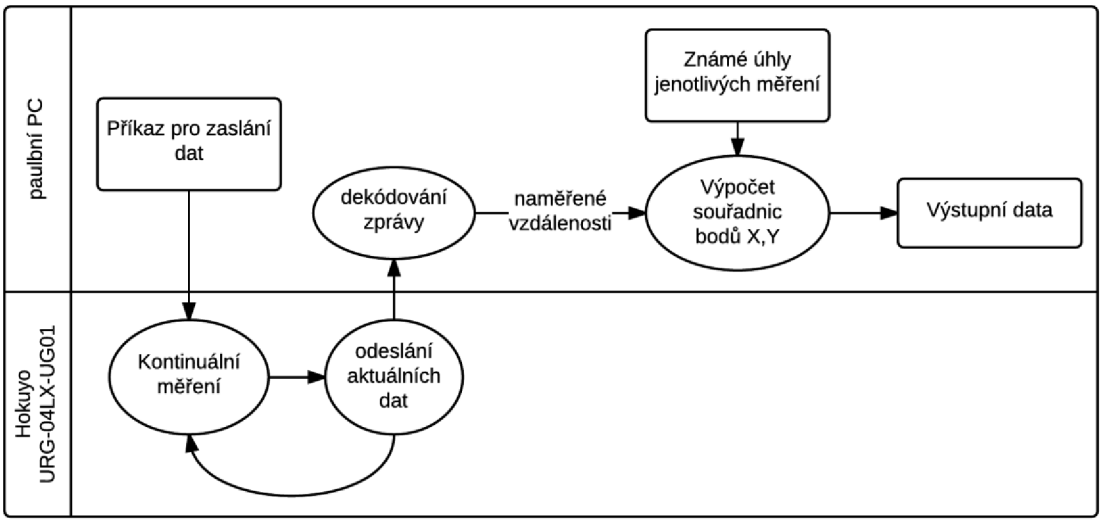
Hokuyo URG-04LX-UG01

Toto zařízení bylo taktéž použito již dříve v rámci [2], nicméně bylo zjištěno, že původní komunikační skript má mnoho nedostatků. I přesto, že výrobcem deklarovaná frekvence měření senzoru je 10 Hz, nebyl původní skript schopen vyčítat data rychlostí vyšší než 6 Hz. To bylo dáno nevhodným naprogramováním, které (mimo dalších nedostatků) nevyužívalo ukončovacích znaků "LF" (Line Feed) na konci zpráv přicházejících ze senzoru, čímž docházelo ke zbytečným prodlevám.

V rámci nového programu, který byl vytvořen již jako součást této práce, jsou všechny tyto problémy odstraněny a bylo dosaženo maximální možné rychlosti 10Hz. Jak bylo zmíněno výše, skript byl napsán v MATLABu, přičemž pro komunikaci se zařízením bylo využito virtuálního sériového portu.

Poté co je provedena inicializace přejde senzor do stavu, kdy neustále měří nová data. Pokud dostane patřičný příkaz, zašle aktuální sadu dat a pokračuje v měření (více o syntaxi příkazů v [2], kapitola 3.3.2). Přijatá data je nakonec nutné ještě dekódovat.

Senzor posílá vždy pouze informace o naměřené vzdálenosti v milimetrech, jelikož úhly paprsků jednotlivých měření jsou pevně stanoveny.



Obr. 4.2: Schéma komunikace PC-Hokuyo

Robopeak RPLIDAR

Pro tento senzor komunikační protokol v MATLABu z dřívější doby neexistoval a bylo tedy nutné vytvořit jej od začátku.

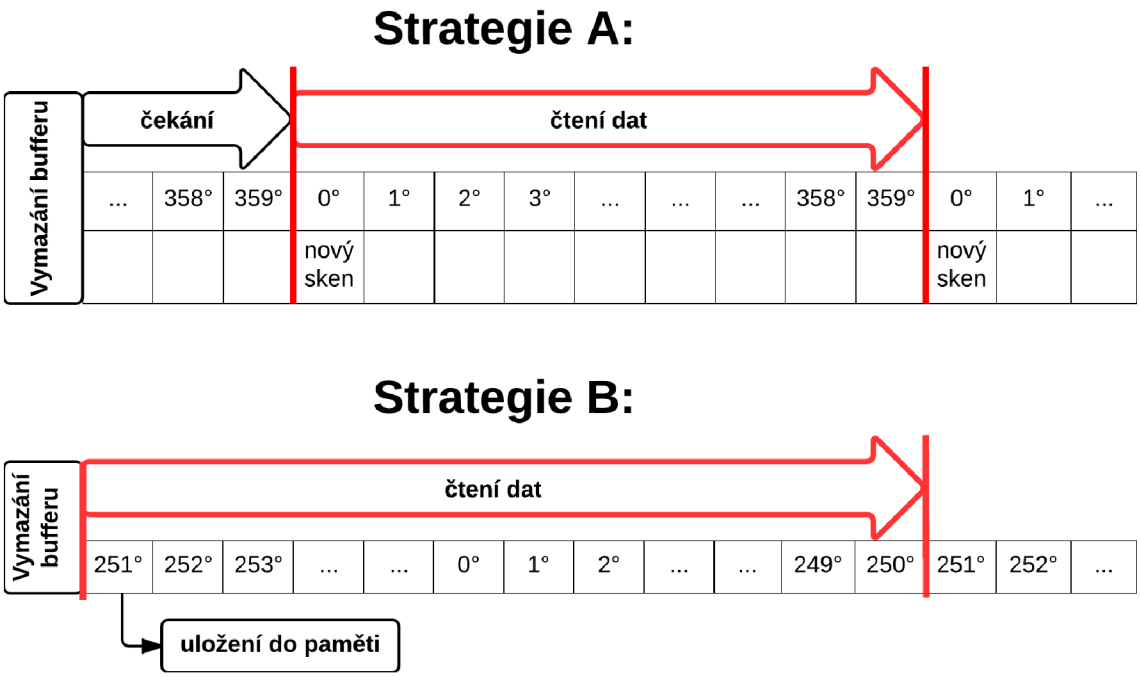
Největším potenciálním problémem u tohoto senzoru je fakt, že po inicializaci posílá naměřená data kontinuálně po jednotlivých krocích a ne v ucelených blocích na vyžádání jako to bylo v případě snímače Hokuyo.

V případě že po zahájení komunikace není možné vyčítat data stejně rychle jako jsou posílána, dojde po čase k přetečení vstupního bufferu na straně PC. To má za následek ztrátu informací a může způsobit chyby při čtení nekonzistentních dat. I kdyby byl buffer silně předimenzovaný, tak vyčítaná data nejsou časem kvůli zpoždění aktuální, přičemž senzor neposílá spolu s měřením žádnou časovou značku.

Dalším problémem je fakt, že se v posloupnosti dat vyskytují náhodné skoky. V ideálním provozu jsou data zasílána v takovém pořadí, že úhel natočení paprsku postupně roste, tedy například 1°,2°,…,359°,0°,1°,… Čas od času se ovšem v posloupnosti objevují skoky jako 10°,11°,5°,6°,12°,13°,… což může způsobit problémy při dalším zpracování dat.

Pro každé jedno měření zasílá RPLIDAR údaje o úhlu (ve stupních), naměřené vzdálenosti (v milimetrech), a také indikátor zda se jedná o začátek nového 360° skenu. Tento indikátor je možné využít pro rozdělení přijímaných dat do jednotlivých ucelených bloků.

Při vytváření komunikačního protokolu byly zvoleny dvě různé strategie, viz Obr. 4.3. Na začátku vždy dojde k vymazání bufferu aby byla zajištěna aktuálnost dat. Strategie A poté čeká na indikátor nového skenu a čte, dokud nepřijde další indikátor. Strategie B čte okamžitě, zapamatuje si úhel prvního měření a čte, dokud k tomuto úhlu znovu nedorazí.



Obr. 4.3: Srovnání dvou strategií pro čtení dat ze senzoru RPLIDAR

Výhodou strategie A je spolehlivost čtení dat, nevýhodou je určité zpoždění (teoreticky až jeden celý sken). Strategie B je sice rychlejší, ovšem je náchylná k různým chybám zejména kvůli výše zmíněným skokům v datech (což jde do značné míry ošetřit).

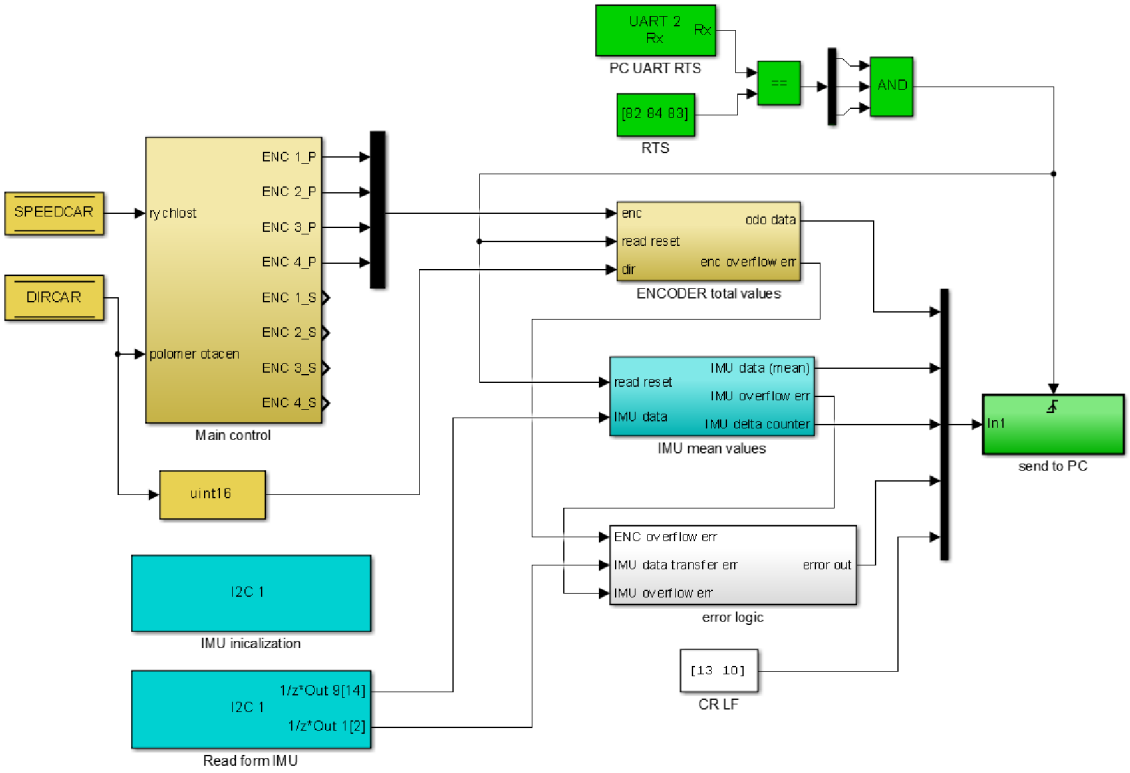
Při testování obou přístupů byla naměřená rychlost měření u strategie A 3,2 Hz a u strategie B 5,2 Hz. Proto byla nejprve upřednostňována varianta B, avšak po přetrvávajících problémech s chybami měření zvítězila díky své spolehlivosti varianta A. Později se ukázalo, že rychlost 3 - 3,5 Hz je pro funkci SLAM algoritmů dostatečná.

Inerciální měřicí jednotka (IMU)

Jak je vidět z Obr. 4.1, IMU je připojená na hlavní řídící jednotku vozidla Car4 přes rozhraní I^2C. Komunikace se senzorem byla tedy řešena v rámci hlavního programu této jednotky, který byl vytvořen generováním kódu ze Simulinku.

Zapisování a čtení dat probíhá poměrně jednoduše prostřednictvím jednotlivých registrů IMU. Při inicializaci je nutné vždy nejprve jednotku probudit z režimu spánku a provést případná další nastavení jako filtrace signálů, citlivosti snímačů nebo frekvence měření. Poté stačí pouze vyčítat údaje z registrů, do kterých jednotka nahrává nejnovější data.

Aby nedošlo ke ztrátě informací mezi jednotlivými žádostmi o měření ze strany PC, probíhá v řídící jednotce vzorkování a ukládání dat z IMU na frekvenci 1 kHz, přičemž při žádosti z PC dojde k výpočtu průměrné hodnoty všech veličin za dobu uplynulou od posledního odeslání dat. Blokové schéma celého procesu je na Obr. 4.4.



Obr. 4.4: Schéma programu hlavní řídící jednotky pro odesílání dat do PC (zelená - komunikace s PC, žlutá - data z enkodérů a natočení kol, modrá - data z IMU)

Data z IMU je nakonec v PC nutné převést na fyzikální jednotky, na základě nastavené citlivosti akcelerometru a gyroskopu, jelikož vyčítané hodnoty ze snímače jsou bezrozměrném znaménkovém datovém formátu int16. Pro výpočet ujeté vzdálenosti či změny úhlu orientace mezi jednotlivými kroky měření využijeme úpravy vztahů 2.9, 2.10 pro numerické řešení:

$$\Delta s_{k-1,k} = v_{k-1} \cdot \Delta t_{k-1,k} + \frac{1}{2} \cdot \overline{a_{k-1,k}} \cdot \Delta t_{k-1,k}^2 \tag{4.1}$$

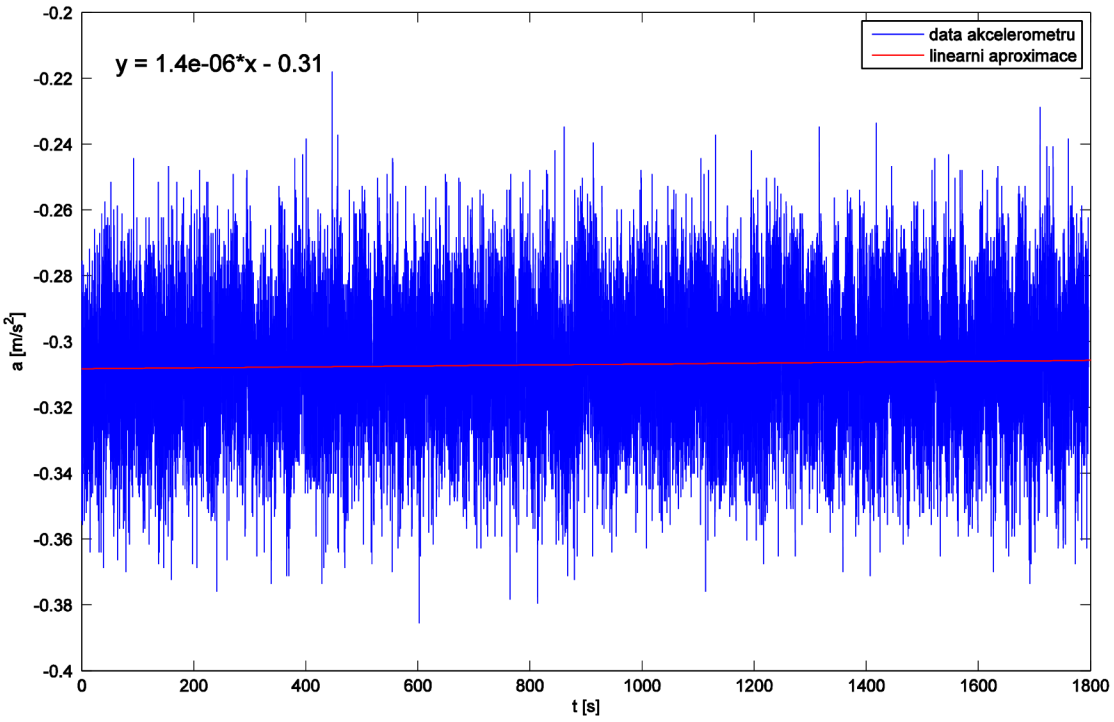
$$v_k = v_{k-1} + \overline{a_{k-1,k}} \cdot \Delta t_{k-1,k} \tag{4.2}$$

$$\Delta\theta_{k-1,k} = \overline{\omega_{k-1,k}} \cdot \Delta t_{k-1,k} \tag{4.3}$$

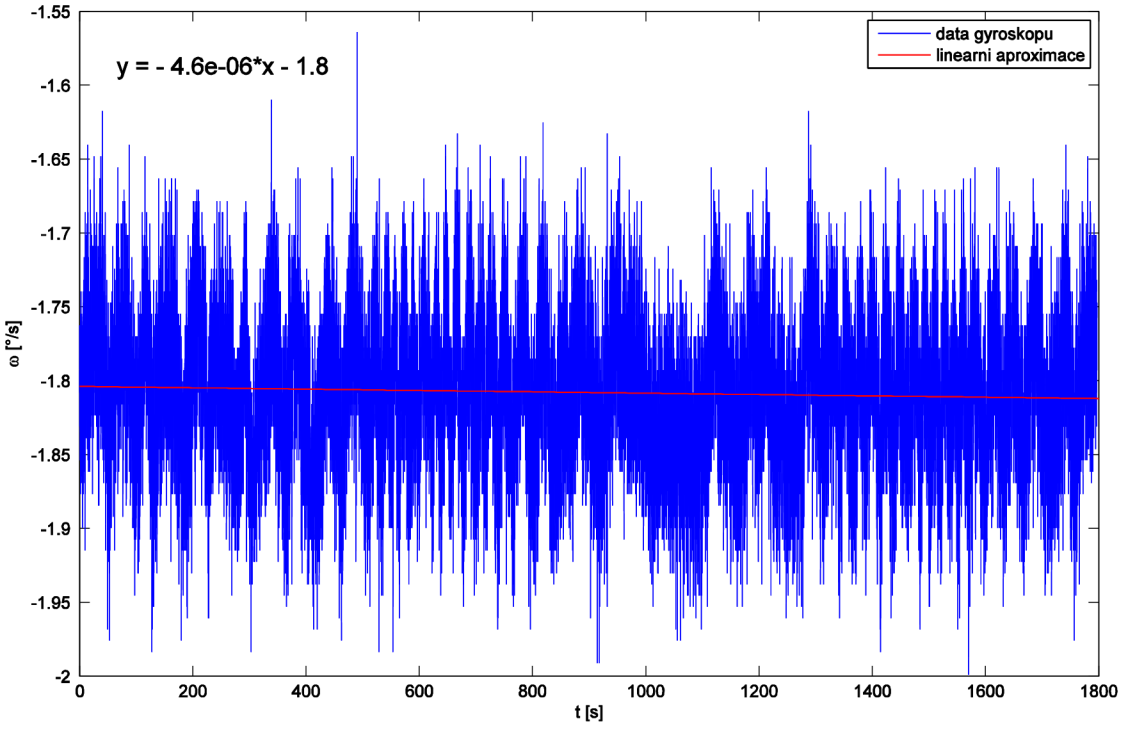
kde

v_{k-1}, v_k	rychlost v minulém (*k-1*) resp. aktuálním (*k*) kroku
$\Delta s_{k-1,k}$	vzdálenost uražená mezi minulým a aktuálním krokem
$\Delta\theta_{k-1,k}$	změna úhlu orientace mezi minulým a aktuálním krokem
$\Delta t_{k-1,k}$	doba mezi minulým a aktuálním krokem
$\overline{a_{k-1,k}}$	průměrná hodnota zrychlení mezi minulým a aktuálním krokem
$\overline{\omega_{k-1,k}}$	průměrná hodnota úhlové rychlosti mezi minulým a aktuálním krokem

U IMU je také nutné prověřit, nakolik se u ní projevují typické problémy jako drift a offset měřené veličiny. Z tohoto důvodu bylo provedeno měření údajů z akcelerometru a gyroskopu o délce 30 minut a se vzorkováním 10 Hz, přičemž senzor byl po celou dobu v klidu.



Obr. 4.5: Průběh zrychlení akcelerometru IMU jednotky



Obr. 4.6: Průběh úhlové rychlosti gyroskopu IMU jednotky

Jak můžeme vidět z Obr. 4.5 a Obr. 4.6, u obou veličin je patrný určitý offset, který je tedy nutné změřit (nejlépe v rámci počáteční inicializace senzoru na začátku všech testů) a kompenzovat. Také jistý nepatrný drift je zde pozorován, ovšem jeho velikost je v obou případech zanedbatelná (viz. rovnice ve výše uvedených grafech), když vezmeme v úvahu, že délky testů se pohybují v řádu maximálně desítek minut.

Enkodéry a natočení kol

Enkodéry jsou fyzicky napojeny na podřízené řídící jednotky, v nichž probíhá i zpracování jejich digitálních signálů. Výstupem, který je dále přeposílán do hlavní řídící jednotky je neznaménkové číslo o velikosti 16 bitů (uint16), které udává aktuální natočení motorů jako počet tiků enkodérů.

V hlavní řídící jednotce pak na základě těchto údajů probíhá výpočet celkového potočení motorů od chvíle poslední žádosti o data ze strany PC. Při následující žádosti jsou tyto údaje spolu s informací o aktuálním natočení kol odeslána do palubního počítače.

Data je nakonec opět nutné přepočítat na ujetou vzdálenost podle vztahu 2.8. Pro výpočet uražené dráhy mezi dvěma kroky (žádostmi o data) tedy platí:

$$\Delta s_{k-1,k} = \frac{\Delta n^{A}_{k-1,k} + \Delta n^{B}_{k-1,k}}{2} \cdot \frac{1}{1064} \cdot \pi \cdot D \qquad 4.4$$

kde

$\Delta s_{k-1,k}$	vzdálenost uražená mezi minulým a aktuálním krokem
$\Delta n^{A}_{k-1,k}$	počet tiků ekodéru mezi minulým a aktuálním krokem (levý motor - A)
$\Delta n^{B}_{k-1,k}$	počet tiků ekodéru mezi minulým a aktuálním krokem (pravý motor - B)
D	průměr kola auta

4.2 Srovnání snímačů a výběr nejvhodnější kombinace

4.2.1 Odometrie - přehled a srovnání metod výpočtu

V rámci testování přesnosti odmetrie, bylo vyzkoušeno několik různých způsobů jejího měření za použití dat z ekodérů, natočení serv, IMU a obrazu z kamery. Celkem bylo srovnáváno pět metod:

1) Pouze údaje z IMU (zrychlení a úhlová rychlost)
2) Kombinace údajů z IMU a enkodérů
3) Údaje z ekodérů a natočení kol - využití kinematického modelu z [29]
4) Vizuální odometrie - zpracování obrazu z kamery

Pouze údaje z IMU

V tomto případě využijeme měření dopředného zrychlení vozidla (akcelerometr) a rychlost otáčení kolem vlastní svislé osy (gyroskop). Vstupem funkce jsou údaje o průměrném zrychlení a úhlové rychlosti mezi dvěma kroky $\overline{a_{k-1,k}}$, $\overline{\omega_{k-1,k}}$ a také údaj o době uplynulé mezi nimi $\Delta t_{k-1,k}$. S využitím výše uvedených rovnic 4.1 až 4.3 můžeme vypočítat polohu a orientaci vozidla v aktuálním kroku jako:

$$x_k = x_{k-1} + \Delta s_{k-1,k} \cdot \cos\theta_{k-1} \tag{4.5}$$

$$y_k = y_{k-1} + \Delta s_{k-1,k} \cdot \sin\theta_{k-1} \tag{4.6}$$

$$\theta_k = \theta_{k-1} + \Delta\theta_{k-1,k} \tag{4.7}$$

Kombinace údajů z IMU a enkodérů

Zde využijeme IMU pouze na výpočet změny orientace vozidla, avšak uraženou dráhu získáváme z údajů enkodérů.

Při jízdě do zatáčky se vozidlo pohybuje po kružnici, přičemž kola na vnitřní straně urazí menší vzdálenost než kola na vnější straně kružnice. Proto je třeba získat střední hodnotu průměrováním dat z vnitřních a vnějších kol. Bohužel v době měření byl jeden z enkodérů zadní nápravy nefunkční, tudíž výpočet probíhal pouze z údajů kol přední nápravy.

Vstupem do funkce jsou údaje z IMU o průměrné úhlové rychlosti mezi dvěma kroky $\overline{\omega_{k-1,k}}$ a době uplynulé mezi nimi $\Delta t_{k-1,k}$ a poté údaje z enkodérů přední nápravy $\Delta n^A_{k-1,k}$ a $\Delta n^B_{k-1,k}$. S využitím rovnic 4.3 a 4.4 pro výpočet ujeté dráhy, získáme údaje o aktuální poloze a orientaci stejně jako v předchozím případě (rovnice 4.5 až 4.7).

Údaje z ekodérů a natočení kol - využití kinematického modelu z [29]

V rámci jedné z předchozích diplomových prací na projektu Car4 byl v Simulinku vytvořen neholonomní kinematický model vozidla s Ackermannovou kinematikou natáčení všech čtyř kol. V modelu jsou vždy dvě kola jedné nápravy redukována na jedno imaginární "střední" kolo. Tento model byl převzat a využit jako další možnost výpočtu odometrie.

Vstupem do tohoto modelu jsou úhlové rychlosti změny směru natočení "středních" kol přední a zadní nápravy dále rychlost otáčení "středního" kola přední nápravy a také údaje o vzdálenosti mezi nápravami vozidla L a poloměru kol R (viz. Obr. 4.7).

Tyto údaje získáme ze známých úhlů natočení kol φ^A, φ^B a dat z enkodérů $\Delta n^A_{k-1,k}$, $\Delta n^B_{k-1,k}$ a $\Delta t_{k-1,k}$ pomocí následujících vztahů:

$$\varphi^{AB} = \frac{\varphi^A + \varphi^B}{2} \tag{4.8}$$

$$\dot{\varphi}^{AB}_k = \frac{\varphi^{AB}_k - \varphi^{AB}_{k-1}}{\Delta t_{k-1,k}} \tag{4.9}$$

$$\dot{\varphi}^{CD}_k = -\dot{\varphi}^{AB}_k \tag{4.10}$$

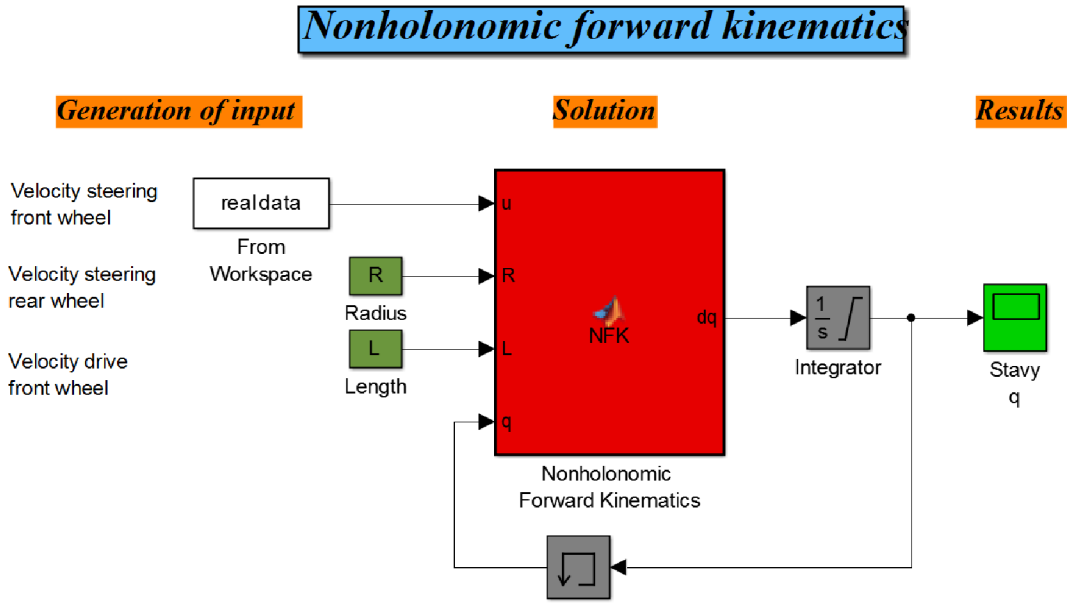
$$\Delta\vartheta^{AB}_{k-1,k} = \frac{\Delta n^A_{k-1,k} + \Delta n^B_{k-1,k}}{2} \cdot \frac{1}{1064} \tag{4.11}$$

$$\dot{\vartheta}^{AB}_k = \frac{\Delta\vartheta^{AB}_{k-1,k}}{\Delta t_{k-1,k}} \tag{4.12}$$

kde

φ^A, φ^B	úhel natočení levého (A) resp. pravého (B) kola přední nápravy vozidla
φ^{AB}	úhel natočení "středního" kola přední nápravy
$\dot{\varphi}_k^{AB}, \dot{\varphi}_k^{CD}$	úhlová rychlost natáčení "středního" kola přední nápravy (AB) resp. zadní nápravy (CD) v aktuálním kroku
$\Delta\vartheta_{k-1,k}^{AB}$	pootočení "středního" kola přední nápravy mezi minulým a aktuálním krokem
$\dot{\vartheta}_k^{AB}$	rychlost otáčení "středního" kola přední nápravy v aktuálním kroku
$\Delta n_{k-1,k}^A$	počet tiků ekodéru mezi minulým a aktuálním krokem (levý motor - A)
$\Delta n_{k-1,k}^B$	počet tiků ekodéru mezi minulým a aktuálním krokem (pravý motor - B)
$\Delta t_{k-1,k}$	doba mezi minulým a aktuálním krokem

Výstupem z modelu jsou pak již přímo souřadnice vozidla *x*, *y* a jeho orientace θ pro každý krok.



Obr. 4.7: Neholonomní dopředný kinematický model vozidla Car4 v Simulinku [29]

Vizuální odometrie - zpracování obrazu z kamery

V případě vizuální odometrie bylo využito programu LIBVISO2 popsaného v kapitole 2.2.2.

Vstupem do programu (verze pro jednu kameru - monokulární vizuální odometrie) byly obrazy z kamery snímače Xtion převedené do černobílé podoby, dále optické vlastnosti kamery (např. ohnisková vzdálenost) a také výška kamery nad zemí.

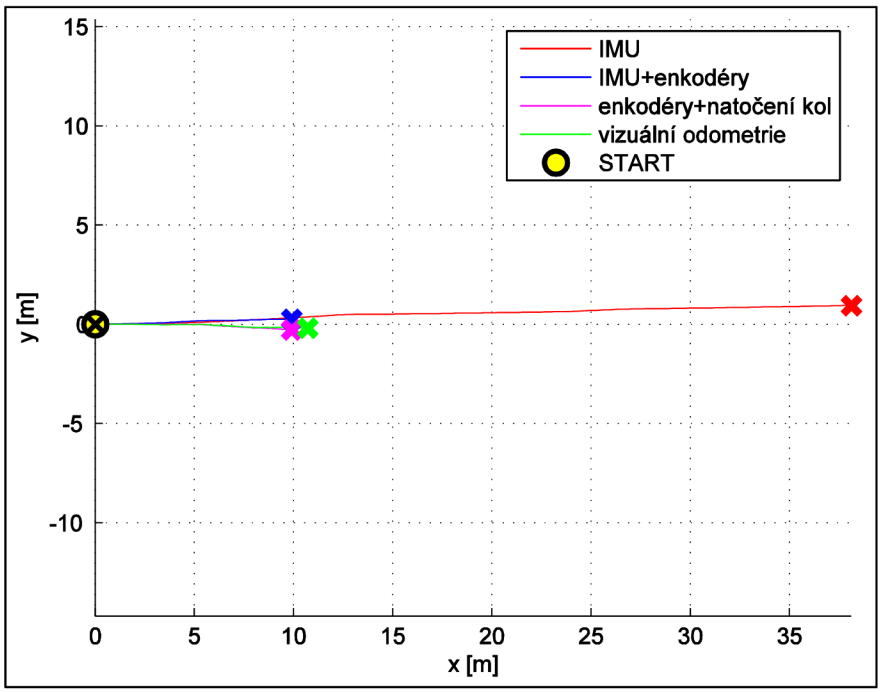
Výstupem pak programu jsou pak již přímo souřadnice vozidla *x, y* a jeho orientace θ pro každý krok.

Srovnání všech metod měření odometrie v praxi

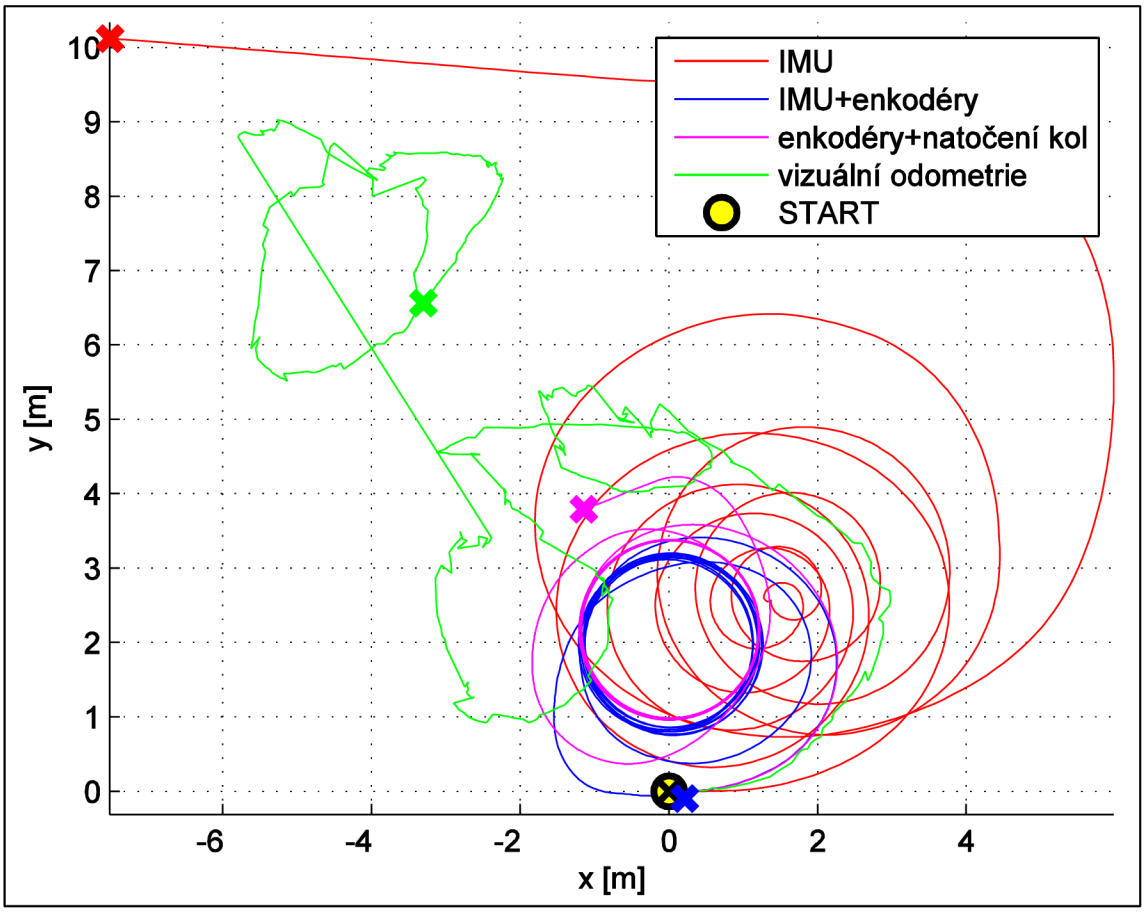
Při srovnávání použitelnosti jednotlivých výše uvedených metod jsme provedli sérii testů, které prověřily jak přesnost měření jednotlivých veličin (poloha *x*, *y* a orientace θ), tak celkové výsledky na tratích s různou délkou a komplikovaností. Testy můžeme rozdělit do tří obecných kategorií:

1) Jízda přímo rovně na dráze 10 m (testování přesnosti měření polohy *x*, *y*)
2) Jízda po kruhové dráze (10 nebo 5 kol) se stejnou počáteční i konečnou orientací vozidla (testování přesnosti orientace θ)
3) Jízda po různě složitých a dlouhých drahách se stejnou počáteční i konečnou orientací i polohou vozidla (testování všech tří veličin)

Výsledky těchto testů jsou uvedeny níže, křížkem jsou vždy označeny odhadované finální pozice podle jednotlivých metod.



Obr. 4.8: Testy odometrie - přímá dráha 10 m (chodba)

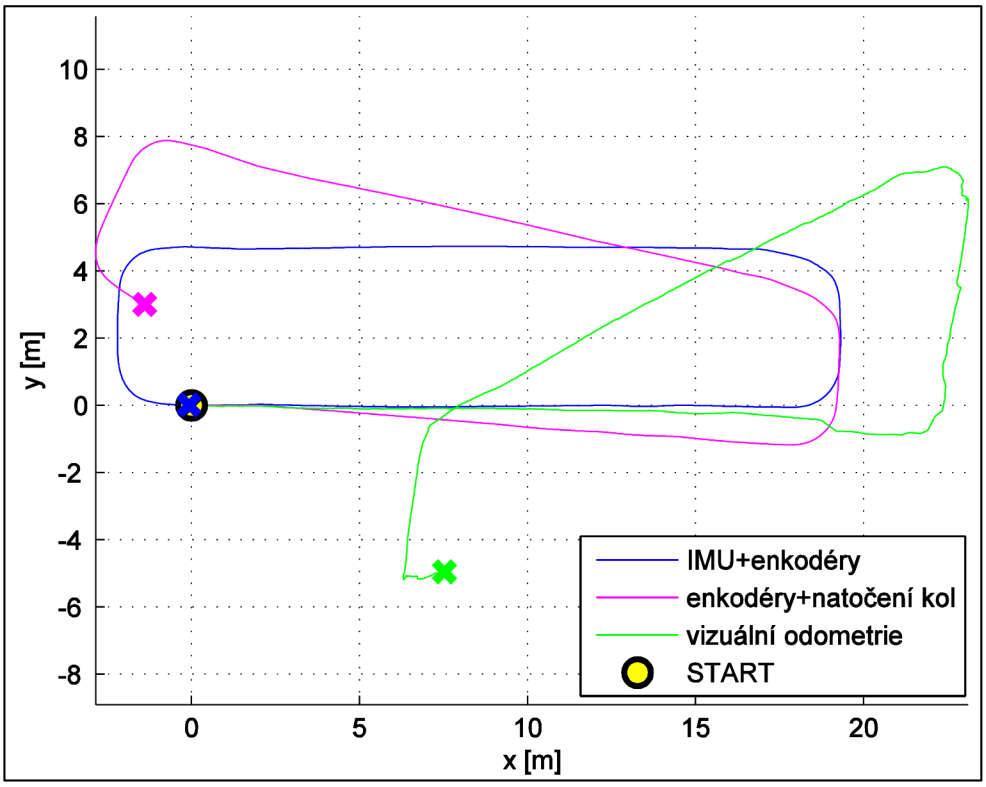


Obr. 4.9: Testy odometrie - 10 kol (chodba)

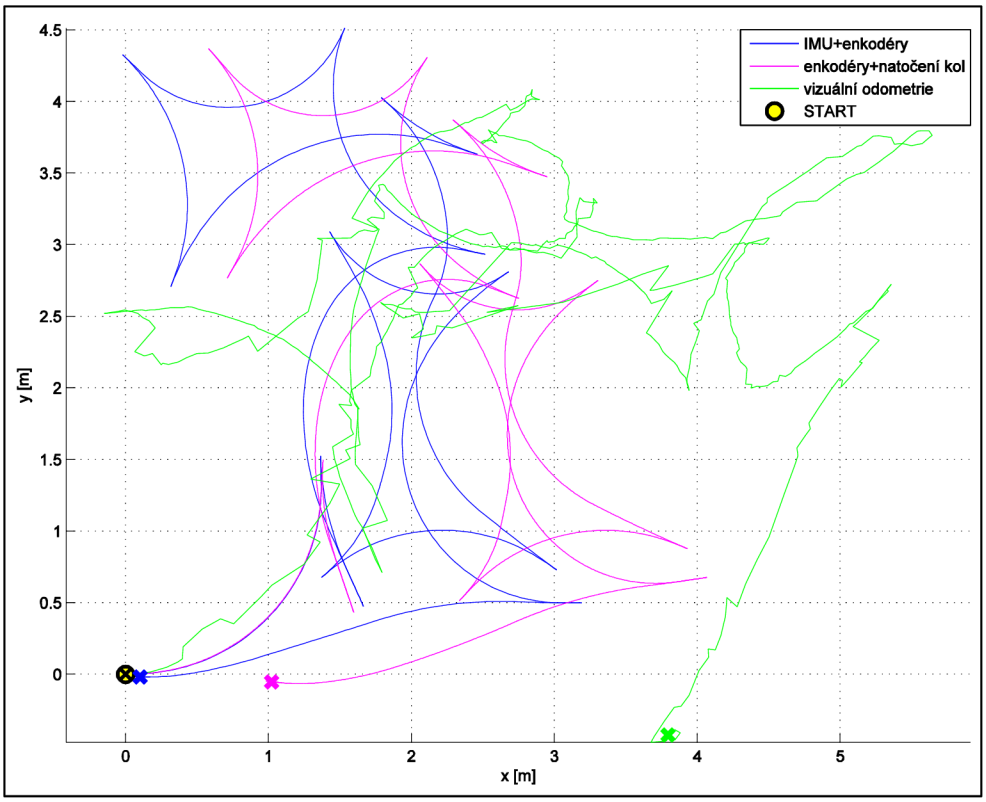
Jak vidíme již z prvních dvou testů na Obr. 4.8 a Obr. 4.9, přesnost při použití samotné IMU je naprosto nevyhovující, za což pravděpodobně může velká citlivost výpočtu ujeté dráhy na nepřesnosti měření akcelerometru (viz. dvojný integrál v rovnici 2.9). Z tohoto důvodu již nebudeme výsledky získané použitím samotné IMU v dalších grafech zobrazovat.

Také výsledky vizuální odometrie nejsou zejména u druhého testu příliš přesné, což ovšem může být dáno pouze nedostatečnou "pestrostí" snímané scény (velké bílé plochy stěn chodby).

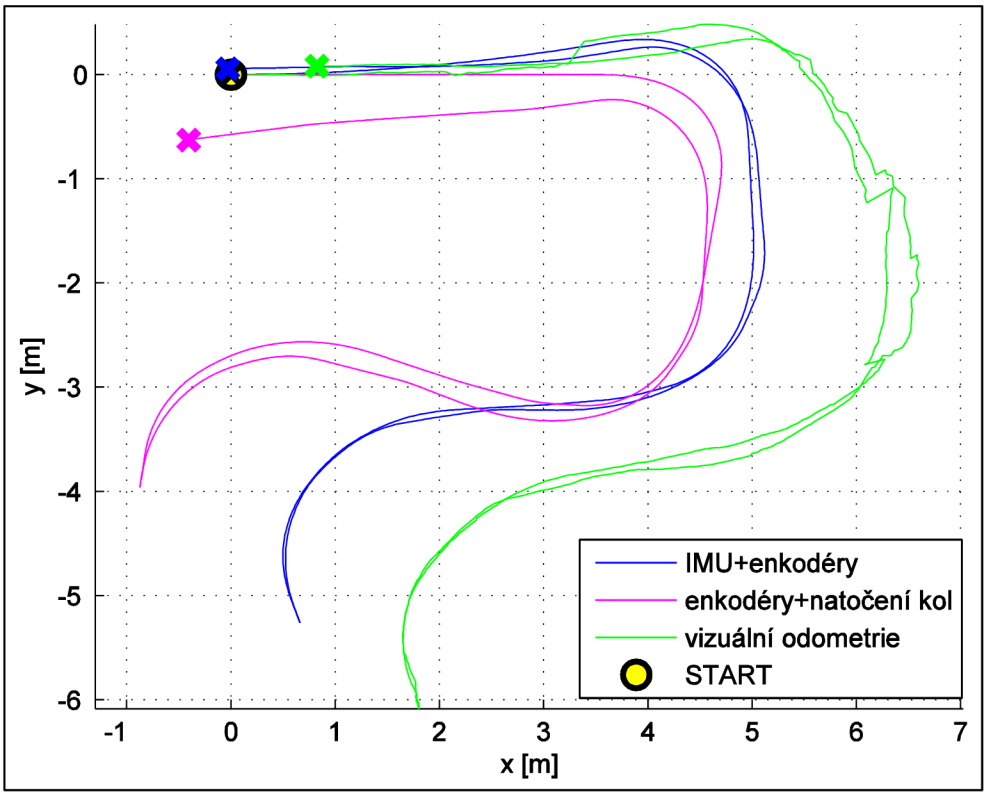
Další uvedené testy již probíhaly po dráhách se stejnou počáteční i konečnou orientací i polohou vozidla.



Obr. 4.10: Testy odometrie - obdélníková dráha (chodba, návrat na start)



Obr. 4.11: Testy odometrie - chaotická dráha (chodba, návrat na start)



Obr. 4.12: Testy odometrie - "tam a zpátky" (laboratoř, návrat na start)

Na Obr. 4.10 a Obr. 4.11, se jen potvrzuje, že přesnost vizuální odometrie je na chodbách velmi špatná. V posledním testu (Obr. 4.12), který se odehrával v laboratoři a kde již byla snímaná scéna pestřejší, si tato metoda vedla mnohem lépe. Obecně špatné výsledky jsou pravděpodobně způsobeny zejména ne příliš vysokou kvalitou obrazových dat, příliš rychlým pohybem obrazu při zatáčení (rozmazání, velký posun mezi dvěma snímky) a také citlivostí metody na přesné nastavení vstupních parametrů.

Co se týče srovnání posledních dvou zbývajících metod, je jasně vidět, že nejlepším řešením je bezpochyby použití kombinace IMU a enkodérů. Při aplikaci SLAM algoritmů na vozidlo Car4 jsme tedy použili právě tuto metodu měření odometrie.

Určení nepřesnosti snímače

Poté co byla určena kombinace snímačů, jsme provedli rychlou analýzu dat, abychom získali odhad rozptylu měření odometrie. Jak je vidět z grafů uvedených výše, přesnost vybrané metody měření je skutečně vysoká a hodnoty rozptylu vycházely v řádu 10^{-6} což je velmi malá hodnota.

Během pozdějších testů se ovšem ukázalo, že toto platí pouze za ideálních podmínek. Může se stát, že dojde chybě, která poměrně hodně ovlivní výstupní data

odometrie. Takovou chybou může být proklouznutí kol či smyk ve vyšší rychlosti, nebo chyba měření IMU při nárazu ochranného rámu do zdi, případně porucha přenosu dat.

Tyto události pak mají velký vliv na celý EKF SLAM algoritmus, protože při nastavení malé hodnoty rozptylu systém stále velmi věří měření odometrie a diverguje od správného řešení.

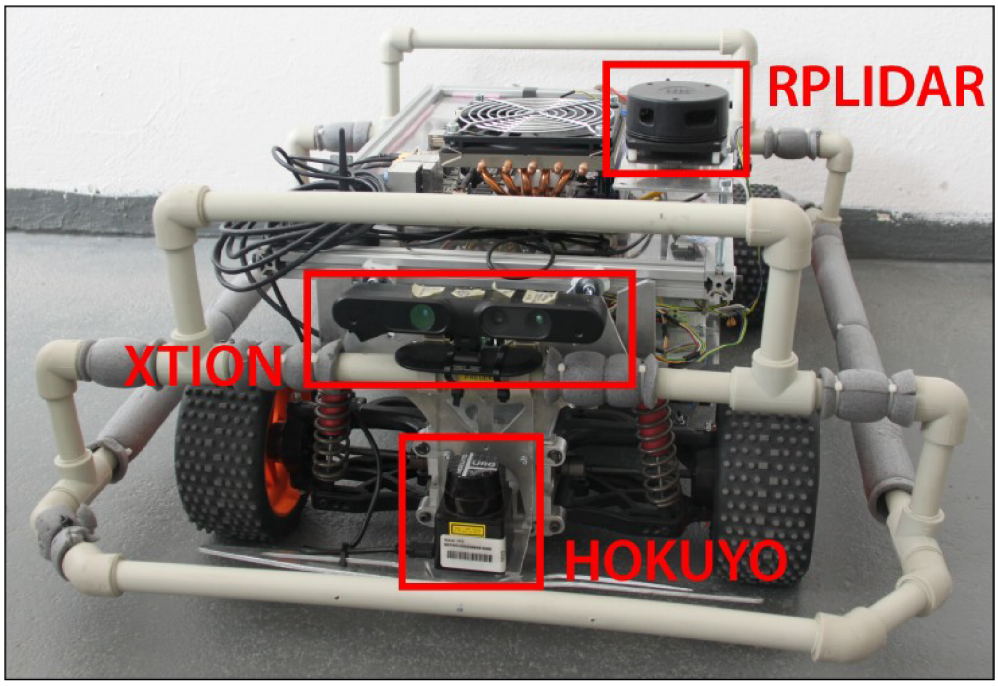
Zejména z tohoto důvodu - eliminace vlivu těchto náhodných chyb, jsme se rozhodli pro nastavení hodnoty rozptylu měření odometrie v řádu 10^{-2}, což ve většině případů silně podceňuje skutečnou přesnost, ovšem algoritmus je poté mnohem odolnější vůči výše zmíněným vlivům.

Hodnota q pro kovarianční matici $\boldsymbol{Q}$ (kapitola 2.1.4, Obr. 2.9) je tedy předběžně stanovena na 0,01 s možností úprav podle výsledků dalších praktických testů.

4.2.2 Snímání okolí - přehled a srovnání snímačů

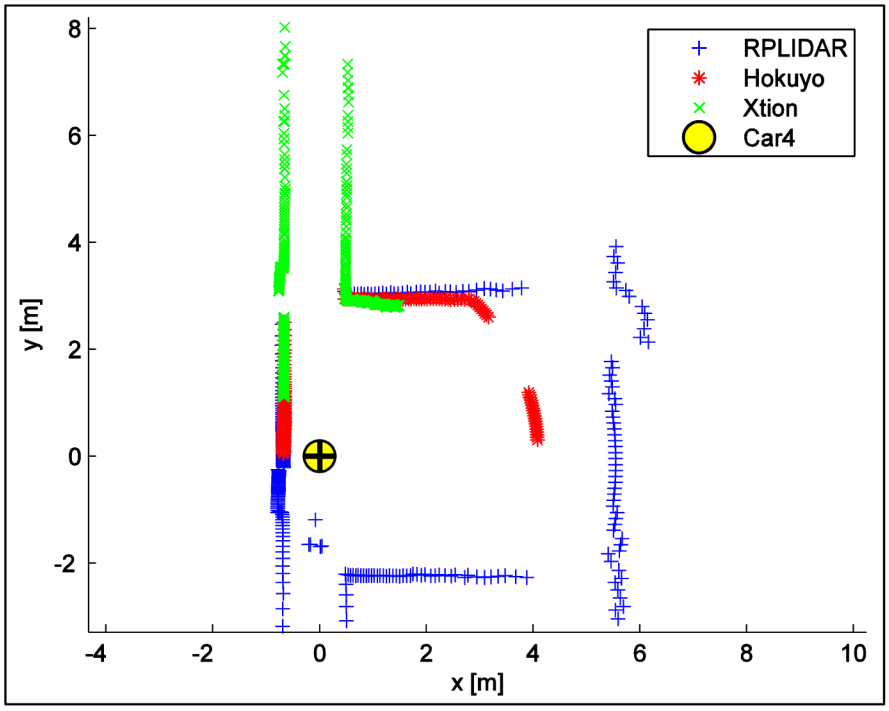
U senzorů pro snímání okolí Asus Xtion a Hokuyo jsme na základě předchozí práce [2] již znali jejich charakteristiky jako směrodatná odchylka měření nebo chyba měření. V případě RPLIDARu jsme tyto charakteristky neznali, nicméně před jejich měřením jsme nejprve provedli obecné srovnání všech senzorů, abychom zjistili, zda tento snímač bude vůbec použitelný pro další práci.

Pro toto srovnání byly všechny tři senzory připevněny na vozidlo Car4 tak, aby měly k dispozici co nejširší zorné pole a navzájem si nepřekážely. Umístění senzorů je patrné z Obr. 4.13. V případě senzoru Hokuyo bylo díky jeho umístění omezeno zorné pole z 240° na cca 180°.

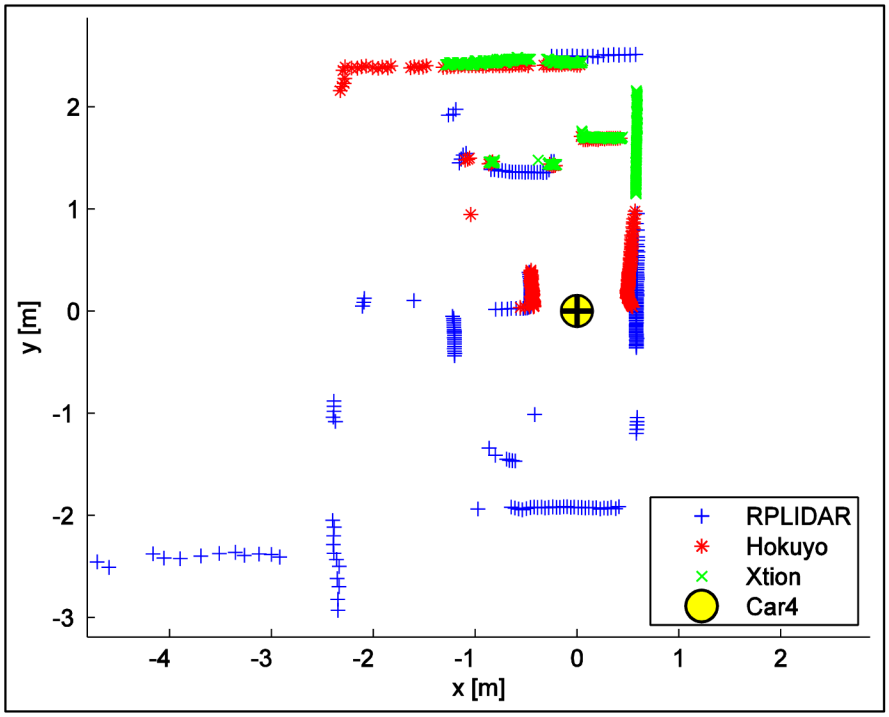


Obr. 4.13: Umístění senzorů Xtion, RPLIDAR a Hokuyo na vozidle Car4

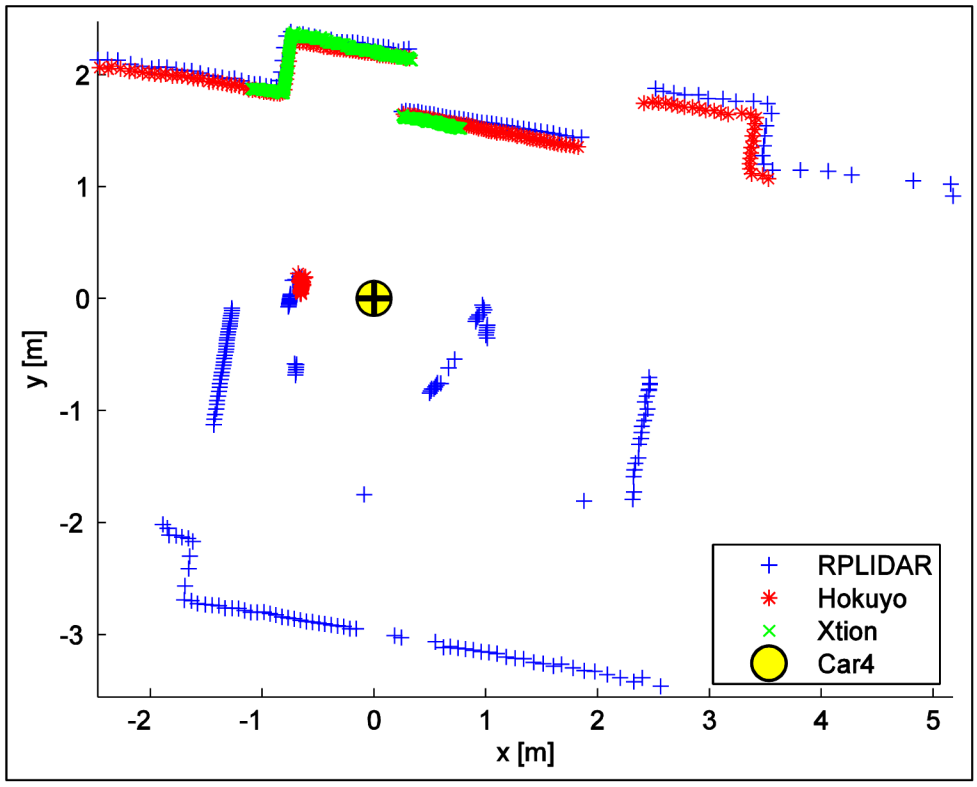
Poté byla provedena série měření v různých typech indoor prostředí jako například dlouhá chodba, prázdná místnost, učebna se stoly atd. V rámci srovnání jsme zkoumali vzájemné odlišnosti výstupních dat senzorů při detekci identických překážek. Grafy vybraných situací jsou na Obr. 4.14, Obr. 4.15 a Obr. 4.16.



Obr. 4.14: Srovnání senzorů Xtion, RPLIDAR a Hokuyo - menší prostranství a chodby



Obr. 4.15: Srovnání senzorů Xtion, RPLIDAR a Hokuyo - učebna



Obr. 4.16: Srovnání senzorů Xtion, RPLIDAR a Hokuyo - prostor schodiště a navazující chodba

Jak je vidět na výše uvedených výsledcích, nejmarkantnější rozdíl je v zorném poli jednotlivých snímačů. V tomto ohledu jasně vítězí RPLIDAR i při uvážení omezeného výhledu u Hokuya.

Dalším výrazným rozdílem je vzdálenost, na kterou jsou senzory schopné detekovat překážku. To je patrné zejména na Obr. 4.14, kde má ve výhledu do dlouhé chodby nejlepší výsledky Xtion. Naopak Hokuyo dává na konci svého dosahu velmi zkreslená měření (zakřivená část v pravé polovině).

Menší odchylky mezi výstupy jednotlivých zařízení jsou způsobeny jednak jejich chybami měření a dále ne zcela dokonalým sesazením všech skenů do jednoho grafu.

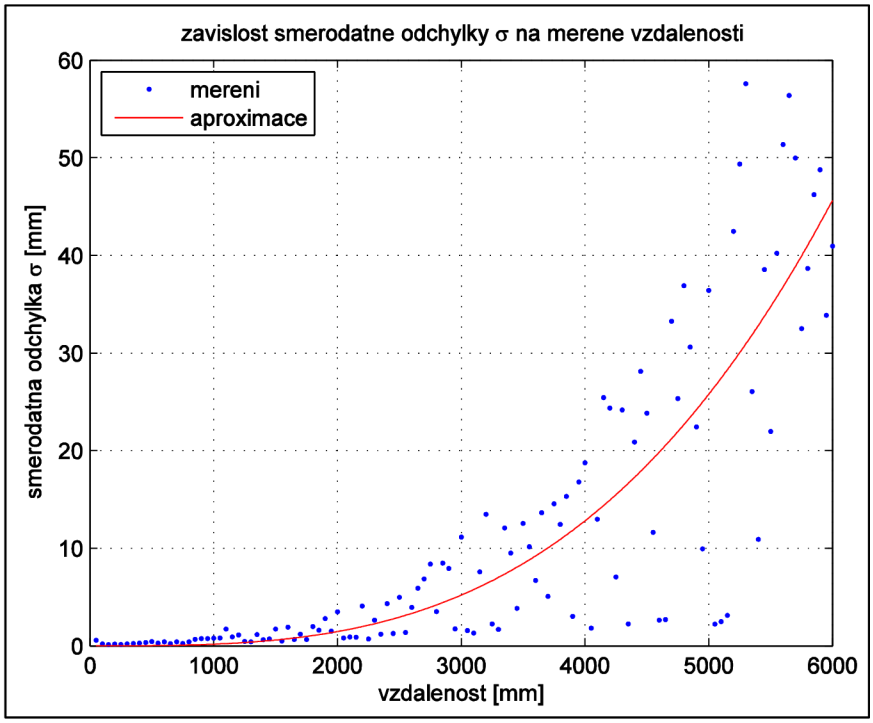
Z tohoto testu vyplývá, že zejména senzor RPLIDAR má největší potenciál pro SLAM právě díky svému zornému poli a relativně velkému dosahu.

Přesnost snímače RPLIDAR (a určení rozptylu měření)

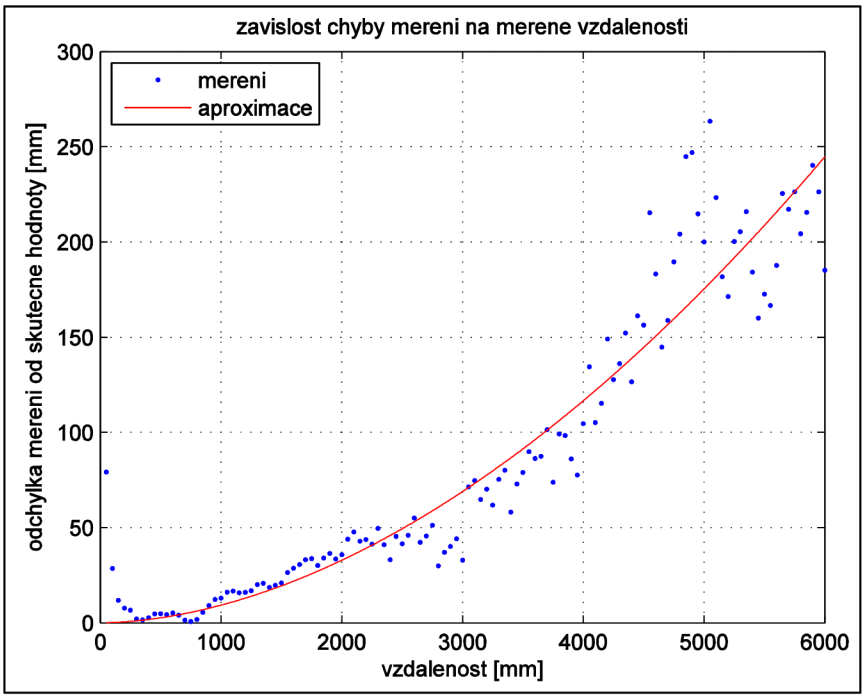
Pokud tedy chceme RPLIDAR dále používat, je nutné provést měření alespoň základních charakteristik přesnost tohoto snímače. V rámci této práce byly změřeny:

- závislost směrodatné odchylky měření na měřené vzdálenosti
- závislost chyby měření na měřené vzdálenosti
- závislost rozlišitelnosti na měřené vzdálenosti

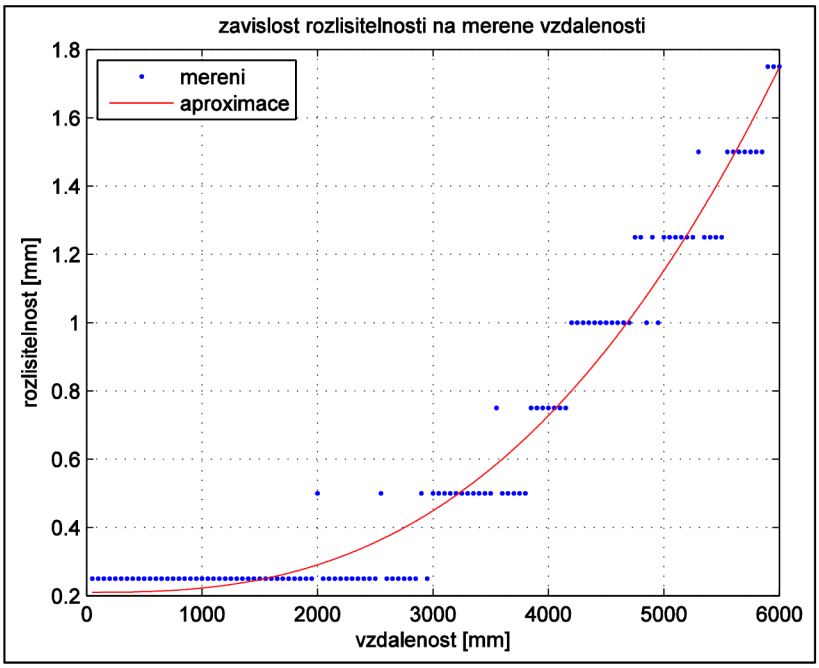
Všechna tato měření byla prováděna v celém rozsahu uváděném výrobcem (tedy až do vzdálenosti 6m) pro jeden paprsek s úhlem ±0°. Povrch, na který paprsek dopadal, byl bílý papír. Měření probíhalo ve 120 sériích (každá po 50 měřeních) s krokem 5 cm. Výsledky měření jsou na Obr. 4.17, Obr. 4.18 a Obr. 4.19.



Obr. 4.17: Směrodatná odchylka RPLIDARu



Obr. 4.18: Chyba měření RPLIDARu



Obr. 4.19: Rozlišitelnost RPLIDARu

Jak je vidět z uvedených grafů, všechny veličiny rostou se zvětšující se vzdáleností a zejména u směrodatné odchylky a chyby mají nezanedbatelnou velikost. V případě rozlišitelnosti jsou hodnoty v porovnání se směrodatnou odchylkou zanedbatelné - jinými slovy, rozlišitelnost je naprosto dostačující. Naměřené hodnoty lze aproximovat mocninnými funkcemi.

Směrodatná odchylka:

$$\sigma = (6{,}4547 \cdot 10^{-11}) \cdot r^{3{,}1363}\ [mm] \tag{4.13}$$

Chyba měření:

$$e = (3{,}0175 \cdot 10^{-5}) \cdot r^{1{,}8286}\ [mm] \tag{4.14}$$

Rozlišitelnost:

$$r_a = (1{,}0887 \cdot 10^{-10}) \cdot r^{2{,}6865} + 0{,}2098\ [mm] \tag{4.15}$$

kde *r* je velikost měřené vzdálenosti v jednotkách [mm].

Tyto výsledky odpovídají přepokládaným průběhům jak byly zmíněny v kapitole 2.3.2, resp. v [27].

Naměřená data jsou dále použita v rámci SLAM algoritmů pro kompenzaci chyby měření a také pro definování míry přesnosti (resp. šumu) snímače, vyjádřené v matici ***R*** (viz. kapitola 2.1.4, Obr. 2.10). Pro zjednodušení užíváme v celém výpočtu nejhorší možnou variantu - tedy nevyšší hodnotu směrodatné odchylky 58 mm.

5 Aplikace SLAM algoritmů s vybranými senzory

5.1 CAS-toolbox – úpravy pro potřeby Car4

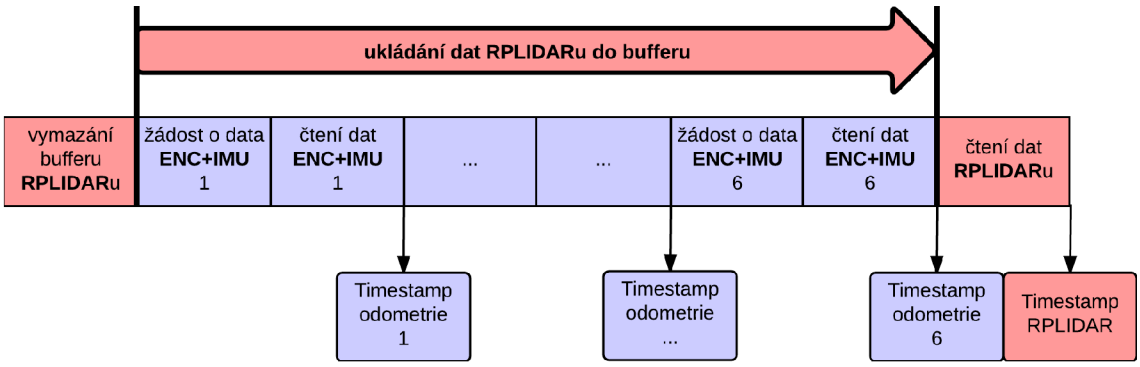
Díky flexibilní struktuře celého toolboxu, je pro úspěšnou aplikaci na vozidlo Car4 nutné provést jen velmi malé množství změn.

Hierarchie senzorů a časování

První je třeba vyřešit hierarchii senzorů a časování měření. Jak již bylo zmíněno v kapitole 2.2.3, je nutné jeden ze vstupních senzorů označit jako hlavní a v závislosti na něm se budou vykonávat jednotlivé kroky algoritmu. V naší aplikaci jsme za hlavní snímač označili laserový skener RPLIDAR, mimo jiné proto, že je výrazně pomalejší než měření dat z enkodérů + IMU.

Dále je nutné zajistit, aby data ze všech snímačů měla synchronizované časové značky (timestamp). Zde bylo využito interních funkcí MATLABu "tic" a "toc". První z nich odstartuje velmi přesný interní časovač a druhá vždy vrátí jeho aktuální hodnotu.

Vzhledem k nestejné rychlosti měření vstupních dat, bylo mezi dvěma měřeními RPLIDARu provedeno několik měření odometrie (enkodéry+IMU). Experimentálně bylo zjištěno, že může být takto provedeno až 6 měření odometrie, aniž by došlo ke zpomalení celého procesu. Názorný diagram měření pro jeden krok algoritmu je na Obr. 5.1.



Obr. 5.1: Schéma měření vstupních dat pro SLAM

Model predikce polohy na základě odometrie

CAS-toobox má v sobě zabudován pouze model predikce pro dvoukolového diferenciálně řízeného robota. Bylo tedy nutné vytvořit model nový, který by pracoval s námi měřenými daty a odpovídal chování vozidla Car4.

Tento model byl vytvořen na základě rovnic uvedených v kapitole 4.2.1 v sekci "Kombinace údajů z IMU a enkodérů". Dále bylo využito algoritmů pro predikci pozice robota EKF popsaných v kapitole 2.1.4.

Vstupem do modelu predikce jsou stavy vozidla a jeho kovarianční matice z předchozího kroku a dále naměřená data odometrie pro aktuální krok (série 6 měření). Výstupem jsou pak stavy a kovarianční matice vozidla ve finálním pozici, Jakobián modelu predikce a také všechny mezistavy (pro 1. - 6. měření odometrie).

Okomentovaný zdrojový kód je uveden v příloze na Obr. 9.1.

Úprava konfiguračního souboru

Nakonec je také třeba upravit soubor obsahující nastavení celého experimentu, který zapisuje údaje do proměnné "params" používané v celém algoritmu.

Hlavními změnami je zde nastavení přesnosti námi používaných senzorů na základě naměřených hodnot (viz. kapitola 4.2), vymazání nepotřebných parametrů a nastavení polohy laserového skeneru vůči středu vozidla. Dále jsou zde uvedeny parametry metody extrakce orientačních bodů, jejichž ideální hodnoty byly laděny v průběhu testování celého algoritmu SLAM.

5.2 Výběr metody extrakce orientačních bodů

Pro další práci je také nutné zvolit, jakou metodu extrakce orientačních bodů budeme používat. Nicméně vhledem k tomu, že přímo v toolboxu je již zabudovaná funkce pro extrakci přímek z dat laserového skeneru a s ohledem na prostředí, ve kterém bude testování probíhat, je volba právě této metody jasná.

Z uživatelského hlediska se může zdát tato reprezentace trochu nepřehledná (viz. například Obr. 2.18), zejména když je přímek v mapě více. Pro robota je ovšem ideální.

Také je třeba připomenout, že mapou se zde rozumí soubor orientačních bodů. Tato mapa slouží výhradně k lokalizaci robota a v podobě v jaké je implementována v CAS-toolboxu ji nelze přímo využít k plánování trajektorie pro autonomní pohyb. Problém plánování je další samostatnou kapitolou oblasti autonomní robotiky a je již nad rámec této práce.

5.3 Nastavení a ladění parametrů algoritmů

Při praktickém testování CAS-toolboxu bylo zjištěno, že je třeba věnovat velkou pozornost nastavení parametrů simulace prováděném ve výše zmíněném konfiguračním souboru, který toto nastavení zapisuje do proměnné "params".

Velký vliv mají zejména parametry funkce pro hledání orientačních bodů - v našem případě extrakce přímek z dat laserového skeneru. V tomto případě i malá změna může mít velký vliv na celkové chování, což se projeví dvěma způsoby:

1) Přesnost korekční fáze EKF ve SLAM algoritmu
2) Rychlost provádění celého algoritmu

Na přesnost mají vliv parametry týkající se maximální odchylky bodů od proložené přímky, tolerance pro fůzi více podobných přímek do jedné, nebo velikosti okna, ve kterém probíhá prokládání bodů přímkou.

Na rychlost provádění má vliv zejména parametr určující minimální délku segmentu, který lze považovat za orientační bod. To proto, že čím kratší segmenty povolíme, tím více orientačních bodů nalezneme. Čím více však máme orientačních bodů, tím větší jsou rozměry stavového vektoru $\boldsymbol{x}$, kovarianční matice $\boldsymbol{P}$ i dalších matic používaných v EKF. Celý algoritmus se tak zpomaluje, jelikož operace s velkými maticemi zabírají více výpočetního času.

Orientační body založené na malých segmentech nemusí být navíc velmi spolehlivé - čím kratší segment, tím menší spolehlivost.

Ladění všech těchto parametrů probíhalo metodou pokus-omyl na základě testů prováděných v reálném indoor prostředí. Pravděpodobně není možné nalézt kombinaci nastavení, která by za všech okolností dávala nejlepší výsledky, nicméně jsme se po sérii testů dopracovali k jakémusi "hrubému optimu", tedy k parametrům poskytujícím uspokojivé výsledky pro většinu situací a jenž slouží jako výchozí bod pro další optimalizaci:

- `windowsize = 21` (velikost okna - počet bodů, jimiž se prokládá přímka)
- `threshfidel = 0.02` ("věrnost přímce" - míra maximální odchylky bodů od prokládané přímky)
- `fusealpha = 0.89999` (míra podobnosti nutná k fůzi více přímek do jednoho orientačního bodu)
- `minlength = 1.2` (minimální délka segmentu (v metrech), který lze považovat za orientační bod)

Je zde také nutné připomenout, že tyto parametry platí pro tuto konkrétní aplikaci, tedy při použití vozidla Car4 a výše uvedených senzorů.

5.4 Implementace pro řešení v reálném čase

Veškeré předchozí testy SLAM algoritmů probíhaly tak, že nejprve byla naměřena data z jednotlivých jízd, která byla až dodatečně zpracována v CAS-toolboxu. To umožnilo rychlé a efektivní odladění chyb v kódu a také testování vlivu různých parametrů simulace na její výsledek při použití identických vstupů.

Aby mohl být splněn jeden z cílů práce, implementace pro řešení v reálném čase, bylo nutné přepsat část kódu hlavní funkce "slam.m". Šlo o odstranění sekce, jenž čte vstupní data z uložených souborů a její nahrazení vlastním kódem, který v každém kroku algoritmu provede 6 měření odometrie a jedno měření laserového skeneru (viz. Obr. 5.1). K tomu samozřejmě bylo nutné připojit soubor funkcí obstarávající komunikaci po sériové lince, dekódování zpráv ze snímačů a zpracování dat do podoby použitelné v dalších funkcích toolboxu.

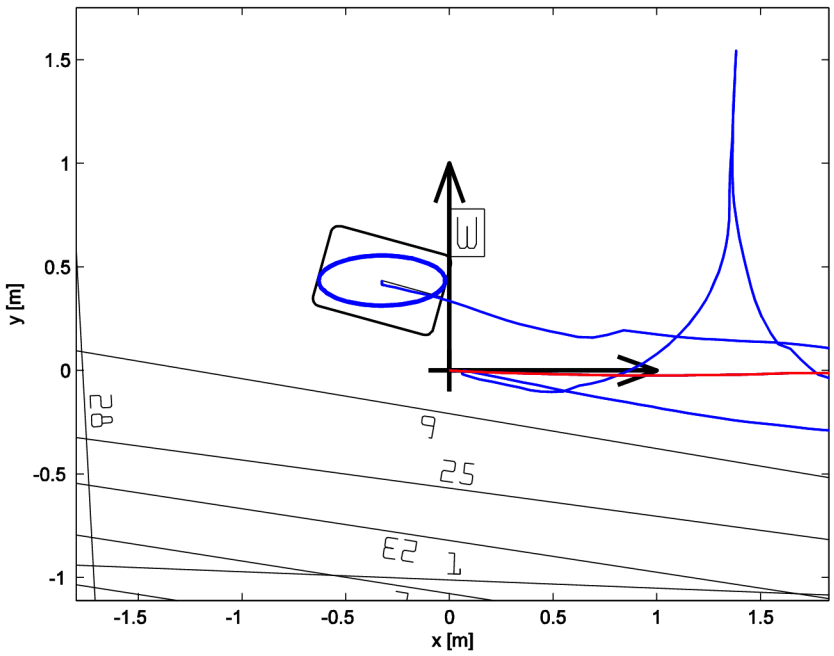
6 Experimenty v reálném prostředí

Finální experimenty probíhaly na chodbách Fakulty strojního inženýrství a byla při nich použita verze programu pro řešení v reálném čase. Ta je sice oproti samotnému snímání a až následnému vyhodnocování pomalejší, i přesto však při rychlosti cca 3,5 Hz dávala naprosto dostačující výsledky.

Průměrná doba každé testovací jízdy byla 10-15 minut, přičemž za tuto dobu vozidlo najelo odhadem několik stovek metrů. Vozidlo vždy zakončilo jízdu ve stejném bodě, z jakého začínalo.

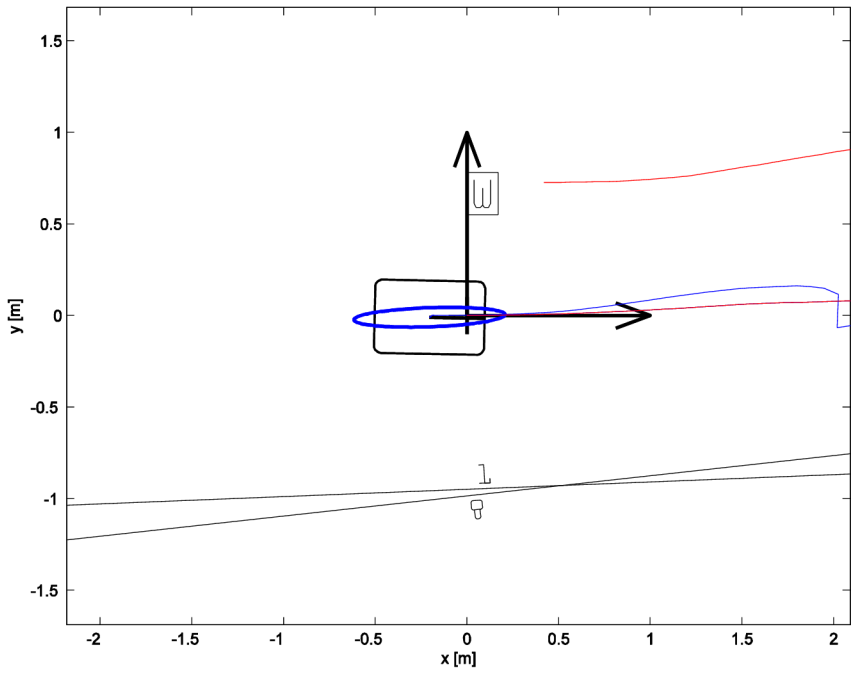
V této kapitole uvedeme dva příklady experimentů. Oba trvaly 15 minut a vozidlo při nich ujelo přes 300 m. Díky správnému nastavení parametrů extrakce přímek, nebylo na mapě více než 30 unikátních orientačních bodů, takže algoritmus běžel plynule a výsledná mapa je stále relativně přehledná.

První je jízda z počáteční fáze testování kompletního SLAM algoritmu. Vzhledem k tomu že nebyly ještě zcela vyladěny parametry simulace ani prevence chyb, došlo k relativně výraznému odchýlení měření odometrie od skutečnosti. EKF ovšem zafungoval dobře a ve finále byla odchylka startovní a finální polohy zhruba 53 cm. To že skutečná poloha neleží ve znázorněné elipse nejistoty polohy vozidla je opět důsledek ne zcela optimálního nastavení parametrů. Zvětšená část zobrazující startovní/cílovou oblast je na Obr. 6.1. Celá mapa s vyznačenou dráhou generovanou samotnou odometrií a dráhou ze SLAMu je v příloze na Obr. 9.2.



Obr. 6.1: Detail mapy v místě startovní/cílové zóny pro ukázkový Test č.1

Druhá jízda je z jednoho z posledních provedených testů. Došlo k menší změně parametrů simulace a také k přidání několika mechanismů pro prevenci náhodných chyb. Výsledkem jsou velmi přesná měření odometrie, takže paradoxně zde není vliv EKF až tak viditelný. Ve výsledku skončilo vozidlo 20 cm od startovní pozice, která tentokrát již leží v elipse nejistoty polohy. Tento výsledek již můžeme považovat za velmi uspokojivý. Detail cílové oblasti je na Obr. 6.2, celá mapa opět v příloze jako Obr. 9.3.



Obr. 6.2: Detail mapy v místě startovní/cílové zóny pro ukázkový Test č.2

7 Závěr

Cílem této práce bylo zejména zprovoznění a otestování nových senzorů a dále aplikace SLAM algoritmů na experimentální vozidlo Car4.

V rešeršní části je uveden základní popis algoritmů pro SLAM s využitím rozšířeného Kalmanova filtru (EKF) a dále výsledky průzkumu v oblasti dostupných softwarových nástrojů pro MATLAB, které tuto problematiku řeší. Také je zde kapitola zabývající se popisem všech snímačů, použitých dále v této práci.

Výstupem rešerše je obecný přehled problematiky SLAM z hlediska jak použitého hardwaru, tak funkce celého algoritmu. Dále byl zvolen konkrétní softwarový nástroj, CAS-toolbox, s jehož pomocí byly algoritmy SLAM aplikovány na vozidlo Car4, a byl proveden popis jeho fungování a zabudovaných metod.

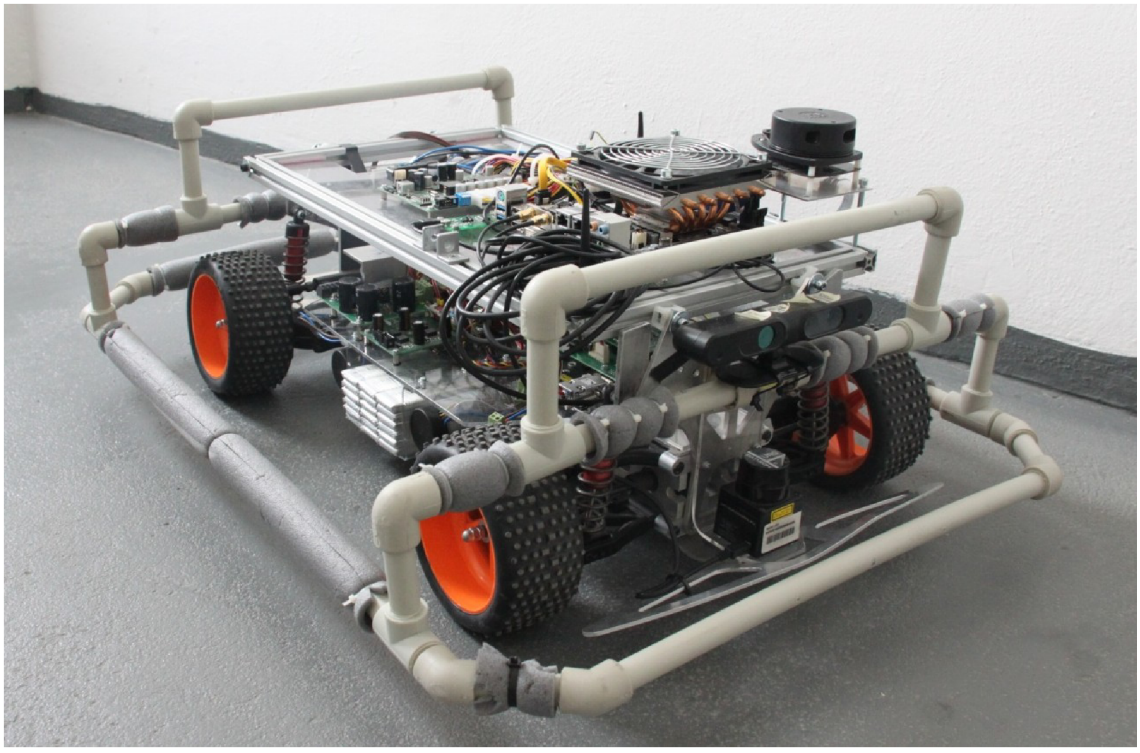
První polovina praktické části se nejprve zabývala jednotlivými senzory a získáváním dat pro další zpracování. Poté byla provedena první velká série testů, které zkoumaly vhodnost těchto snímačů pro měření odometrie a pro snímání okolního prostředí. Výsledkem byl výběr kombinace enkodérů, IMU a laserového skeneru RPLIDAR, jelikož nejlépe vyhovovaly našim požadavkům na přesnost a funkčnost.

Při testování vyvstaly u některých senzorů problémy, díky kterým nebyly vybrány pro další použití. Pokud by ovšem bylo v budoucnu aplikováno vhodné řešení těchto nedostatků, mohly by sloužit stejně dobře, možná i lépe než současná kombinace. Toto však již nebylo cílem této práce a mohlo by být námětem pro její další rozšíření a vylepšení v budoucnu.

Druhá polovina praktické části se zabývala aplikací SLAM algoritmů s vybranými senzory na vozidlo Car4. Byly provedeny nezbytné úpravy některých funkcí toolboxu a na základě druhé velké série testů byly naladěny parametry programu tak, abychom dosáhli co nejlepších výsledků - tedy přesné lokalizace vozidla v neznámém indoor prostředí.

Zjistili jsme, že se systém dokázal velmi dobře vyrovnat i s většími náhodnými chybami odometrie. Po provedení drobných úprav, další úpravě parametrů a omezením těchto chyb jsme dosáhli poměrně vysoké přesnosti lokalizace s odchylkou pouze 10 - 20 cm při ujeté vzdálenosti přes 300 m. Na výsledných mapách z těchto testů pak byl jasně patrný vliv EKF srovnáním odhadu polohy na základě odometrie s výsledky získanými korekcí - hledáním orientačních bodů pomocí laserového skeneru.

Jak již bylo řečeno, v rámci této práce byly objeveny některé problémy a neprozkoumané oblasti, které by mohly být námětem na další rozšíření a vylepšení. Mezi ně patří například plné využití potenciálu IMU pomocí vhodných algoritmů pro fůzi a vzájemnou korekci všech dostupných dat, dále rozšíření v oblasti vizuální odometrie za použití vhodných algoritmů zpracování obrazu a sterovize, nebo aplikace pokročilejšího kinematického/dynamického vozidla. Také v případě zařízení Xtion bychom mohli jistě získat zajímavé výsledky při využití celého jeho potenciálu kombinací obrazu a hloubkové mapy. Pokud bychom chtěli dosáhnout lepšího výkonu co do rychlosti celého algoritmu a zpracování dat, bylo by vhodné implementovat navržené postupy v jiném programovacím jazyce než je MATLAB (například C++).



Obr. 7.1: Vozidlo Car4 ve finálním stadiu této práce

8 Seznam použitých zdrojů

[1] Google Self-Driving Car Project. *Google+* [online]. 2015 [cit. 2015-05-25]. Dostupné také z: https://plus.google.com/u/0/+SelfDrivingCar/

[2] NAJMAN, Jan. *Rozšíření robotu Car4 o palubní počítač a snímače Kinect a Hokuyo*. Brno, 2013. Bakalářská práce.

[3] RIISGAARD, Søren a Morten Rufus BLAS. *SLAM for Dummies: A Tutorial Approach to Simultaneous Localization and Mapping* [online]. 2003, s. 126 [cit. 14.5.2015].

[4] SIEGWART, Roland. *Introduction to autonomous mobile robots*. 1. vyd. Massachusetts: MIT Press, 2004, 321 s. ISBN 02-621-9502-X.

[5] KREJSA, J. a S. VECHET Infrared Beacons based Localization of Mobile Robot. *Electronics and Electrical Engineering*. 2012, **117**(1): -. DOI: 10.5755/j01.eee.117.1.1046. ISSN 2029-5731. Dostupné také z: http://www.eejournal.ktu.lt/index.php/elt/article/view/1046

[6] GRISETTI, Giorgio, Cyrill STACHNISS, Kai ARRAS a Wolfram BURGARD. *Robotics 2 Data Association* [online]. 2009 [cit. 20.5.2015]. Dostupné také z: http://ais.informatik.uni-freiburg.de/teaching/ws09/robotics2/pdfs/rob2-11-dataassociation.pdf

[7] WELCH, Greg a Gary BISHOP. *An Introduction to the Kalman Filter* [online]. 2006, s. 16 [cit. 19.5.2015]. Dostupné také z: https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf

[8] NAMINSKI, Megan R. *An Analysis of Simultaneous Localization and Mapping (SLAM) Algorithms* [online]. 2013 [cit. 20.5.2015]. Dostupné také z: http://digitalcommons.macalester.edu/mathcs_honors/29/

[9] THRUN, Sebastian. *Probabilistic robotics*. Massachusetts: MIT Press, 2006, xx, 647 s. ISBN 02-622-0162-3.

[10] Covariance matrix. *Wikipedia* [online]. 2015 [cit. 2015-05-21]. Dostupné také z: http://en.wikipedia.org/wiki/Covariance_matrix

[11] KOK SENG CHONG, a L. KLEEMAN Accurate odometry and error modelling for a mobile robot. In: *Proceedings of International Conference on Robotics and Automation*. Albuquerque, NM: IEEE, 1997, s. 2783-2788. DOI: 10.1109/ROBOT.1997.606708. ISBN 0-7803-3612-7. Dostupné také z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=606708

[12] BAILEY, Tim. Matlab Utilities: SLAM Simulations. *Homepage of Tim Bailey* [online]. 2004 [cit. 2015-05-22]. Dostupné také z: http://www-personal.acfr.usyd.edu.au/tbailey/software/slam_simulations.htm

[13] HAIQIANG, Zhang. CEKF-SLAM. *OpenSLAM* [online]. 2007 [cit. 2015-05-22]. Dostupné také z: https://www.openslam.org/cekfslam.html

[14] JUNEJA, Jai. Simultaneous Localisation and Mapping (SLAM) in MATLAB. *Jai Juneja Blog* [online]. 2013 [cit. 2015-05-22]. Dostupné také z: http://www.jaijuneja.com/blog/2013/05/simultaneous-localisation-mapping-matlab/

[15] Linear SLAM. *OpenSLAM* [online]. 2013 [cit. 2015-05-23]. Dostupné také z: https://www.openslam.org/linearslam.html

[16] LIANG ZHAO, , SHOUDONG HUANG a Gamini DISSANAYAKE. Linear SLAM: A linear solution to the feature-based and pose graph SLAM based on submap joining. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tokyo: IEEE, 2013, s. 24-30. DOI: 10.1109/IROS.2013.6696327. ISBN 978-1-4673-6358-7. Dostupné také z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6696327

[17] LIBVISO2: C++ Library for Visual Odometry 2. *Andreas Geiger* [online]. 2011 [cit. 2015-05-23]. Dostupné také z: http://www.cvlibs.net/software/libviso/

[18] ARRAS, Kai O. The CAS Robot Navigation Toolbox. *KTH | Centre for Autonomous Systems - CAS* [online]. 2004 [cit. 2015-05-24]. Dostupné také z: http://www.cas.kth.se/toolbox/

[19] ARRAS, Kai O. *The CAS Robot Navigation Toolbox: Users Guide and Reference*. CAS-KTH (Stockholm), 2004. Dostupné také z: http://www.cas.kth.se/toolbox/UsersGuide-0.9.pdf

[20] *Transmotec - PD42 series DC motors* [online]. b.r. [cit. 14.5.2015]. Dostupné také z: http://www.transmotec.com/dc-motors/planetary-gear/PD42-Series.aspx

[21] *MPU-9150 Product Specification* [online]. 2012 [cit. 14.5.2015]. Dostupné také z: http://www.invensense.com/wp-content/uploads/2015/02/MPU-9150-Datasheet.pdf

[22] SparkFun 9 Degrees of Freedom Breakout - MPU-9150. *Sparkfun* [online]. 2012 [cit. 2015-05-14]. Dostupné také z: https://www.sparkfun.com/products/11486

[23] ASUS - Xtion PRO LIVE. *ASUS* [online]. 2011 [cit. 2013-04-24]. Dostupné také z: http://www.asus.com/Multimedia/Xtion_PRO_LIVE

[24] Distance Measurements with Lasers. *RP Photonics* [online]. 2013 [cit. 2015-05-18]. Dostupné také z: http://www.rp-photonics.com/distance_measurements_with_lasers.html

[25] *Leddar M16 Specification sheet* [online]. 2014 [cit. 18.5.2015]. Dostupné také z: http://leddartech.com/files/documents/3m/66/m16.pdf

[26] *RPLIDAR Low Cost 360 degree 2D Laser Scanner (LIDAR) System: Introduction and Datasheet* [online]. 2014 [cit. 18.5.2015]. Dostupné také z: http://rplidar.robopeak.com/download.html

[27] KONOLIGE, Kurt, Joseph AUGENBRAUN, Nick DONALDSON, Charles FIEBIG a Pankaj SHAH. A low-cost laser distance sensor. In: *2008 IEEE International Conference on Robotics and Automation*. Pasadena, CA: IEEE, 2008, s. 3002-3008. DOI: 10.1109/ROBOT.2008.4543666. ISBN 978-1-4244-1646-2. Dostupné také z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4543666

[28] RPLIDAR - 360 degree Laser Scanner Development Kit. *Seeed* [online]. 2014 [cit. 2015-05-18]. Dostupné také z: http://www.seeedstudio.com/depot/RPLIDAR-360-degree-Laser-Scanner-Development-Kit-p-1823.html

[29] JASANSKÝ, Michal. *Návrh dynamických modelů pro řízení trakce experimentálního vozidla*. Brno, 2010. diplomová práce.

Přehled použitých zkratek:

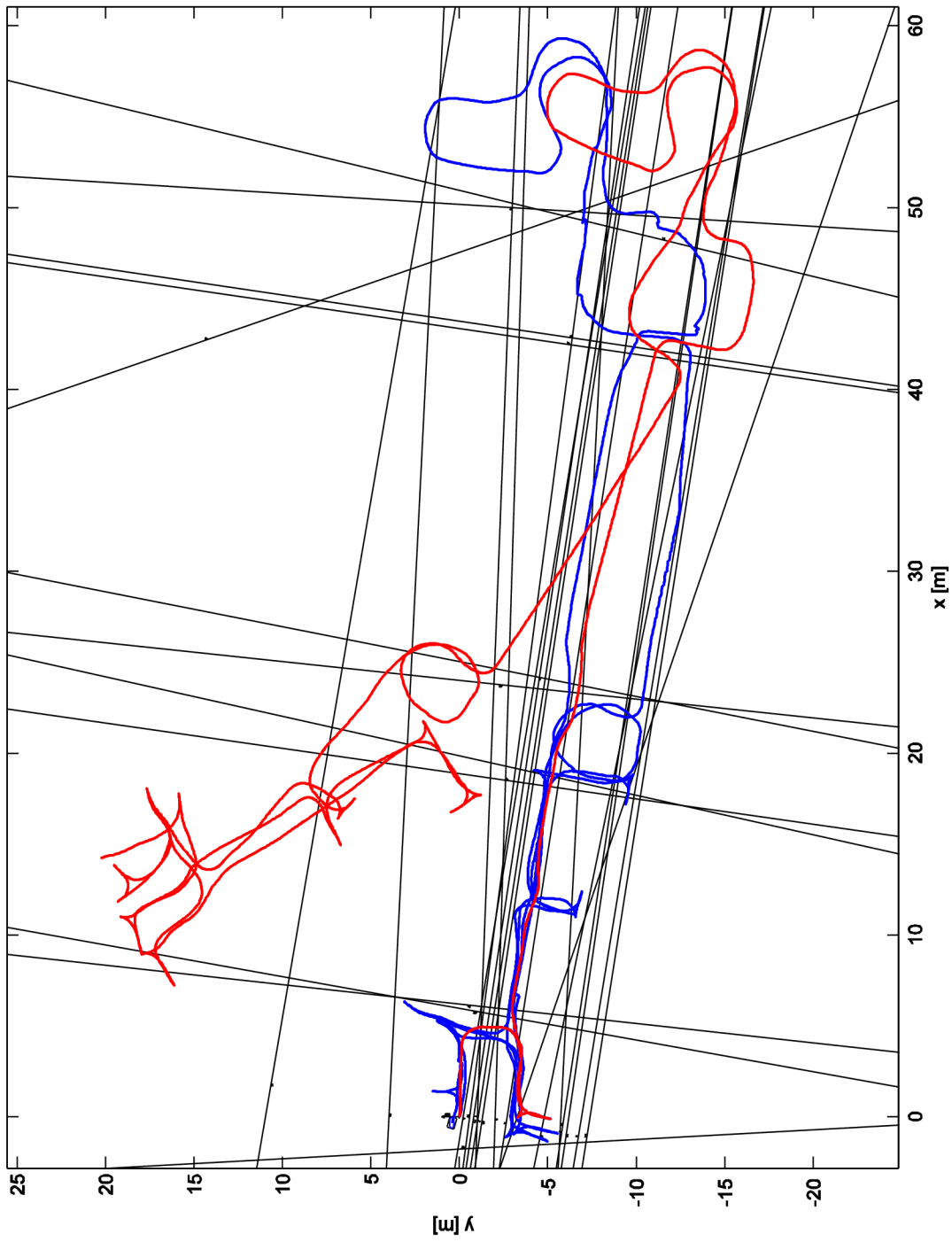
SLAM	Simultaneous Localization And Mapping	simultánní lokalizace a mapování
IMU	Inercial Measurment Unit	inerciální měřicí jednotka
USB	Universal Serial Bus	univerzální sériová sběrnice
UART	Universal Asynchronous Receiver/Transmitter	asynchronní sériové rozhraní
EKF	Extended Kalman Filter	rozšířený Kalmanův filtr
GUI	Graphical User Interface	grafické uživatelské rozhraní
CEKF	Compressed Extended Kalman Filter	komprimovaný rozšířený Kalmanův filtr

9 Přílohy

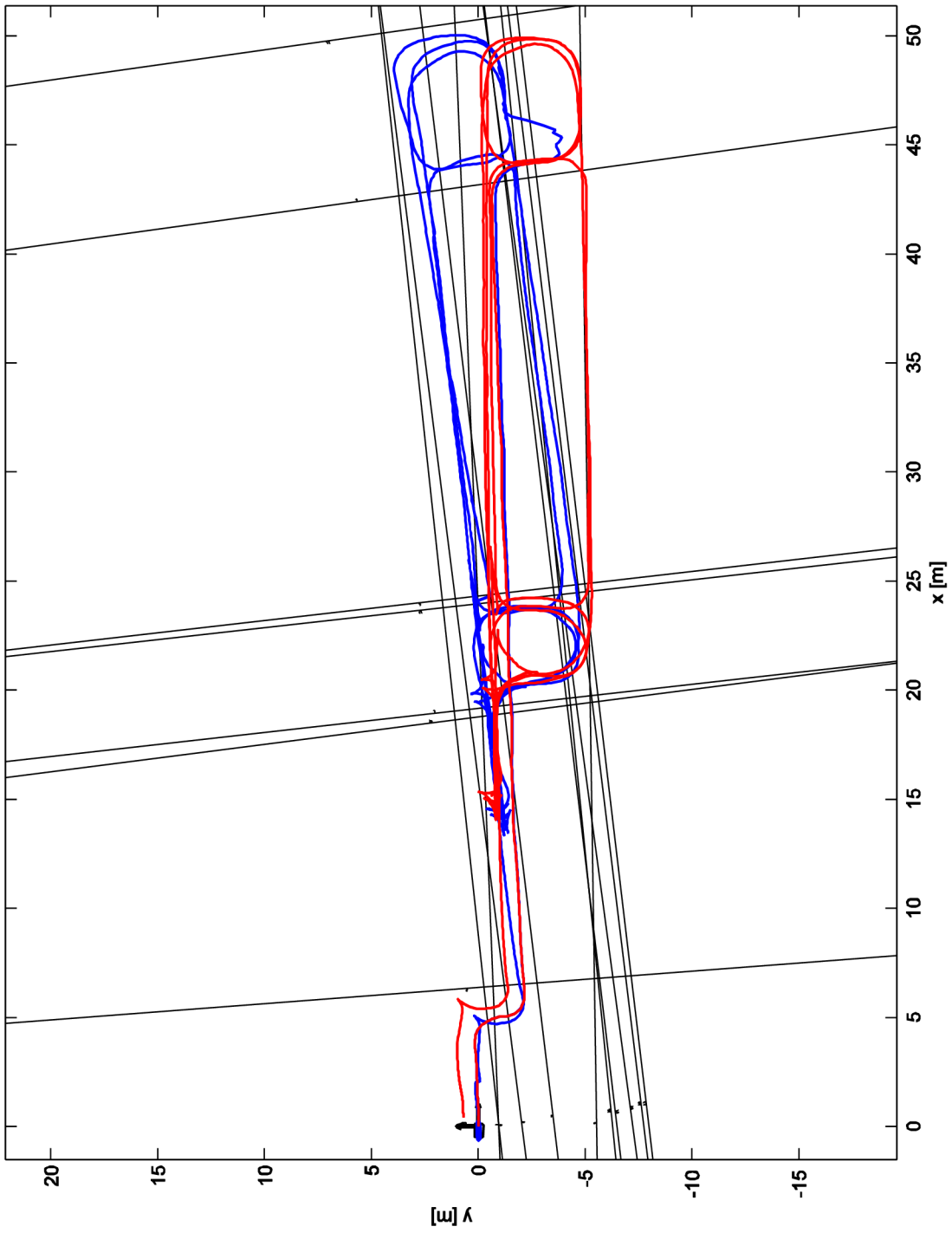
```
function [r,A_out,path] = predict_new3(r,enc);
%VSTUPY:
% r = struktura s aktualnimi stavy robota a jeho kovariancni matici
% enc = data z odometrie (enkodery+IMU)

global qq;

% Konstanty ze souboru nastaveni
qq = enc.params.qq;

nSteps = length(enc.steps);
% Otestovani zda mezi dvema merenimi laserového skeneru probehlo
% alespon jedno mereni odometrie
if nSteps > 0,

  % Inicializace
  % nacteni stavu a kovarinacni matice z posledniho kroku
  % ! kovariance pozice robota v DP jako 'Prr', zde znaceno 'C' !!
  from.x = r.x(1); from.y = r.x(2); from.theta = r.x(3);
  from.C = r.C;
  path(1).x = r.x;
  path(1).C = r.C;

  %akumulace jakobianu modelu predicke A pri vice krocich
  %(po skonceni vsech kroku = vystupni jakobian)
  A_acc = eye(3);

  % Simulacni smycka pro provedeni zpracovani vsech mereni odometrie
  % mezi dvema merenimi laseroveho skeneru
  for i = 1:nSteps,

    % vypocet zmeny stavu
    delta_s=((enc.steps(i).data1/1064*(0.15*pi))+...
            (enc.steps(i).data2/1064*(0.15*pi)))/2;
    delta_x=delta_s*cos(from.theta);
    delta_y=delta_s*sin(from.theta);
    delta_theta=enc.steps(i).data3;

    % aktualizace stavu
    to.x=from.x+delta_x;
    to.y=from.y+delta_y;
    to.theta=from.theta+(-delta_theta);

    % inicializace jakobianu A pro aktualni krok
    A = eye(3);

    %doplneni jakobinanu A pro aktualni krok
    A(1,3)=-delta_y;
    A(2,3)=delta_x;

    %aktualizace "akumulace" jakobianu A
    A_acc = A_acc*A;
```

```
    %vypocet matice Q
    W=[delta_x;delta_y;delta_theta];
    Q=W*qq*W';

    %aktualizace matice C pro kovarinaci robota
    %(v DP oznaceno jako 'Prr', zde znaceno 'C')
    to.C=A*from.C*A'+Q;

    %Zapsani prubeznych stavu a kovarianci matice
    path(i+1).x = [to.x; to.y; to.theta];
    path(i+1).C = to.C;

    %prepsani promennych pro dalsi iteraci
    %koncovy bod --> pocatecni bod
    from.x = to.x; from.y = to.y; from.theta = to.theta;
    from.C = to.C;
  end;

  %zapsani finalnich stavu, kovariancni matice a jakobianu A
  %do vystupnich hodnot
  r.x = [to.x; to.y; to.theta];
  r.C = to.C;
  A_out = A_acc;

  %osetreni chyby vypoctu matice C
  if det(to.C) < 0 ,        % C is negative definit -> PROBLEM!
    disp('--> applyodometrydd: cov negative definite!');
  end;

% Pokud mezi mezi dvema merenimi laserového skeneru neprobehlo ani
% jedno mereni odometrie - zadna zmena stavu a ani kovarinacni matice
else
  A_out = eye(3);
  path(1).x = r.x;
  path(1).C = r.C;
end;
```

Obr. 9.1: Zdrojový kód funkce pro predikci polohy robota



Obr. 9.2: Mapa z ukázkového testu č.1 pro demonstraci výsledků SLAM algoritmů v praxi (červená - pouze data z odometrie, modrá - data ze SLAM, černá - orientační body (přímky))



Obr. 9.3: Mapa z ukázkového testu č.2 pro demonstraci výsledků SLAM algoritmů v praxi (červená - pouze data z odometrie, modrá - data ze SLAM, černá - orientační body (přímky))