VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

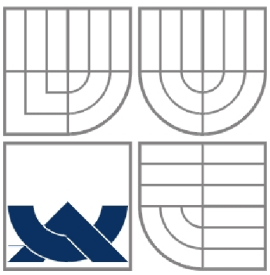
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

LOKALIZACE MOBILNÍHO ROBOTA A MAPOVÁNÍ POMOCÍ KALMANOVY FILTRACE

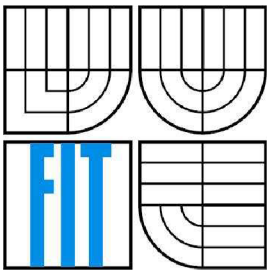
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MATĚJ CAHA
AUTHOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

LOKALIZACE MOBILNÍHO ROBOTA A MAPOVÁNÍ POMOCÍ KALMANOVY FILTRACE

MOBILE ROBOT LOCALIZATION AND MAPPING BASED ON KALMAN FILTERING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MATĚJ CAHA
AUTHOR

VEDOUCÍ PRÁCE — ING. DAVID HERMAN
SUPERVISOR

BRNO 2011

Abstrakt

Cílem této bakalářské práce je navrhnout metodu lokalizace a mapování (zvané SLAM) pomocí mobilního robota s ohledem na jeho senzorický systém a pohyb ve vnitřních statických prostorách. Pro metodu SLAM využít Kalmanovy filtrace a toto navržené řešení implementovat. V této práci je také rozebrána problematika odometrie a problematika extrakce a asociace vhodných význačných bodů.

Abstract

The goal of this bachelor's thesis is design a method for localization and mapping (called SLAM) for mobile robot considering its senzoric system and movement in indoor static enviroment. In this thesis is analyzed the odometry problem and the problem of landmarks extraction and association.

Klíčová slova

SLAM, Simultánní lokalizace a mapování, Kalmanova filtrace, Extended Kalman Filter, odometrie, význačné body v prostředí, extrakce význačných bodů, asociace význačných bodů, architektura mobilního robota

Keywords

SLAM, Simultaneous localization and mapping, Kalman filtration, Extended Kalaman Filter, odometry, landmarks in enviroment, landmarks extraction, landmarks association, mobile robot archotecture

Citace

Caha Matěj: Lokalizace mobilního robota a mapování pomocí Kalmanovy filtrace, bakalářská práce, Brno, FIT VUT v Brně, 2011

Lokalizace mobilního robota a mapování pomocí Kalmanovy filtrace

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Davida Hermana a uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Matěj Caha
18. května 2011

Poděkování

Děkuji svému vedoucímu bakalářské práce za trpělivost a ochotu při zodpovídání mých dotazů. Také za jeho odbornou pomoc, cenné rady a konstruktivní připomínky při vypracovávání mé bakalářské práce.

Obsah

1 Úvod

Robot, jak jej český spisovatel Karel Čapek v roce 1921 prvně definoval, představoval poslušný, inteligentní stroj podobající se člověku a plnící práci, která se člověku dělat nechtěla. Od té doby uplynulo 90 let a pojem "robot" doznal celosvětového rozšíření, nejenom jako slovo, ale hlavně jako stroj ulehčující člověku práci, jak jej původně, s trochou nadsázky, představil Čapek. Robot je dnes definován jako programovatelný systém, který je schopen vnímat a rozpoznávat okolní prostředí, popřípadě manipulovat s předměty, nebo se sám pohybovat.

V dnešní době nalezneme aplikaci robota v nejrůznějších oborech, ať už to je primitivní robot, který je vytvořen k jedinému účelu a v podstatě vykonává jedinou předem danou činnost, nebo to je plně automatizovaný robot schopný vlastního rozhodování a plánování. V posledních letech zažívá robotika všeobecně velký rozmach, což je mimo jiné způsobeno také růstem v oblasti nových technologií, výpočetní techniky a výpočetních metod. Za autonomní inteligentní roboty bych položil jeden příklad, kterým je humanoid Asimo firmy Honda (1), aktuálně nejpokročilejší humanoid na světě, který je navržen pro pomoc a asistenci lidem.

Další hodně prozkoumávanou oblastí je vývoj autonomních vozidel. Hlavní ideou je vytvořit takové vozidlo, které se dokáže samostatně pohybovat po světě bez nutnosti zásahu člověka. S touto problematikou je spojeno mnoho dílčích problémů, jako třeba mechanické a softwarové řešení, nebo senzorické vybavení vozidla. Velmi diskutovaným tématem je právě lokalizace mobilního robota v prostředí. Vývoj v této oblasti je silně motivován jednak požadavky moderní doby, ale také různými soutěžemi na profesionální i amatérské úrovni, jejichž výhrou často bývá, kromě řádného zadostiučinění, také tučná suma. Jako nejznámější soutěž tohoto druhu bych uvedl DARPA Grand Challenge (2), kde autonomní vozidla musí urazit více než 100 km a to bez zásahu člověka.

Této problematice se věnuje i tato bakalářská práce. Bude zaměřená především na metodu SLAM (Simultánní lokalizace a mapování) založenou na Kalmanově filtraci. Lokalizace mobilního robota v prostředí, ve kterém se pohybuje, je nelehký úkol. Robot musí být schopen rozeznávat okolní prostředí a porovnávat jej s modelem světa (tzv. mapou), který mu byl poskytnut, nebo si tento model sám postupně tvořit. Záleží tedy na senzorické výbavě robota a použité metodiky zpracování senzorických dat.

V kapitole Model mobilního robota stručně popíšu architekturu mobilního robota, kde se předně zaměřím na model použitého robota, respektive robota, s jehož záznamem jízdy pracuji. Dále v kapitole Odometrie rozeberu tento jednoduchý proces sledování pozice robota a opět popíšu konkrétní aplikaci odometrie na typ použitého mobilního robota. V kapitole SLAM proces budu diskutovat problematiku lokalizace a mapování pomocí mobilního robota, opět se zaměřením na cíle této práce (viz kap. 1.1). V kapitolách Význačné body a Návrh metody SLAM založené na Kalmanově filtraci se již zaměřím na konkrétní navrhované řešení SLAM metody založené na Kalmanově filtraci. Potom v kapitole Implementace stručně popíšu konkrétní implementaci tohoto návrhu a v kapitole

Výsledky testování předvedu dosažené výsledky v závislosti na nastavení parametrů. Nakonec v kapitole Závěr shrnu tyto výsledky a zhodnotím přínos práce.

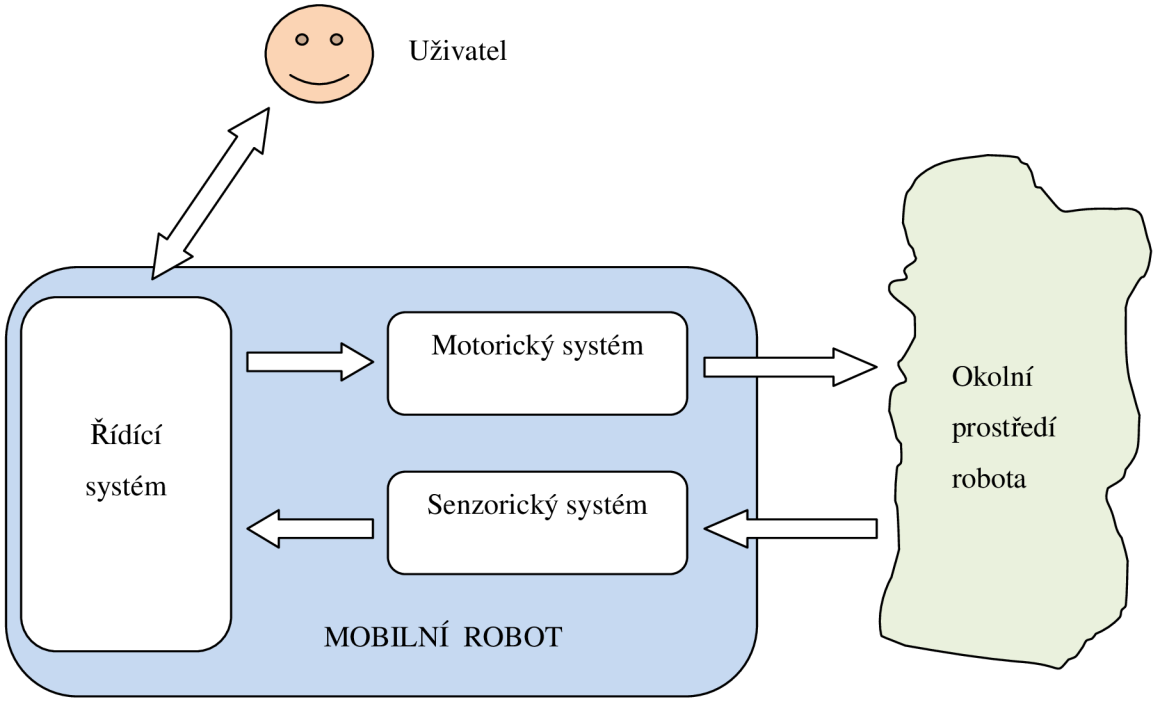
1.1 Stanovení cílů

Cílem této bakalářské práce je navrhnout metodu SLAM založenou na Kalmanově filtraci s ohledem na senzorický systém robota, za předpokladu, že se robot pohybuje ve vnitřních statických prostorách. Navrženou metodu implementovat v jazyce C++ a implementaci ověřit na záznamu jízdy robota (v režimu post-processing).

2 Model mobilního robota

V této kapitole předvedu blokovou architekturu autonomních mobilních robotů, jak byla definována různými autory. Jednotlivé bloky popíšu, přičemž se zaměřím na obecné principy a možná úskalí.

Architektura mobilních robotů je rozdělena do tří základních bloků. Jedná se o motorický systém, který zajišťuje pohyb robota, dále pak senzorický systém sloužící k vnímání/rozpoznání okolního prostředí a v neposlední řadě systém řídící, který řídí činnost robota a také slouží ke komunikaci s obsluhou, tento blok zde popisovat nebudu, protože je nad rámec této práce. Na Obr. 2.1 jsou znázorněny vazby mezi jednotlivými systémy (3). V následujících podkapitolách tyto bloky popíšu.



Obr. 2.1: Schéma architektury mobilního robota

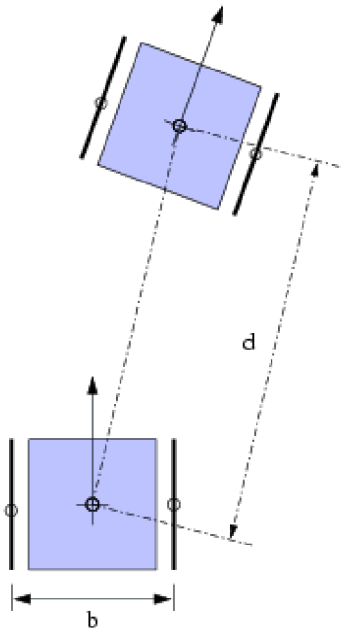
2.1 Motorický systém

Motorický systém zajišťuje pohyb mobilního robota v prostředí. Pohybový aparát robota musí být přizpůsoben prostředí, ve kterém má robot operovat. Nejčastější typ mobilních robotů jsou roboti pozemní, kteří se pohybují po pevném povrchu. Pohybový aparát takových robotů pak může být kolový, pásový, chodící, plazící atd. Právě tímto typem robota (s diferenciálním kolovým podvozkem), se budu zabývat. Další typy robotů jsou například akvatický, pohybující se na vodní hladině, nebo pod ní, a je tomu přizpůsoben lodním šroubem, nebo vodními tryskami, dále robot létající, který se pohybuje ve vzduchu, svým pohybovým aparátem může napodobovat helikoptéry,

letadla, případně i ptáky (3). Pohyb robota je zaznamenáván různými enkodéry. Informaci získanou z enkodérů pohybu je potřeba transformovat na změnu pozice a natočení robota (4). Tímto se zabývá odometrie, ke které se dostanu v kapitole 3.

Nyní zde popíšu konkrétní pohybový aparát robota, jehož záznam jízdy (viz kap. 2.3) zpracovávám. Jedná se o mobilního robota s kolovým podvozkem s diferenciálním řízením.

Diferenciální podvozek sestává z dvou nezávisle řízených kol, případně ještě opěrným bodem (např. nezávislé třetí kolečko), pro udržení stability. Tento typ robota má jednoduché řízení pohybu, jehož základním principem je otáčení kol nezávisle na sobě. Pokud se obě kola točí stejnou rychlostí a stejným směrem, robot jede rovně. Pokud se ale kola točí stejnou rychlostí a opačným směrem, pak se robot otáčí na místě kolem svého středu mezi koly. Tento bod je většinou brán jako bod referenční. Při otáčení kol stejným směrem, ale jinou rychlostí, se bude robot pohybovat po kružnici s poloměrem $r = \frac{1}{2} \cdot b \cdot \frac{vL+vR}{vL-vR}$, kde b je vzdálenost kol a vL a vR jsou rychlosti otáčení levého a pravého kola. Změna natočení robota je dána vztahem $\theta = \frac{dL-dR}{b}$, kde θ je úhel určující změnu natočení robota v radiánech a dL, dR jsou vzdálenosti ujeté levým a pravým kolem. Celková ujetá vzdálenost se spočítá jako průměr vzdáleností ujetých jednotlivými koly $d = \frac{dl+dR}{2}$. Více o stavbě tohoto podvozku napoví Obr. 2.2. Popis diferenciálního řízení kolového podvozku byl převzat z (5).



Obr. 2.2: Model diferenciálního podvozku

Pro obecnější přehled vypíšu i další způsoby řízení robota s kolovým podvozkem (5):

- Ackermanovo řízení - kategorie nonholonomních robotů, to znamená, že neumí měnit svou pozici a natočení nezávisle na sobě. Podvozek je dvounápravový a změna polohy a pozice sestává ze dvou kroků - natočení směrové nápravy a následný pohyb vpřed, či vzad. Toto řízení zahrnuje naprostou většinu dnešních automobilů. Čtyřkolový podvozek je často přepočítáván na tříkolový, to kvůli nutnosti dobře definovat střed otáčení a nemít žádné kolo ve smyku.
- Diferenciální řízení - popsáno výše.
- Řízení s všesměrovým podvozkem - jak již název napovídá, tento podvozek umožňuje robotovi pohyb všemi směry. Je to dáno speciální konstrukcí kol.

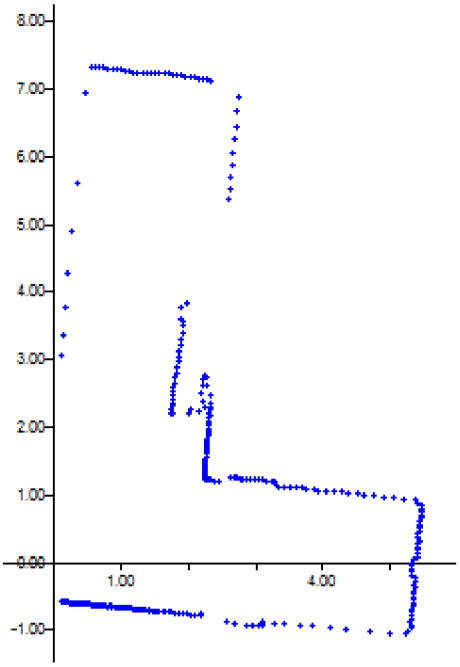
2.2 Senzorický systém

Podobně jako má člověk smyslové vnímání, tak má robot senzorický systém, který mu slouží k získávání a zpracování informací o jeho okolí, případně robotu samotném. Senzory můžeme dělit interní na externí.

Interní senzory monitorují stav robota samotného, jako například stav vybití baterie, jeho rychlost, pozice, natočení a další vnitřní parametry. Externí senzory slouží k vnímání okolního prostředí. Může se jednat o obecné vlastnosti prostředí (teplota, zvuk, světelnost, atp.), nebo informace o objektech v okolí (vzdálenost, tvar, barva, rozměry, atp.), nebo také charakteristiky fyzikálních polí (magnetické, gravitační, atp.).

Metoda SLAM (viz kap. 4) využívá informace jak z interních (pohyb robota), tak z externích senzorů (korelace pozice robota dle měření okolního prostředí). Pro vnímání prostředí při lokalizaci a mapování jsou nejčastěji používány senzory měřící vzdálenost, jako třeba sonar, nebo laserový senzor a kamery, které získávají obrazovou informaci o okolí robota.

Robot, jehož záznam jízdy zpracovávám, je vybaven laserovým dálkoměrem, který je v poslední době hojně využíván, protože poskytuje poměrně přesnou informaci o vzdálenostech objektů v okolí. Laserový dálkoměr funguje na principu měření doby letu vyslaného signálu, nebo porovnání vlnových délek vyslaného a přijatého paprsku. Oproti sonaru jsou laserové dálkoměry mnohem spolehlivější a rychlejší, protože jsou odproštěny mnohých chyb, které zatěžují právě sonar. Laserové dálkoměry nemají pozorovatelnou závislost na změnách vlastností prostředí (vlhkost, teplota, tlak). Laserové paprsky jsou vysílány v jedné rovině, což znamená, že v jednom okamžiku získáme informaci o vzdálenostech v celém rozsahu intervalu úhlů, který často bývá kolem 180º (viz Obr. 2.3). To může být nevýhodou laserových dálkoměrů, protože nejsou schopny detekovat překážky, které jsou nižší, nebo jsou výše položené, než snímaná rovina. Také může nastat nesprávná detekce překážek, jejichž tvar se mění s výškou. Proto je vhodné kombinovat laserové a sonarové senzory (4).



Obr. 2.3: Příklad výsledku jednoho měření vnitřních prostor laserovým dálkoměrem.

2.3 Záznam jízdy robota

V předchozích podkapitolách jsem popsal motorický a senzorický systém robota. Protože veškeré informace, se kterými robot pracuje, jsou v digitální podobě, lze je jednoduše zaznamenat pro pozdější zkoumání, experimentování, či opakované prohlížení. Existuje mnoho různých formátů, neboť si každý programátor může stanovit svůj vlastní.

Já jsem ale pracoval s jistým standardizovaným formátem RAWLOG, jak jej představili vývojáři MRTP (The Mobile Robot Programming Toolikt) (6). Tento formát může nabývat dvou různých podob. V prvním případě se do souboru ukládají všechna měření všech senzorů (odometrie, dálkoměry, kamera, atp.) chronologicky za sebou a všechny tyto záznamy jsou vedeny jako pozorování. Kdežto v druhé podobě RAWLOG formátu se záznamy vedou jako sdružené n-tice měření, která jsou prováděna ve stejný čas. Měření získané odometrií je zaznamenáno jako akce a měření externích senzorů jsou zaznamenány jako pozorování. Já využívám této druhé podoby, protože je velmi přijatelná pro princip lokalizace a mapování (viz kap. 4).

3 Odometrie

Odometrie je proces, při kterém se transformují data přijatá z enkoderů pohybu na změnu pozice a natočení. Je to nejjednodušší metoda pro sledování pozice mobilního robota. Odometrie nabízí okamžitý a docela přesný výsledek, který je počítaný z pulsů generovaných otáčením kol robota (3). Ideou odometrie je získání absolutní polohy integrací informací, které s časem přibývají (4):

$$\overrightarrow{x_1} = \overrightarrow{x_0} + \int_{t_0}^{t_1} \frac{d\vec{x}}{dt} dt \tag{3.1}$$

kde $\frac{d\vec{x}}{dt}$ značí změnu pozice a $\overrightarrow{x_0}$ a $\overrightarrow{x_1}$ značí pozici robota v čase t_0, respektive t_1.

Pro diferenciální řízení robota, s jehož záznamem pracuji, lze absolutní polohu vyjádřit následující maticí (odvozeno z kap. 2.1):

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{2\pi r}{P} \frac{(n_r + n_l)}{2} \cos\theta_{t-1} \\ \frac{2\pi r}{P} \frac{(n_r + n_l)}{2} \sin\theta_{t-1} \\ \frac{2\pi r}{P} \frac{(n_r + n_l)}{b} \end{bmatrix} \tag{3.2}$$

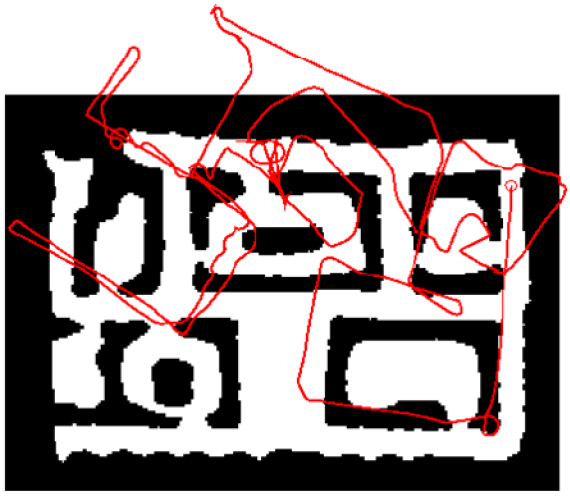
kde $\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}$ a $\begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix}$ jsou souřadnice robota v kartézské soustavě souřadnic a jeho natočení v čase t a $t-1$, dále r a P jsou poloměr kola a počet pulsů na jednu otáčku, n_r a n_l jsou počet pulsů z pravého a levého čidla v časovém intervalu $< t-1, t >$ a b je vzdálenost mezi koly.

Odometrie je levné a jednoduché řešení sledování pozice robota. Umožňuje provádět měření s vysokou frekvencí a na krátké vzdálenosti je velice přesná. Avšak problémem odometrie je právě její hlavní idea, integrace informací s časem přibývajících. Při měření krátkých vzdáleností je chyba odometrie opravdu zanedbatelná, ale při měření delších vzdáleností je tato chyba postupem času a integrací dalších měření akumulována (přičítána). Hlavně akumulace chyb v natočení robota způsobuje v konečném důsledku velkou chybu v pozici robota, která se zvětšuje s délkou ujeté dráhy (3). Toto činí odometrii jakožto samostatné řešení lokalizace nepoužitelnou. Chyby měření jsou způsobeny řadou faktorů a dělí se do dvou kategorií: systematické a nesystematické.

Systematické chyby měření jsou způsobené nedokonalým návrhem motorického systému a jeho implementací. Chyby této kategorie většinou zůstávají v čase konstantní a lze je proto detekovat a eliminovat. Hlavní myšlenka kompenzace těchto chyb je založena na experimentálním měření, kdy zjišťuji odchylku ujeté trajektorie a následné kalkulaci parametrů systému (3). Nejčastější příčiny systematických chyb jsou:

- rozdílný průměr kol,

- průměrný průměr kol se liší od nominálního,
- nepřesnost v určení vzdálenosti kol,
- nesouosost kol,
- rozlišení enkodérů,
- omezená maximální vzorkovací frekvence enkodérů.



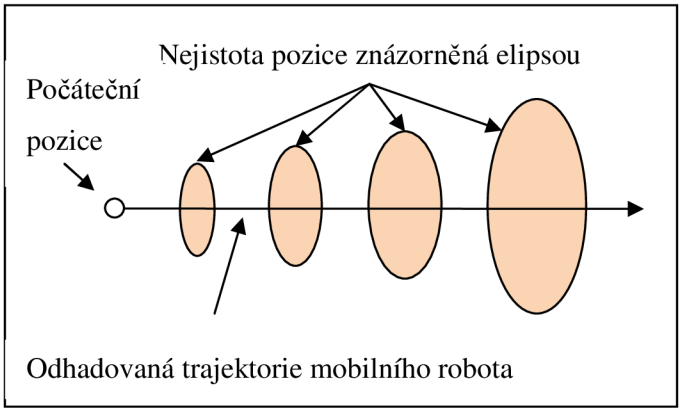
Obr. 3.1: Vliv chyb odometrie na vypočtenou trajektorii mobilního robota. Převzato z (5)

Nesystematické chyby měření vznikají nepředpokládanými vlastnostmi prostředí, které při jízdě působí na motorický systém robota, a chováním robota v nestandardních situacích. Tyto chyby nelze detekovat a eliminovat, protože se vyskytují neočekávaně a náhodně. Nesystematické chyby dokážou způsobit poměrně velkou odchylku mezi naměřenou a skutečnou pozicí a to i pokud se vyskytnou v krátkém časovém okamžiku. Nejčastější příčiny nesystematických chyb jsou:

- nerovnost povrchu, po kterém se robot pohybuje,
- neočekávané objekty na podlaze,
- zrychlení pohybu nad možnosti vzorkování měřícího systému,
- prokluz kol zapříčiněný různými vlivy,
    - kluzká podlaha,
    - prudká akcelerace,
    - interakce s jinými předměty v prostředí (např. náraz do překážky),
    - rychlé zatočení vedoucí ke smyku.

Je potřeba rozlišit systematické a nesystematické chyby a ty systematické se pokusit eliminovat. Jejich největší problém je to, že se konstantě akumulují (4). Vliv chyb odometrie na vypočtenou trajektorii robota je znázorněn na Obr. 3.1. I přes toto všechno je odometrie často základem lokalizačních algoritmů, protože poskytuje slušný odhad pozice robota mezi jednotlivými měřeními, nebo když měření ostatních senzorů chvilkově nemůže být zahrnuto. Tyto algoritmy potom počítají

s nejistotou správnosti odhadu pozice na základě odometrie. Tato nejistota je znázorněna elipsami okolo odhadované pozice robota (viz Obr. 3.2). Tento problém nejistoty odhadu pozice bude diskutován v kapitole věnované SLAM (viz kap. 4).



Obr. 3.2: Elipsy znázorňující, s délkou ujeté trajektorie zvětšující se, nejistotu odhadu pozice mobilního robota

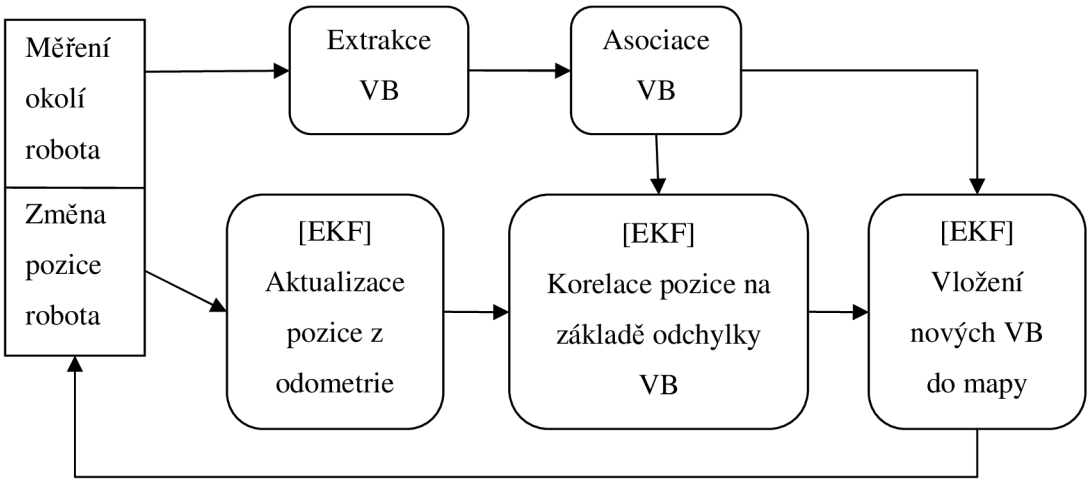
4 SLAM proces

V této kapitole budou shrnuty poznatky ze zdrojů (3) a (7).
V předchozích kapitolách jsem představil základní charakteristiku mobilních robotů a nyní se budu věnovat samotnému jádru této práce, a to je metoda SLAM (=Simultaneous localization and mapping) založená na EKF (=Extended Kalman Filter). Mobilní robot musí znát svou přesnou polohu v okolním světě a také by měl znát mapu svého okolí. Toto zajišťuje právě metoda SLAM, která je brána spíše jako princip, než konkrétní algoritmus, ten se naopak liší dle použitého přístupu.

Obecně je metoda SLAM reprezentována takovouto dekompozicí problému: odhad aktuální pozice, extrakce význačných bodů z okolí, porovnání těchto bodů s body v mapě, aktualizace polohy robota a pozicí význačných bodů v mapě. Obr. 4.1 ilustruje schéma SLAM procesu založeném na EKF.

Stěžejním pilířem metody SLAM je vnesení do výpočtu pozice robota měření jeho okolního prostředí. Odometrie (viz kap. 3) je jako samostatný lokalizační prostředek nevyhovující. Pro vnášení měření okolního prostředí do výpočtu pozice robota je třeba znát očekávanou podobu tohoto prostředí. To zajišťuje vnitřní mapa (model okolního světa), kterou má robot uloženou. Tato mapa může být robotu známá předem, pak se jedná o lokalizaci ve známém prostředí, nebo si robot tuto mapu tvoří sám, a pak se jedná o lokalizaci v neznámém prostředí, což je náplní této práce. V následujících podkapitolách přiblížím samostatně lokalizaci a mapování.

Při procesu SLAM lze říci, že se jedná o podobný problém, jako myslitelská hříčka "Co bylo dřív, kuře nebo vejce?". Ve SLAM je totiž lokalizace závislá na vnitřním modelu okolního světa (mapa) a porovnání aktuálních měření s tímto modelem a naopak tvorba tohoto modelu okolního světa je závislá na dobrém zlokalizování mobilního robota. Při tomto přístupu nepracuji s celou mapou, ale tzv. mapou význačných bodů, jejichž problematiku popíšu v kapitole 5.



Obr. 4.1: Schéma SLAM procesu založeném na EKF

4.1 Lokalizace

Cílem lokalizace je zjistit přesnou polohu robota pomocí jeho senzorických dat. Kromě odometrie se využívají další senzory, jako např. proximitní senzory (sonar, laser), nebo kamera. Lokalizace se dělí na globální a lokální lokalizaci.

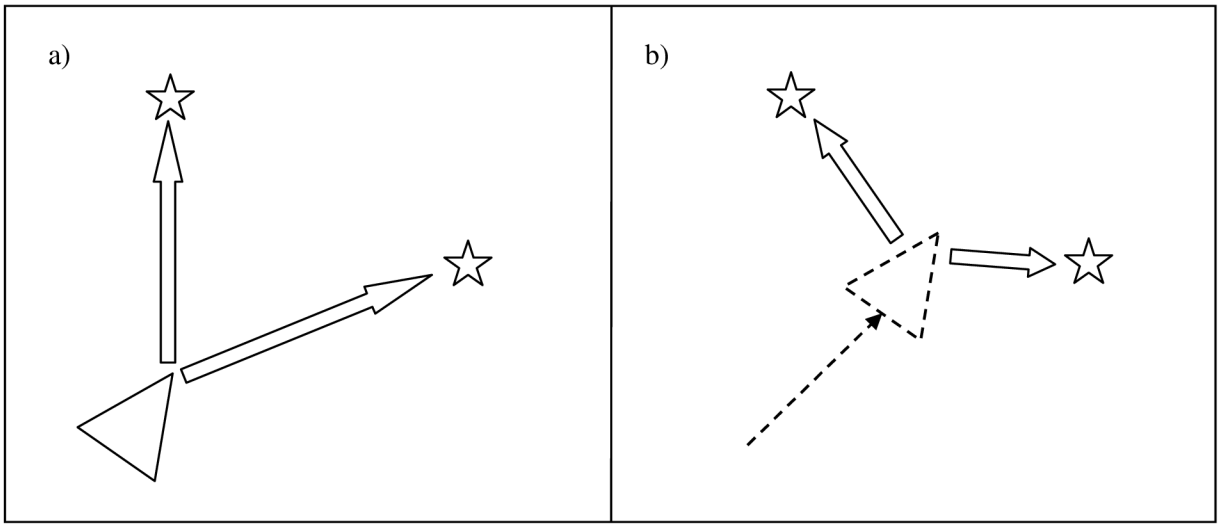
Globální (absolutní) lokalizace nastává v případě, že není známa počáteční poloha robota, což nastane např. ihned po zapnutí. Řešení tohoto problému spočívá v získání senzorického měření z několika různých pozic a po každé změně polohy se vygenerují možné předpokládané pozice a porovnají se s kandidáty z předchozího kroku.

Lokální (kontinuální) lokalizace je častějším přístupem SLAM. Jedná se o korekci polohy během jízdy robota. Je potřeba znát počáteční polohu robota. Tato metoda není schopná se vyrovnat se ztrátou pozice (např. případ únosu). Při této metodě je poloha robota vyhledávána v blízkém okolí odhadované pozice (odhad např. pomocí odometrie), ne v celém stavovém prostoru. Lokální lokalizace je lehčí než globální a skládá se ze čtyř základních kroků(viz také Obr. 4.2 a Obr. 4.3):

1. Přemístění robota.
2. Odhad pozice z řídících povelů (případně odometrie).
3. Získání měření okolí robota.
4. Porovnání měření s modelem světa (mapou).

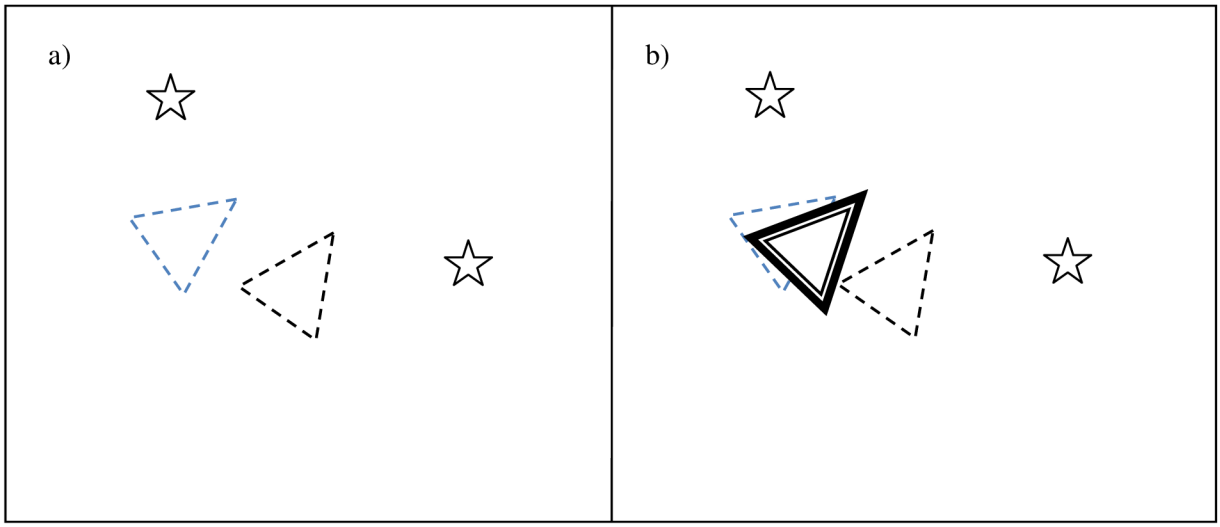
Naprostá většina lokalizačních metod je založena na pravděpodobnostním přístupu. Senzorické měření je zatíženo chybou, která má Gaussovo rozložení. Kromě Kalmanova filtru vysvětleného níže se pro lokalizaci využívají metody Monte Carlo, která je založená na reprezentaci odhadu polohy robota pomocí hustoty pravděpodobnosti. Obě tyto metody využívají dva základní kroky, a to sice predikci a korekci. Dále bych zmínil Markovovu lokalizaci, která využívá pravděpodobnostního rozdělení na všech možných pozicích v prostředí (4).

V praxi je popsáno několik lokalizačních případů, které se v mobilní robotice řeší.



Obr. 4.2: a) První měření robota (hvězdy jsou význačné body. b) přemístění robota a odhad jeho polohy na základně odometrie + následné měření

Nejjednodušším případem je lokální lokalizace, kdy robot zná svou počáteční polohu a jeho úkolem je pouze sledovat svůj pohyb na základě odometrie a senzorických měření. Speciálním případem je navigace robota, kdy robot zná i mapu celého prostředí a je pomocí ní (lokalizace v ní) navigován za svým cílem. Dalším, již složitějším případem je globální lokalizace, kdy robot nezná svou počáteční polohu a je potřeba, aby se nejdříve zorientoval v prostředí. Nejsložitější je potom případ únosu, kdy je dobře zlokalizovaný robot přenesen na úplně neznámé místo. Rozdílnost od globálního lokalizačního problému je v tom, že robot stále pevně věří, že je zlokalizován správně, ačkoliv se ve skutečnosti nachází na úplně jiném místě (4).



Obr. 4.3: a) černě čárkovaný trojuhelník značí odhadovanou polohu robota z odometrie, kdežto modře čárkovaný značí odhad polohy na základě provedeného měření. b) konečná poloha je potom váženým průměrem těchto poloh, kde většinou poloha odpovídající senzorickému měření má vyšší váhu

4.2 Mapování

Mapování je vytváření modelu světa (dále jen mapa). Mapa napomáhá robotu při pohybu prostředím, aby se mohl pohybovat bezkolizně, nebo aby mohl plánovat svou cestu. Stavba mapy závisí např. na použitých senzorech, úrovni požadované abstrakce, nebo cílu jejího dalšího využití. Robot by měl umět tvořit kvalitní mapu prostředí, ať už slouží jenom k jeho lokalizaci, nebo je vytvoření mapy cílem mise robota vypuštěného do neznámého prostředí. Lokalizační metody jsou závislé na vytvořené mapě, protože porovnávají senzorická měření okolí s touto mapou a tím korigují aktuální odhadovanou polohu robota. Nejrozšířenější typy map (4):

- Senzorické mapy - popsáno níže.
- Geometrické mapy jsou založeny na reprezentaci okolí pomocí geometrických primitiv (úsečky, čtverce, kružnice, atd.).
- Topologické mapy neuchovávají informaci o metrikách, ale pouze o vztazích mezi objekty.

- Symbolické mapy slouží pro komunikaci robota s obsluhou. Obsahují informace, které robot nemůže zjistit svými senzory, ale jsou potřebné pro komunikaci s obsluhou, jako například symbolická jména objektů.

Senzorické a geometrické mapy spadají do kategorie metrických map, to znamená, že je v mapách uchována informace o metrikách prostředí (vzdálenosti, velikosti, natočení objektů vůči sobě, atd.).

V této práci se zabývám vytvořením senzorické mapy, která reprezentuje prostředí uživatelsky vhodným způsobem. Tento typ mapy obsahuje buď přímá měření senzorů, nebo poskytuje první prostor pro fúze jednotlivých měření. Senzorická mapa je postavená na pevné souřadné soustavě. Základním představitelem senzorických map je mřížka obsazenosti, která rozděluje prostor na dílčí buňky, nejčastěji body, z nichž každý udržuje míru obsazenosti/volnosti. Vytvoření map rozsáhlých území pomocí mřížky obsazenosti v podstatě nelze technicky řešit, protože klade velké paměťově nároky pro uložení informací v mřížce.

Pro práci SLAM procesu je využívána mapa význačných bodů, která je rovněž metrická, ale neposkytuje uživatelsky vhodnou reprezentaci prostředí.

5 Význačné body

Již dříve jsem popsal, že princip odhadu polohy robota spočívá v porovnání senzorického měření aktuálního okolí robota s mapou prostředí (modelem světa, který si robot vytvořil, nebo mu byl poskytnut). Pro lepší názornost uvedu příklad z lidského života. Dejme tomu, že se nacházíte v dobře známém domácím prostředí. Nyní si zavážete oči a snažíte se dostat z jedné místnosti do druhé. Jak ale víte, kam přesně máte jít? To zjistíte jednoduše pomocí dotýkání se objektů, zdí, dveřních rámů, apod. Všechny tyto doteky vám pomohou zjistit (alespoň odhadnout), kde se právě nacházíte. Robot tuto činnost provádí podobně, jeho senzory jsou prostředky pro vnímání okolí stejně, jako pro člověka jeho smysly (zrak, hmat, sluch, atd.).

Já se v této práci zaměřuji na porovnávání pomocí tzv. význačných bodů (dále jen VB; v anglické literatuře značeno jako "*landmarks*") prostředí. Tyto VB jsou takové body/části světa, které jsou pro dané prostředí typické a význačné. Při rozhodování jaké VB budou brány v potaz, záleží na prostředí, ve kterém se robot pohybuje. Pro pohyb ve vnitřních prostorách můžeme detekovat například rohy místností, výklenky dveří, rovné zdi, atp. Pro venkovní prostředí potom můžeme použít rohy budov, sloupy, stromy (ale ne celý les), a další překážky.

Základním předpokladem dobrého VB je jeho opakovaná detekovatelnost a to i z různých poloh robota. VB by měl být dostatečně jedinečný, aby bylo vhodné ho správně přiřadit i při pozdější detekci. Z toho plyne, že by VB neměly být moc blízko u sebe, protože může nastat špatné přiřazení při pozdější detekci jednoho z nich. Při rozhodování, jaké VB by měl robot rozeznávat, by se měla brát v úvahu jejich hojnost v prostředí. Robot by se neměl pohybovat dlouhou dobu bez možnosti detekce VB, mohlo by se stát, že se robot ztratí. Dalším požadavkem na VB je, že by měl být statický. To znamená, že by měl být detekovatelný stále na stejném místě. Použití člověka, nebo okolo jedoucího auta jako VB tedy není vhodné. Robot potřebuje jako VB pevné body, aby podle nich mohl korigovat svou odhadovanou pozici. Následující seznam shrnuje vlastnosti vhodných VB:

- znovuobjevitelnost,
- jedinečnost,
- hojnost v prostředí,
- statičnost.

5.1 Extrakce význačných bodů

V předchozí části jsem popsal problematiku VB a zde popíšu jejich extrakci ze senzorických měření. Existuje mnoho přístupů k extrakci VB ze senzorických měření, tyto přístupy silně závisí na typu používaných VB a také typu použitých senzorů okolního prostředí robota. Já se zde zaměřím na konkrétní řešení, které jsem navrhl a implementoval.

Nejdřív ale připomenu pár věcí, dle kterých jsem volil typ VB: robot se pohybuje ve vnitřních prostorách a je vybaven laserovým dálkoměrem. Z toho plyne, že je schopen měřit vzdálenosti

k překážkám, jako jsou například zdi. Právě tyto zdi mne budou zajímat, protože jsou snadno detekovatelné, jsou statické a ve vnitřních prostorách jsou všudypřítomné. Následující odstavce již popisují metodiku extrakce VB.

Pro detekci zdí v aktuálním senzorickém měření (znázorněném na Obr. 2.3) jsem zvolil metodu RANSAC detekci přímek (popsána níže). Dále jsem z detekovaných přímek extrahoval jeden konkrétní bod, který jsem určil jako VB. Samotnou extrakci tohoto bodu popíšu ke konci této podkapitoly.

RANSAC

V této kapitole jsem čerpal ze zdrojů (8) , (9) a (10).

RANSAC je název poskládaný ze slov RANdom SAmple Consensus. RANSAC je iterativní metoda k získání odhadu parametrů matematického modelu ze vstupního souboru dat. Jinými slovy lze touto metodou nalézt výskyt matematického modelu (přímka, kružnice, atp.) v obraze (snímek kamery, bodový výstup proximitních senzorů, atp.). Já jsem metodu RANSAC využil k detekci přímek v aktuálním měření laserového dálkoměru.

Princip metody RANSAC spočívá ve vybrání náhodného vzorku vstupních dat a aplikování metody nejmenších čtverců (více o této metodě ve zdroji (11)) pro nalezení přímky, která je tímto vzorkem dat definována. Potom jsou testovány všechny body měření, jestli odpovídají této přímce, čili jestli jejich vzdálenost od zkoumané přímky je menší než určená hraniční hodnota. Jestliže je počet těchto bodů (tzv. inlierů) větší než stanovená hodnota, která je nazývána konsensus, je tato přímka ještě znovu přepočítána pomocí všech jejich odpovídajících bodů a stanovena jako zeď (přidána do seznamu detekovaných přímek). Navíc body asociovány k přímce jsou odstraněny ze souboru dat a postup se opakuje nad zbylými daty. Celý algoritmus je stručně popsán na následujících řádcích:

Dokud platí

- počet neasociovaných bodů je větší než konsensus
- a bylo zatím provedeno méně než K pokusů detekce přímky

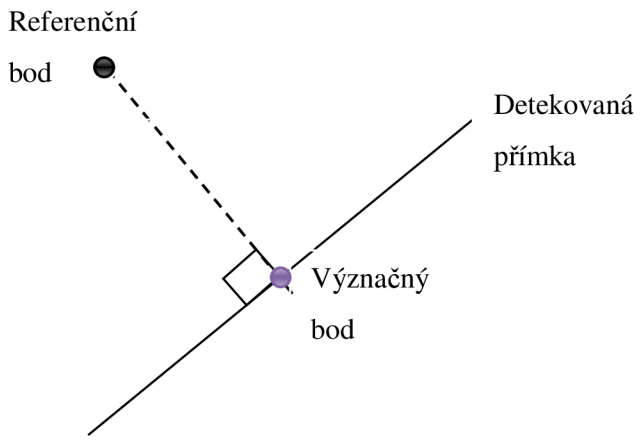
prováděj

- inkrementuj K o 1
- vyber vzorek S náhodných bodů ze vstupních dat
- najdi odpovídající přímku tomuto vzorku dat (pomocí metody nejmenších čtverců)
- zjisti kolik bodů celého vstupního souboru leží ve vzdálenosti X od této přímky
- jestliže je těchto bodů více než C (konsensus) proveď následující:
  - vypočti nové parametry přímky odpovídající všem asociovaným bodům
  - přidej tuto přímku do seznamu detekovaných přímek

```
o  odstraň asociované body ze vstupního souboru dat
```

Tento algoritmus je tedy závislý na nastavení vstupních parametrů K, S, X, C, které popíši zde:
K - počet maximálního počtu cyklů metody RANSAC (pokusů o nalezení přímky),
S - počet náhodných bodů zahrnutých do výpočtu inicializačních parametrů přímky,
X - maximální vzdálenost bodu od přímky, aby byl asociován k této přímce,
C - minimální počet asociovaných bodů, aby přímka byla uznána jako regulérní.

Výsledkem je seznam detekovaných přímek v aktuálním měření. Nyní ještě potřebuji určit VB. V začátku této kapitoly jsem popsal potřebné vlastnosti dobrého VB. Z přímek tedy potřebuji extrahovat body, které jsou statické v modelu světa, kterou si robot tvoří. Tento bod získám pomocí jiného pevně daného referenčního bodu, který si stanovím. VB potom bude bod, který leží na dané přímce a je nejblíže referenčnímu bodu (kolmice detekované přímky vedená z tohoto bodu prochází referenčním bodem - viz Obr. 5.1).



Obr. 5.1: Schéma znárňující výpočet význačného bodu, jako nejbližšího bodu detekované přímky a bodu referenčního

5.2 Asociace (přiřazení) význačných bodů

Cílem asociace význačných bodů (dále jen VB) je správné přiřazení VB z aktuálního měření okolí k VB, které jsou uloženy v paměti (tvoří mapu význačných bodů). Tento proces je velice důležitý a závisí na něm celé jádro EKF lokalizace. Pokud by asociace VB neprobíhala vůbec, nebylo by možné provést aktualizaci polohy robota, naopak, pokud by asociace VB probíhala špatně (byly by k sobě přiřazeny VB, které ve skutečném světě sobě neodpovídají), aktualizace polohy robota by probíhala na základě této špatné asociace, a proto by byla chybná. V praxi mohou nastat následující problémy:

- VB není detekován, nebo asociován v každém kroku.

- Detekce a zanesení do systému takového VB, který už nikdy jindy viděn není (pravděpodobně špatná extrakce VB).
- Chybné asociování dvou VB, které sobě ale ve skutečném světě neodpovídají.

První dva případy by měla ošetřit již samotná extrakce VB (viz kap. 5.1), protože v takovém případě není detekovaný VB vhodným VB. Často se ale stane, že i při dobře navržené extrakci občas zahrneme do systému špatný VB, a proto by asociace měla být navržená tak, aby minimalizovala problémy s tímto spojené. Třetí případ, chybné asociace VB, vede k velkým chybám při lokalizaci robota, protože si robot myslí, že se nachází jinde, než doopravdy je.

Nyní popíšu svůj návrh řešení asociace VB, který jsem v této práci implementoval. Vycházím s faktu, že extrakce VB není vždy stoprocentní a může se stát, že z aktuálního měření detekuje VB, který není vhodný pro zanesení do systému. Určil jsem si tedy, že nový detekovaný VB je spolehlivý, pokud je viděn alespoň N-krát po sobě. Tím omezím vnášení náhodných VB do systému. Řešil jsem to pomocí dvou databází VB, kde první databáze (dále nazývána databáze EKF) odpovídá vnitřnímu stavu EKF, čili obsahuje kompletní informace o VB zanesených do EKF. Druhou databází je databáze tzv. čekatelů na zanesení do systému. Jinými slovy zde ukládám každý detekovaný VB a až je splněna podmínka, že je vhodným VB pro zanesení do systému, tak ho z databáze čekatelů vyjmu a vložím do databáze EKF, kde už bude zohledněn do výpočtu stavu EKF. Tímto postupem z velké části eliminuji zanášení špatných VB do systému.

Samotnou asociaci řeším technikou zvanou nearest-neighbor (=nejbližší soused), která spočívá v přiřazení VB z nového měření k jeho nejbližšímu protějšku v databázi EKF, případně databázi čekatelů. Nejbližšího souseda spočítám pomoci euklidovské vzdálenosti těchto dvou bodů. Ještě je potřeba zjistit zda tyto dva, sobě nejbližší VB opravdu odpovídají jednomu VB v reálném světě. Tomuto procesu se říká validace a je popsán pod znázorněným algoritmem asociace. Dále jsem zde zavedl ještě další dvě databáze VB, kde první z nich (dále databáze asociovaných) uchovává VB z aktuálního měření, které projdou asociací s databází EKF a druhá je pro záznam nových VB, čili takových, které neprojdou asociací ani v databázi EKF ani v databázi čekatelů. Tyto nové VB jsou nakonec přidány do databáze čekatelů, to se ale provádí až v posledním kroku EKF (viz kap. 6). Následující algoritmus shrne proces asociace v pár bodech:

Pro každý detekovaný VB v aktuálním měření,

- najdi nejbližší VB v databázi EKF,
- proveď validaci těchto dvou VB,
    - jestliže je validace úspěšná, přidej tento VB z aktuálního měření do databáze asociovaných,
    - jestli validace není úspěšná, najdi nejbližší VB v databázi čekatelů,
        - jestli je jejich vzdálenost menší než X, spočítej z těchto dvou VB jeden průměrný VB

a vlož ho do databáze čekatelů místo VB původního,

- jestli je jejich vzdálenost větší než X, přidej tento nový VB do databáze nových VB.

kde X je nastavená mezní vzdálenost, při které mohu určit, že naměřený VB odpovídá VB v databázi.

Validace využívá vlastností EKF, přesněji toho, že si EKF uchovává míru nejistoty pozice VB. Úkolem validace je potom zjistit, zda naměřený VB leží v oblasti nejistoty pozice VB v databázi EKF. Validace je dána vzorcem: $v_i^T S_i^{-1} v_i \leq \text{threshold}$, kde v_i je inovace a S_i je inovace kovariance, obě proměnné jsou vysvětleny v kap. 6, a threshold je nastavená mez.

Asociaci VB lze tedy ovlivňovat nastavením hodnot X a threshold.

6 Návrh metody SLAM založené na Kalmanově filtraci

V této kapitole jsem čerpal ze zdrojů (7), (8), (12) a z vlastních poznatků.
EKF (Extended Kalman Filter) je nelineární odhadovací metoda používaná v navigačních systémech, GPS a hlavně taky v mobilní robotice. Je založena na principu lineárního Kalmanova filtru, pro který se často používá označení - vylepšený odhad plovoucího průměru. V nelineárních systémech, které jsou zatíženy nejistotou a pravděpodobností, se kromě jiných využívá právě EKF, což je nelineární verze Kalmanova filtru, která linearizuje okolo střední hodnoty a kovariance[1]. EKF se stala standardem v teorii odhadování stavu nelineárního systému(13).

Tato metoda obsahuje dva kroky:

1. predikce nového stavu,
2. korekce integrací nového měření.

Pro použití EKF v SLAM procesu, je nutno přidat krok aktualizace modelu světa, který si robot vytváří. Také jsou upraveny matice, se kterými EKF pracuje (o maticích v EKF více v kap. 6.1). Po nastavení těchto matic a počátečního stavu je EKF jen postupná aplikace rovnic (aplikace EKF v kap. 6.2). Po úpravě a upřesnění vypadá postup EKF v SLAM následovně (viz také Obr. 4.1):

1. predikce nového stavu z odometrie robota,
2. korekce stavu pomocí porovnání aktuálního měření okolí robota s modelem světa,
3. aktualizace modelu světa o nově detekované význačné body.

Takto upravený EKF jsem použil v této práci. V dalších pár řádcích rozepíšu jednotlivé kroky EKF.

V prvním kroku nám jde pouze o predikci nového stavu systému, tedy odhad nové polohy robota. K tomu si vezmeme na pomoc odometrii (viz kap. 3), jinými slovy, k předchozí poloze robota přičteme změnu polohy, kterou zaznamenala odometrie za poslední časový úsek.

V kroku druhém chceme upravit predikovaný stav podle aktuálně provedeného měření. Extrakci a asociaci význačných bodů již máme za sebou (popsána v kap. 5). Musíme tedy z predikované polohy spočítat také očekávanou pozici VB uložených v databázi, která je potom porovnána s aktuálně naměřenými VB k nim asociovaným. Obvykle je zde mírná odchylka, která je nazývána inovace. Tato inovace potom odpovídá rozdílu predikované polohy robota s polohou založenou na aktuálním měření robotova okolí. Výsledná poloha robota je tedy vážený průměr těchto dvou poloh, kde váha každé z nich je určená její neurčitostí. Obvykle bývá senzorické měření okolí robota přesnější než odometrie, proto by měla mít poloha vzniklá z měření okolí větší váhu, než z odometrie predikovaná poloha

[1] kovariance - střední hodnota součinu odchylek dvou náhodných veličin od jejich středních hodnot

robota. Tyto váhy ale systém vypočítává sám a bere v potaz jak neurčitost predikované polohy robota, tak neurčitost odometrie, měření i pozice VB. V tomto kroku je také aktualizována matice kovariancí, která určuje neurčitosti všech elementů systému.

Třetí krok je vložení nově detekovaných VB do systému. Vkládáme jejich pozice s neurčitostí vypočítanou z aktuální polohy robota a také nastavujeme informace o jejich vazbách k ostatním VB v systému.

6.1 Popis struktury EKF

V této podkapitole popíšu jednotlivé matice využívané EKF. Začnu nejdůležitějšími maticemi, což jsou matice stavu systému, Kalmanova zesílení a matice kovariancí. Poté představím Jacobiho matice[2] a nakonec matice reprezentující Gaussův šum odometrie a senzorického měření.

Matice stavu systému: X

Tato matice je jedna z nejdůležitějších v systému. Určuje totiž stav systému, čili aktuální polohu robota (souřadnice x, y a natočení θ) a pozice jednotlivých význačných bodů (souřadnice x, y). Tato matice stavu je vertikální, aby následně odpovídala všem rovnicím aplikace EKF. Matice X je znázorněna napravo. Její rozměry jsou 1 sloupec a 3+n*2 řádků, kde n je počet význačných bodů se kterými EKF pracuje. Je důležité, aby všechny odpovídající hodnoty byly stejných fyzikálních jednotek. Já používám metry pro vzdálenost a radiány pro úhel.

x_r
y_r
θ_r
x_1
y_1
...
...
x_n
y_n

Část této matice, ve které jsou uloženy pozice VB, je v podstatě odlehčenou kopií databáze EKF (viz kap. 5.2), která udržuje informace o VB zanesených do systému EKF. Proto pamatujme na to, že jakákoliv změna v databázi EKF se musí zaznamenat do této matice stavu a naopak.

Matice Kalmanova zesílení: K

Kalmanovo zesílení nám určuje, jak moc věříme senzorickému měření, přesněji tedy, jak moc budeme chtít upravit predikovaný stav o hodnoty zjištěné měřením okolí robota. Například pokud zjistíme, že inovace (rozdíl mezi odhadovanou polohou robota z odometrie a odhadovanou polohou z měření jeho okolí) je 5cm, tak aktualizace polohy robota závisí právě na hodnotě Kalmanova zesílení, které je vypočítáno z kovariancí polohy robota a pozicí VB uložených v matici P. Aktualizace může tedy být třeba o 4cm, pokud měření věříme hodně, naopak, pokud spíše věříme odometrii robota, pak bude aktualizace třeba jen o 2cm. Jde tedy o nalezení kompromisu mezi daty z odometrie a daty ze senzorického měření. Matice K, jejíž struktura je znázorněna napravo, má 2 sloupce a 3+n*2 řádků, kde n je počet VB se kterýma EKF pracuje. Řádky postupně za sebou odpovídají řádkům v matici X. V prvním

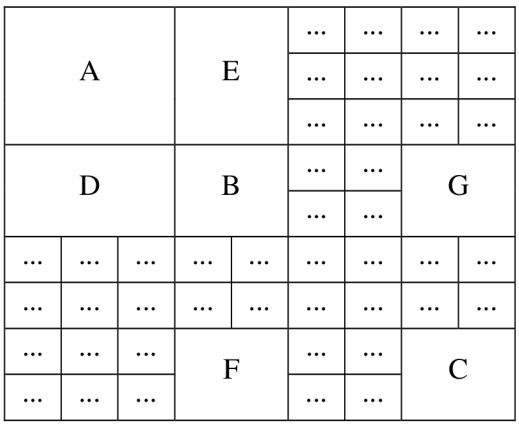
x_v	x_s
y_v	y_s
θ_v	θ_s
$x_{1,v}$	$x_{1,s}$
$y_{1,v}$	$y_{1,s}$
...	...
...	...
$x_{n,v}$	$x_{n,s}$
$y_{n,v}$	$y_{n,s}$

[2] Jacobiho matice je matice parciálních derivací.

sloupci matice K je Kalmanovo zesílení určující jak moc bude upravena odpovídající hodnota v matici X v závislosti na vzdálenosti a druhý sloupec určuje Kalmanovo zesílení pro odpovídající hodnotu v matici X v závislosti na natočení/směru.

Matice kovariancí: P

Tato matice je centrem EKF, protože určuje kovariance, tedy neurčitost jednotlivých elementů v systému. Když to rozepíšu přesněji, tak určuje kovarianci polohy robota, kovarianci pozice VB, kovarianci mezi polohou robota a pozicí VB, a nakonec kovarianci mezi VB. Jak je znázorněno napravo, můžeme si matici P rozdělit do buněk, které si teď popíšeme. Buňka A je 3x3 matice obsahující kovarianci polohy robota. Buňka B obsahuje kovarianci prvního VB, je rozměrů 2x2, protože oproti poloze robota neobsahuje natočení θ. Buňka C je potom kovariance posledního VB a má také velikost 2x2. Buňka E je kovariance prvního VB a polohy robota, přičemž buňka D je kovariance polohy robota a prvního VB a rovná se transponované submatici E. Buňka G je posledního VB a prvního VB a buňka F je zase kovariance prvního VB a posledního VB a dá se spočítat jako transponovaná submatice G. Z tohoto popisu je tedy zřejmé, že matice P je budována velmi systematicky. V počátečním stavu matice P obsahuje pouze buňku A, tedy kovarianci polohy robota. Je nutné nastavit tuto počáteční kovarianci jako nenulovou a to i v případě, že počáteční poloze robota pevně věříme.



Jacobiho matice modelu měření: H

Jacobiho matice H je matice parciálních derivací funkce pro zjištění pozice význačného bodu, respektive jeho vzdálenosti a směru do aktuální polohy robota. Tuto funkci ale nemůžeme přímo aplikovat na kovariance, a proto je převedena na Jacobiho matici. Nejprve si tedy představíme funkce pro výpočet vzdálenosti a směru VB od polohy robota.

$$h = \begin{bmatrix} \text{vzdálenost} \\ \text{směr} \end{bmatrix} = \begin{bmatrix} \sqrt{(x_{vb} - x_r)^2 + (y_{vb} - y_r)^2} \\ \tan^{-1}\left(\frac{(y_{vb} - y_r)}{(x_{vb} - x_r)}\right) - \theta_r \end{bmatrix} \quad (6.3)$$

kde x_{vb} a y_{vb} jsou souřadnice VB a x_r, y_r a θ_r jsou souřadnice a natočení robota. Výsledek této operace h nám tedy udává odhadovanou vzdálenost a směr VB od robota.

Jacobiho matice H potom vypadá takto:

$$\begin{bmatrix} \frac{x_r - x_{vb}}{vzdálenost} & \frac{y_r - y_{vb}}{vzdálenost} & 0 \\ \frac{y_{vb} - y_r}{vzdálenost^2} & \frac{x_{vb} - x_r}{vzdálenost^2} & 1 \end{bmatrix} \tag{6.4}$$

kde první řádek závislost změny vzdálenosti VB od robota na změně polohy robota (x_r, y_r a θ_r). Podobně druhý řádek značí závislost změny směru VB od robota na změně jeho polohy. Abychom mohli matici H použít v EKF pro SLAM, je nutné ji trochu poupravit, aby zapadala do všech rovnic. Úprava je jednoduchá, rozšíření matice H pro všechny VB v databázi, do těchto polí ale vložíme 0, protože Jacobiho matice H je počítána vždy pro jeden VB. Rozšířená matice H je odvozená takto:

$$\begin{bmatrix} A & B & C \\ D & E & F \end{bmatrix} = \begin{bmatrix} \frac{x_r - x_{vb}}{vzdálenost} & \frac{y_r - y_{vb}}{vzdálenost} & 0 \\ \frac{y_{vb} - y_r}{vzdálenost^2} & \frac{x_{vb} - x_r}{vzdálenost^2} & 1 \end{bmatrix} \tag{6.5}$$

$$H = \begin{matrix} x_r & y_r & \theta_r & x_{vb1} & y_{vb1} & x_{vbi} & y_{vbi} & x_{vbn} & y_{vbn} \end{matrix} \\ \begin{bmatrix} A & B & C & \dots & \dots & -A & -B & \dots & \dots \\ D & E & F & \dots & \dots & -C & -D & \dots & \dots \end{bmatrix} \tag{6.6}$$

kde první řádek H je pouze informativní a x_r, y_r, θ_r odpovídá poloze robota, dále x_{vb1},y_{vb1} odpovídá prvnímu VB, x_{vbi},y_{vbi} právě zpracovávanému VB a x_{vbn},y_{vbn} je poslednímu VB. Nastaveny jsou pouze pole odpovídající poloze robota a zpracovávaném VB, zbytek je 0. Pro VB jsou započítány pouze dva sloupce (pro x a y), protože VB nemá žádné natočení.

Jacobiho matice predikce polohy: A

Jacobiho matice A je matice parciálních derivací funkce predikce nového polohy, která je počítána z polohy poslední polohy a přírůstku z odometrie. Na základě vzorců z odometrie predikovanou polohu vypočteme takto:

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} \Delta d * \cos\theta \\ \Delta d * \sin\theta \\ \Delta\theta \end{bmatrix} \tag{6.7}$$

kde $\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}$ je poloha robota v čase t, Δd je ujetá vzdálenost robota a $\Delta\theta$ je změna natočení robota v posledním kroku. Po derivaci tedy dostaneme Jacobiho matici A:

$$A = \begin{bmatrix} 1 & 0 & -\Delta d * \sin\theta \\ 0 & 1 & \Delta d * \cos\theta \\ 0 & 0 & 1 \end{bmatrix} \tag{6.8}$$

Protože použitý robot má jednoduché diferenciální řízení, můžeme predikci stavu a tím i Jacobiho matici A upravit do následující podoby:

$$\Delta x = \Delta d * \cos\theta \tag{6.9}$$

$$\Delta y = \Delta d * \sin\theta \tag{6.10}$$

$$A = \begin{bmatrix} 1 & 0 & -\Delta y \\ 0 & 1 & \Delta x \\ 0 & 0 & 1 \end{bmatrix} \tag{6.11}$$

kde Δx a Δy je posun robota na osách x a y.

Specifické Jacobiho matice pro SLAM: J_{xr} a J_z

Tyto matice použijeme při přidávání nových význačných bodů do systému. Přidávání pozice nových VB do systému je závislé na aktuální vypočítané poloze robota a kovariance této polohy. J_{xr} jakožto Jacobiho matice výpočtu pozice VB je stejná jako Jacobiho matice predikce polohy A, jen neobsahuje poslední řádek, protože VB nemá natočení, pouze souřadnice x a y. J_z je také Jacobiho matice výpočtu pozice VB, tentokrát ale ve vztahu k vzdálenosti a směru od robota k přidávanému VB. Tyto matice jsou tedy následující:

$$J_{xr} = \begin{bmatrix} 1 & 0 & -\Delta y \\ 0 & 1 & \Delta x \end{bmatrix} \tag{6.12}$$

$$J_z = \begin{bmatrix} \cos(\theta + \Delta\theta) & -\Delta t * \sin(\theta + \Delta\theta) \\ \sin(\theta + \Delta\theta) & \Delta t * \cos(\theta + \Delta\theta) \end{bmatrix} \tag{6.13}$$

kde Δx a Δy je posun robota na osách x a y, dále θ je natočení robota, $\Delta\theta$ je směr od robota k VB a Δt je ujetá vzdálenost robota.

Matice šumu procesu a měření: Q a R

Pohyb robota je vykonáván s určitým šumem, který je Gaussova rozdělení. Tímto šumem je postihnuta každá složka polohy robota. Matice Q zahrnuje všechny tři složky dohromady a může být spočítána v závislosti na pohybu robota následovně:

$$Q = \begin{bmatrix} c\Delta x^2 & c\Delta x\Delta y & c\Delta x\Delta t \\ c\Delta y\Delta x & c\Delta y^2 & c\Delta y\Delta t \\ c\Delta t\Delta x & c\Delta t\Delta y & c\Delta\theta^2 \end{bmatrix} \tag{6.14}$$

kde c je konstanta stanovené podle toho, jak moc je odometrie robotu přesná a $\Delta x, \Delta y$ a Δt jsou přírůstky v x, y souřadnicích pozice robota a jeho natočení.

Měřící zařízení je také zatíženo určitým šumem s Gaussovým rozdělením. Jako reprezentace této skutečnosti je v EKF matice R:

$$R = \begin{bmatrix} c * v & 0 \\ 0 & d * s \end{bmatrix} \tag{6.15}$$

kde c a d jsou konstanty nastavené dle používaného senzoru, dále v a s je vzdálenost a směr naměřeného bodu. Chyba měření vzdálenosti bývá udávána v procentech, proto například pro laserový senzor s chybou 1% nastavíme c = 0.01, kdežto chyba měření směru je konstantní pro interval měření, proto pro chybu 1° můžeme provézt toto přiřazení $d * s = 1$. Upozorním na použití stejných jednotek v celém systému.

6.2 Aplikace EKF do SLAM

V této podkapitole plynule navážu na předchozí podkapitolu, kde jsem popsal vnitřní stavbu EKF používaného pro SLAM. Rozeberu zde v jednotlivých krocích postup výpočtu nového stavu systému (polohy robota a VB v databázi). Bude tento postup aplikace a zrekapitulování věcí již dříve v této práci popsaných, budu často odkazovat na předchozí kapitoly a rovnice. Veškeré akce popsané v této podkapitole jsou již uzpůsobeny této konkrétní práci, ve které pracuji se záznamem robota s diferenciálním řízením, který je vybaven laserovým dálkoměrem (viz kap. 4)

Krok 1: predikce nového stavu z odometrie mobilního robota

V tomto kroku provedeme predikci nového stavu EKF využitím dat získaných z odometrie robota. Nejdříve vypočteme novou polohu robota (první 3 řádky matice X), kterou získáme aplikací rovnice pro výpočet polohy robota (viz rovnice (3.2), respektive (6.7)) .

Dále musíme v každé nové iteraci EKF aktualizovat Jacobiho matici predikce stavu A, což provedeme aplikací rovnice (6.11). Také musíme vypočítat matici Q určující šum dat získaných z odometrie, což bychom provedli rovnicí (6.14). Já mám ale zjednodušenou práci, protože v záznamu jízdy robota je tato matice již součástí odometrických dat.

Nyní provedeme výpočet kovariance polohy robota, kterou obsahuje kovarianční matice P ve své submatici A (viz kap. 6.1). To provedeme tímto výpočtem:

$$P^{rr} = A \cdot P^{rr} \cdot A + Q \tag{6.16}$$

kde P^{rr} je submatice A, čili levé horní 3x3 submatice, a znaménko $\cdot$ značí operaci součin matic. Společně s aktualizací kovariace polohy robota musím aktualizovat i kovariance ve vztahu poloha robota - význačný bod, čemuž odpovídá celý blok prvních tří řádků matice P, kromě prvních tří sloupců, které jsme aktualizovali jakožto kovarianci polohy robota. Po tomto výpočtu také aktualizujeme blok matice P, který odpovídá kovarianci pozice význačného bodu a polohy robota, čemuž odpovídají první tři sloupce matice P, kromě prvních tří řádků, které byly aktualizovány v předchozím výpočtu. Tato aktualizace vypadá takto:

$$P^{ri} = A \cdot P^{ri} \tag{6.17}$$

$$P^{ir} = \left(P^{ri}\right)^{T} \tag{6.18}$$

Krok 2: korekce stavu pomocí porovnání aktuálního měření okolí robota s modelem světa

Odhadnutý stav v kroku 1 neodpovídá přesně skutečné poloze robota kvůli nepřesnosti odometrie, kterou jsem probral v kapitole 3. Pro zpřesnění odhadované polohy využijeme aktuálního měření laserového dálkoměru (viz kap. 2.2). Z tohoto měření extrahujeme význačné body a provedeme asociaci s databází EKF, jak bylo pospáno v kapitole 5. Výsledkem asociace význačných bodů je databáze asociovaných, kde jsou uloženy všechny význačné body aktuálního měření, které odpovídají jednomu z význačných bodů v databázi EKF. Z těchto význačných bodů spočítáme odchylku odhadované polohy robota zjištěné z odometrie (v kroku 1) od odhadované polohy robota na základě jeho senzorického měření okolí. Tuto odchylku poté využijeme pro aktualizaci stavu EKF, tedy polohy robota i pozicí význačných bodů v databázi EKF. Společně s tím aktualizujeme matici kovariancí P. Následující postup je aplikován pro každý asociovaný význačný bod.

Nejprve vypočteme odhadovanou relativní pozici význačného bodu od aktuální odhadované polohy robota. Pomocí rovnice (6.3) získáme odhadovanou vzdálenost a směr význačného bodu od robota. Tu bychom chtěli porovnat s naměřenou vzdáleností a směrem, ale k tomu se dostaneme později. Ještě předtím musíme spočítat Jacobiho matici H predikce pozice význačného bodu pomocí rovnice (6.4) a následně rovnicemi (6.5) a (6.6) převedeme matici H do rozšířeného tvaru pro použití ve SLAM. Výpočet Gaussova šumu měření R provedeme zkonkterizovanou rovnicí (6.15), která pro použitý laserový dálkoměr vypadá takto:

$$R = \begin{bmatrix} 0.01 * v & 0 \\ 0 & \frac{1}{180}\pi \end{bmatrix} \tag{6.19}$$

kde $0.01 * v$ značí chybu měření vzdálenosti 1% a $\frac{1}{180}\pi$ představuje chybu měření směru 1° převedenou na radiány, protože v celém systému pracuji s radiány. Chyba směru samozřejmě s narůstajícím úhlem zůstává konstantní.

Dále spočítáme inovaci kovariance S a následně Kalmanovo zesílení, tedy matici K. Inovace kovariance S jsem již zmínil při asociaci význačných bodů (viz kap. 5.2), jedná se o oblast nejistoty odhadované pozice význačného bodu. Pro výpočet maticí S a K využijeme následující rovnice:

$$S = H \cdot P \cdot H^{T} + R \tag{6.20}$$

$$K = P \cdot H^{T} \cdot S^{-1} \tag{6.21}$$

Připomenu, že matice K obsahuje míru korekce polohy robota a pozice význačných bodů v závislosti na aktuálním měření okolí robota. Tuto korekci matice stavu X provedeme nepřímo tak, že budeme sčítat dílčí výsledky odpovídající jednotlivým asociovaným význačným bodům a tuto sumu dílčích korekcí aplikujeme na konci tohoto kroku. Sčítání dílčích korekcí:

$$v = z - h \tag{6.22}$$

$$sumKv = K \cdot v \tag{6.23}$$

kde inovace v (také použita již při asociaci význačných bodů v kap. 5.2) je odchylka ve vzdálenosti a směru naměřené pozice význačného bodu a jeho odhadu, přičemž odhadovaná relativní pozice význačného bodu definována vzdáleností a směrem od robota h je popsána rovnicí (6.3) a skutečná naměřená vzdálenost a směr význačného bodu od robota z je počítána stejně s rozdílem, že je použita aktuální naměřená pozice význačného bodu.

Také si musím uchovávat dílčí aktualizace kovarianční matice P, které jsou definovány takto:

$$sumKH = K \cdot H \tag{6.24}$$

Na konci kroku 2 aplikujeme sumu dílčích korekcí stavu, jako aktualizaci matice stavu X. Současně aktualizujeme matici kovariancí P pomocí sumy dílčích aktualizací:

$$X = X + sumKv \tag{6.25}$$

$$P = (I - sumKH) \cdot P \tag{6.26}$$

Krok 3: aktualizace modelu světa o nově detekované význačné body

V tomto kroku přidáme do systému nové význačné body, které by měly napomoct přesnější lokalizaci robota v dalších krocích. Tento krok jsem rozdělil do dvou částí z důvodu popsaných v kapitole 5 (problém extrakce vhodných význačných bodů). První část bude přidání nových význačných bodů z databáze čekatelů do databáze EKF a část druhá bude pročištění databáze čekatelů a přidání nových význačných bodů z aktuálního měření do databáze čekatelů.

Nejprve tedy přidáme nové význačné body do databáze EKF. Tyto body získáme z databáze čekatelů, ze které vybereme body, které již byly viděny N-krát, a vložíme je do databáze EKF a současně uložíme jejich absolutní pozici v modelu světa do stavové matice X.

$$X = \begin{bmatrix} X \\ x_{vb} \\ y_{vb} \end{bmatrix} \tag{6.27}$$

Je také nutné rozšířit kovarianční matici P o nové dva sloupce a řádky, které naplníme následujícím způsobem. Připomeňme, že kovarianční matice P je čtvercová matice velikosti

(3+n*2)x(3+n*2). Pro jednoduchost vysvětlení výpočtu nových buněk si na následující stránce prostudujme schéma matice P rozšířené o buňky nového význačného bodu, které jsou podbarveny světle modře. Začneme výpočtem kovariance samotného význačného bodu. Ta je v podstatě závislá na aktuální kovarianci polohy robota a je počítána následovně:

$$D = J_{xr} \cdot A \cdot (J_{xr})^T + J_z \cdot R \cdot (J_z)^T \tag{6.28}$$

kde J_{xr} a J_z jsou specifické Jacobiho matice pro SLAM (vysvětleny v kap. 6.1) a D, A značí submatice matice P, které jsou názorně ukázány na následujícím obrázku.

<table>
<tr><td>A</td><td colspan="6">B</td><td>E</td></tr>
<tr><td rowspan="6">C</td><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td><td rowspan="6">H</td></tr>
<tr><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td></tr>
<tr><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td></tr>
<tr><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td></tr>
<tr><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td></tr>
<tr><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td></tr>
<tr><td>F</td><td colspan="6">G</td><td>D</td></tr>
</table>

Dále musíme spočítat kovarianci ve vztahu poloha robota a pozice význačného bodu značenou jako submatici E, z čehož také odvodíme kovarianci pozice význačného bodu a polohy robota značenou jako submatici F. Obě nové submatice E a F jsou opět závislé na aktuální kovarianci polohy robota.:

$$E = A \cdot (J_{xr})^T \tag{6.29}$$

$$F = E^T \tag{6.30}$$

A jako poslední zaneseme do kovarianční matice hodnotu kovariance nového význačného bodu k těm, které již jsou v databázi. Tento výpočet je také závislý na aktuální kovarianci polohy robota, respektive kovarianci mezi jednotlivými význačnými body a polohou robota. Zbylé buňky G a H vypočteme takto:

$$G = J_{xr} \cdot B \tag{6.31}$$

$$H = B^T \tag{6.32}$$

Tímto jsme dokončili první část tohoto kroku a části druhé odstraníme z databáze čekatelů právě ty význačné body, které nebyly v posledním měření detekovány a naopak přidáme do ní význačné body, které byly detekovány nově (nebyly asociovány).

Takto jsme dokončili jeden kompletní krok SLAM procesu a můžeme pokračovat ve zpracovávání záznamu jízdy robota.

7 Implementace a výsledky testování

V této kapitole stručně popíšu implementaci aplikace *RoboMap*, která zastřešuje implementaci navržené metody SLAM založené na Kalmanově filtraci. Nutno upozornit, že grafické uživatelské rozhraní není předmětem této práce, proto jsem se nezabýval jeho řešením. Aplikace sice disponuje grafickým uživatelským rozhraním, ale opravdu jenom základní formou nutnou pro spuštění zpracování záznamu.

7.1 Použité prostředky

Aplikace RoboMap je implementována v jazyce C++ s využitím multiplatformního toolkitu Qt a multiplatformního toolkitu MRPT v0.9.3 (The Mobile Robot Programming Toolkit (6)), což je open source C++ knihovna pro programování aplikací nad mobilním robotem. Tato knihovna implementuje algoritmy pro SLAM, počítačové vidění, plánování cest, atp. Já jsem ale MRPT použil jen jako podpůrný toolkit pro mou aplikaci, díky kterého mohu zpracovávat záznam jízdy robota ve formátu RAWLOG (viz kap. 2.3), dále z MRPT využívám třídu CLandmark pro uchování význačných bodů, také jsem využil třídy grafického rozhraní, díky které mohu zobrazovat průběh jízdy robota a jeho senzorického měření, případně aplikaci metody detekce přímek pomocí RANSAC v aktuálním měření a zobrazování význačných bodů. V aplikaci RoboMap jsem v lineární algebře využil open source knihovny Eigen, která je také součástí MRPT. Pro vnitřní reprezentaci světa (mapu) jsem využil tříd CSimplePointsMap a COccupancyGridMap2D, která odpovídá vytvořené mapě v uživatelsky přijatelném formátu.

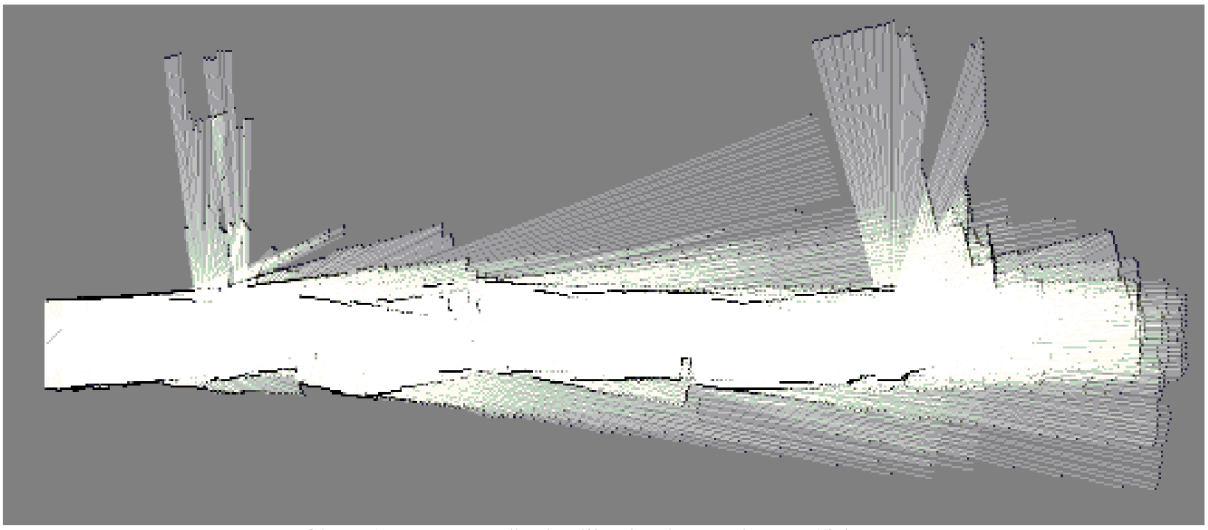
7.2 Vlastní implementace

Jádrem aplikace je objekt třídy slam_process, který řídí celý běh SLAM nad zpracovávaným záznamem jízdy robota. Tato třída obsahuje objekty jednotlivých částí SLAM, jako jsou objekt třídy landmark_worker, který implementuje práci nad extrakcí a asociaci význačných bodů, dále objekt třídy EKF_localization, který implementuje Extended Kalman Filter navržený pro SLAM na základě senzorického vybavení robota. Je zde také implementováno načítání a zpracování záznamu jízdy robota ve formátu RAWLOG, které zpracovává třída použitá z MRPT pojmenovaná CRawlog. Kromě toho, že objekt třídy slam_process zpracovává a řídí SLAM, tak také obsluhuje průběžné zobrazení aktuálního měření robota, v něm zobrazené detekované přímky pomocí RANSAC a význačné body detekované v aktuálním měření, dále řídí průběžné ukládání měření robota do mřížky obsazenosti, která je výsledkem celého procesu SLAM (viz kap. 7.3).

Aplikaci jsem implementoval tak, abych mohl řídit míru důvěryhodnosti odometrie a senzorického měření. Jsou to parametry q a r, které přímo ovlivňují míru důvěryhodnosti odometrie Q a důvěryhodnost měření R. Defaultně jsou tyto hodnoty nastaveny na $q = 100$ a $r = 0.01$

7.3 Výsledky testování

V této kapitole představím výsledky testování aplikace. Předvedu zde výsledek aplikace bez použití EKF, poté s použitím EKF.



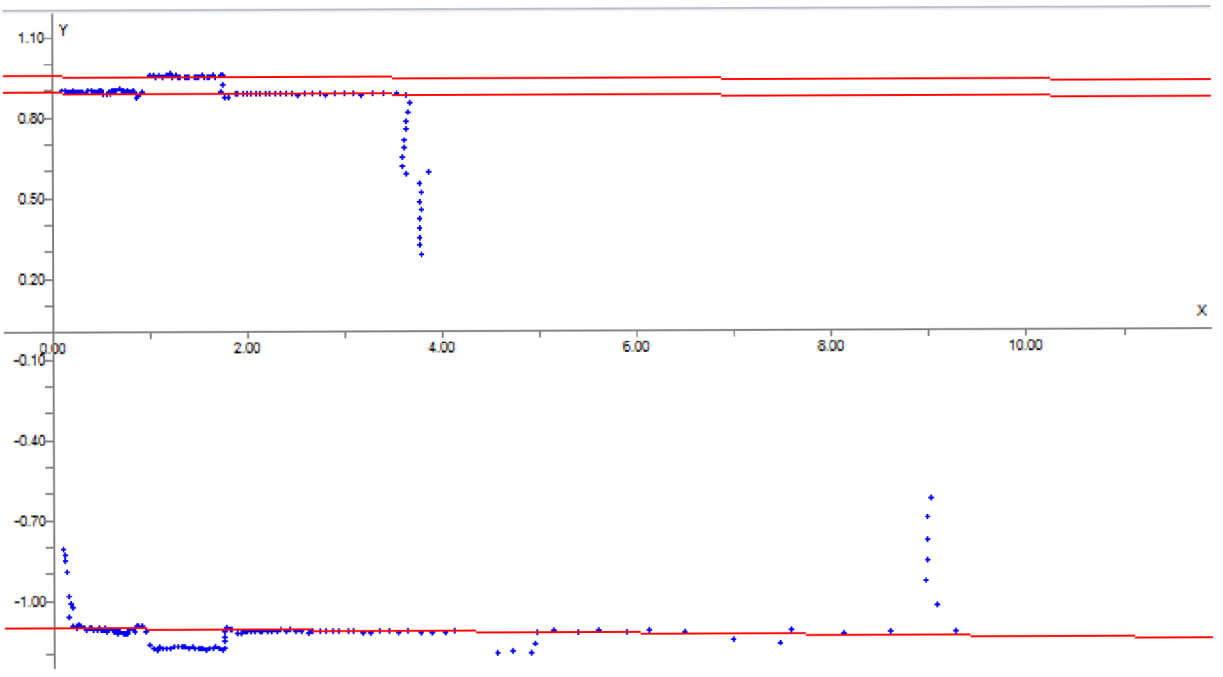
Obr. 7.1: Mapa vytvořená aplikací RoboMap bez použití EKF.

Na Obr. 7.1, je mapa vytvořená aplikací RoboMap, bez aplikace EKF, tedy pouze z odometrie, bez korekce polohy robota. Vidíme zde, že řešení problému lokalizace a mapování pouze pomocí odometrie je nevyhovující. Celá mapa je rozházená, také můžeme pozorovat rozptyl natočení robota (rozptyl paprsků ve střední a pravé části mapy).

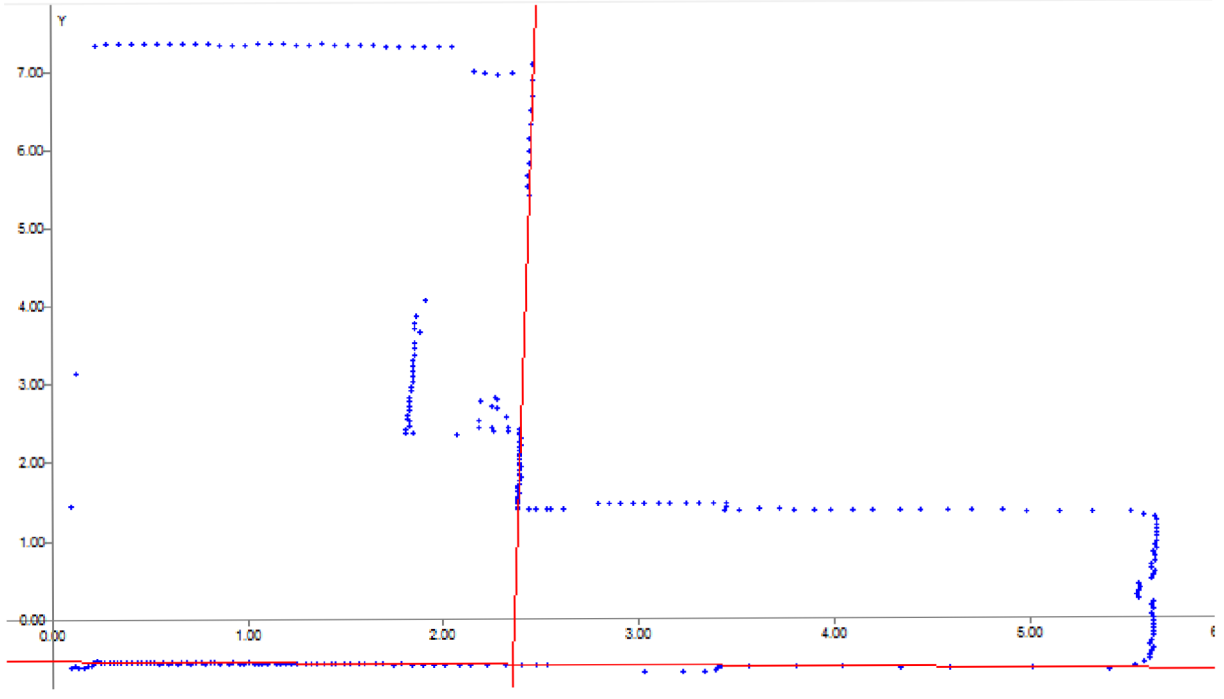
Obr. 7.1: Mapa vytvořená aplikací RoboMap s použitím EKF.

Při použití SLAM založené na EKF je na Obr. 7.2 vidět znatelný rozdíl v ustálení jízdy robota a následně jeho měření zanesených do mapy. Jsou zde již znatelně poznat hranice chodby, dokonce v druhé třetině chodby vidíme i odchylku zdi, která pravděpodobně odpovídá otevřeným dveřím směrem do chodby. Okolo hranic chodby skoro nepřečnívají měřící paprsky, až na konci, kdy robot pravděpodobně začal prudce zatáčet. Tento výsledek je velmi uspokojující.

V následujících obrázcích uvedu ilustrativní výsledky detekce přímek metodou RANSAC, kterou jsem v této práci implementoval. Obr. 7.3 znázorňuje úspěšnou detekci tří přímek, kde dvě odpovídají paralelním zdem chodby a jedna odpovídá zavřeným dveřím ve výklenku. Obr. 7.4 ukazuje detekci dvou přímek, přičemž jsou detekovány zdi na sebe skoro kolmé.



Obr. 7.3: Úspěšná detekce tří přímek provedená metodou RANSAC



Obr. 7.4: Úspěšná detekce dvou přímek provedená metodou RANSAC

8 Závěr

V této bakalářské práci jsem nastínil architekturu mobilního robota, přičemž jsem se zaměřil na motorický a senzorický systém modelu robota, jehož záznam jízdy jsem zpracovával. Popsal jsem kinematický model robota s kolovým podvozkem s diferenciálním řízením, také jsem se zmínil o proximitních senzorech. Potom jsem uvedl problematiku odometrie včetně definice systematických a nesystematických chyb, kterýma je zatížena. Obecně jsem představil princip SLAM procesu a následně jsem popsal návrh SLAM založeného na EKF s ohledem na senzorický systém již dříve popsaného mobilního robota. Detailně jsem představil metodu detekce, extrakce a asociace význačných bodů z prostředí vnitřních prostor. Společně s tím jsem uvedl a implementovat metodu RANSAC pro detekci přímek.

Navrhl jsem konkrétní podobu procesu SLAM založenou na EKF, kterou jsem implementoval a otestoval v režimu post-processing ze záznamu jízdy mobilního robota. Uvedl jsem výsledky tohoto testu, ve kterém je vidět znatelné zlepšení estimace polohy mobilního robota. Aplikaci jsem implementoval v jazyce C++ s využitím multiplatformního toolkitu Qt a volně šířitelné knihovny MRPT.

Tato bakalářská práce pojednávala o aplikaci metody SLAM založené na EKF pro mobilní roboty vybavené základním laserovým senzorem, kteří se pohybují ve vnitřních prostorách. Je to jen zlomek problematiky komplexně autonomního vozidla, ale jako odrazový můstek je to dobrý začátek. Pokračování v této práci by mohlo obsahovat zahrnutí více senzorů, pro přesnější korekci aktuální polohy robota, nebo pro přidání měření v dalším rozměru a následné tvorby 3D mapy. Jinou možnost poskytuje aplikování vytvořené aplikace v on-line režimu.

9 Bibliografie

1. *ASIMO by Honda.* [Online] http://asimo.honda.com/.

2. *Darpa Grand Challenge.com.* [Online] [Citace: 17. Květen 2011.] http://www.darpagrandchallenge.com/.

3. **Kulich, Miroslav.** Lokalizace a tvorba modelu prostředí v inteligentní robotice. *Disertační práce.* [pdf]. Praha : ČVUT FEL, 2003. Dostupné na: <http://labe.felk.cvut.cz/~kulich/diserkulich.pdf>.

4. **Bjelka, Jan.** Návrh a realizace metody mapování okolí pro mobilní robot. *Diplomová práce.* [pdf]. Brno : FSI VUT v Brně, 2007.

5. **Winkler, Zbyněk.** Odometrie. *Robotika.cz.* [Online] 5. Prosinec 2005. [Citace: 16. Květen 2011.] http://robotika.cz/guide/odometry/cs.

6. *The Mobile Robot Programming Toolkit.* [Online] http://mrpt.org/.

7. **Thrun, Sebastian, Burgard, Wolfram a Fox, Dieter.** *Propabilistic Robotics.* místo neznámé : MIT Press, 2000. ISBN 0-262-20162-3.

8. **Riisgaard, Søren a Blas, Morten Rufus.** SLAM for Dummies (A Tutorial Approach to Simultaneous Localization and Mapping). [pdf]. Dostupný na: <http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam_blas_repo.pdf>.

9. **Svoboda, Tomáš.** RANSAC - RANdom SAmple Consensus. [pdf]. Praha : Czech Technical University in Prague, Center for Machine Perception, 2008. Dostupný na: <http://cmp.felk.cvut.cz/cmp/courses/Y33ROV/Y33ROV_ZS20082009/Lectures/RANSAC/ransac.pdf>.

10. RANSAC. *Wikipedia, the free encyklopedia.* [Online] 23. Duben 2011. [Citace: 16. Květen 2011.] http://en.wikipedia.org/wiki/RANSAC.

11. **Mařík, Robert.** Metoda nejmenších čtverců. [pdf]. 2006. Dostupný na: <http://user.mendelu.cz/marik/prez/mnc-cz.pdf>.

12. Extended Kalman Filter. *Wikipedia.* [Online] 4. Duben 2011. [Citace: 16. Květen 2011.] http://en.wikipedia.org/wiki/Extended_Kalman_filter.

13. **Negenborn, Rudy.** Robot Localization and Kalman Filters (On finding your position in a noisy world). *Thesis.* [pdf]. 3. Září 2003.

Seznam příloh

Příloha A. DVD obsahující elektronickou verzi technické zprávy (pdf, docx, doc), spustitelnou verzi programu a zdrojové soubory