Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE

Ondřej Luks

Simultánní lokalizace a mapování v reálném prostředí

Katedra softwarového inženýrství
Vedoucí diplomové práce: *Mgr. Zbyněk Winkler*
Studijní program: Informatika*, obor Teoretická informatika,*

Rád bych poděkoval vedoucímu diplomové práce Mgr. Zbyňku Winklerovi za pomoc a dohled při tvorbě této práce. Zároveň bych rád poděkoval i všem ostatním kteří mě při jejím vzniku radili a podporovali. Zejména pak svým rodičům za jejich velikou trpělivost a podporu po celou dobu mých studií.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 14.prosince 2007

Ondřej Luks

Abstrakt

Název práce: Simultánní lokalizace a mapování v reálném prostředí
Autor: Ondřej Luks
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí diplomové práce: Mgr Zbyněk Winkler
e-mail vedoucího: Zbynek.Winkler@mff.cuni.cz
Abstrakt: Práce se zabývá definicí a řešením problému pohybu autonomního vozidla neznámým prostředím. Přináší podrobný rozbor řešení simultánní lokalizace a mapování pomocí rozšířeného Kalmanova filtru. Také ukazuje další jednodušších řešení prostřednictvím vymezování oblastí a za použití Monte Carlo metod.
Klíčová slova: Robotika, SLAM, BOR filtr, MCL, Monte Carlo metody, lokalizace, mapování

Title: Simultaneous Localization and Mapping of Real World Environments
Author: Ondřej Luks
Department: Department of Software Engineering
Supervisor: Mgr. Zbyněk Winkler
Supervisor's e-mail address: Zbynek.Winkler@mff.cuni.cz
Abstract: This thesis concentrates on definition and solution of simultaneous localization and mapping using extended Kalman filter. It also shows use of subomptimal methods based on bounded region filtering and Monte Carlo methods.
Keywords: Robotic, SLAM, BOR filter, localization, mapping

Obsah

1 Úvod....1
1.1 Definice problému....2
1.2 Možnosti přístupu....2
2 Kalmanův filtr....4
2.1 Model děje....4
2.2 Model pozorování....4
2.3 Odhadovaní....5
2.4 Lineární případ....6
2.5 Nelineární případ....9
2.6 Konzistence Kalmanova filtru....11
3 Matematický model....12
3.1 Stavový vektor....12
3.2 Model robota a čidel....12
3.3 Modely s diskrétním časem....13
4 Relativní pozorování a Kalmanův filtr....15
4.1 Lokalizace ve známém prostředí....15
4.2 Vytváření mapy....16
4.3 Simultánní lokalizace a vytváření mapy....17
4.3.1 Rozšířený stav....18
4.3.2 Korelace....21
4.3.3 Vznik korelací....24
4.3.4 Vývoj mapy....29
4.3.5 Relativní vzdálenosti....29
4.4 Shrnutí....30
5 Suboptimální algoritmy....32
5.1 Úvod do suboptimálních algoritmů....32
5.2 BOR filtr....33
5.2.1 BOR filtr a SLAM....33
5.2.2 Implementace BOR filtru....34
5.2.3 Definice odhadů....34
5.2.4 Predikce stavu robota....34
5.2.5 Pozorování....35
5.2.6 Update pozice objektu....36
5.2.7 Update pozice robota....38
5.2.8 Update orientace robota....38
5.2.9 Výběr pozorovaného objektu....38
5.2.10 Funkce BOR filtru....39
5.3 Monte Carlo lokalizace....41
5.3.1 Úvod do MCL metod....41
5.3.2 Průběh MCL algoritmu....41
5.3.3 MCL a SLAM....42
5.3.4 Stabilizace objektů....43
6 Konstrukce....44
6.1 Robot....44
6.2 Řídící elektronika....44
6.3 Enkodéry....46
6.4 Napájení....47
6.5 Čidla....48

7 Software.....50
7.1 Řídící deska.....50
7.2 Čidlo.....50
7.3 PC.....50
8 Výsledky.....52
9 Závěr.....55

Značení

x_r	x-ová souřadnice robota
y_r	y-ová souřadnice robota
Φ_r	orientace robota
X_r	stavový vektor robota
$\hat{x}_r$	odhad stavového vektoru robota
$\tilde{x}_r$	chyba odhadu stavového vektoru robota
P_{rr}	kovarianční matice odhadu stavu robota
x_i	x-ová souřadnice objektu číslo i
y_i	y-ová souřadnice objektu číslo i
P_{ij}	j-tý rozměr objektu číslo i
p_i	stavový vektor objektu číslo i
$\hat{p}_i$	odhad stavového vektoru objektu číslo i
$\tilde{p}_i$	chyba odhadu stavového vektoru objekt číslo i
P_{ij}	kovarianční matice odhadu stavového vektoru objektu i

1 Úvod

Robotika se snaží vyvíjet autonomní stroje – roboty, schopné plnit určité úkoly. Robot však svoje prostředí rozpoznává pouze pomocí omezeného počtu senzorů. Tyto senzory mají vždy omezený rozsah a omezenou přesnost. Pro mobilního robota je nejdůležitější informací jeho poloha. V ideálním případě dostává robot absolutní informaci o své poloze s nulovou chybou. Ve skutečnosti je ale velmi málo případů kdy dokážeme robotům zprostředkovat absolutní informaci a chyba je vždy nenulová. Příkladem absolutního čidla je třeba GPS přijímač. Ten dává informaci o pozici, která je v rámci dosahu satelitů absolutní. Ne všude je však tento systém k dispozici a ačkoliv se jeho přesnost uvádí do jednoho metru, jeho chyba může být v řádu deseti metrů. U absolutních senzorů se maximální velikost chyby s časem nemění, takže pokud robot používá správný model tohoto senzoru, může informaci z něj stále využívat se stejnou vahou. Pokud ale používáme relativní senzory, tak přestože jejich chyba může být výrazně menší, dochází k její kumulaci a pokud nemáme podle čeho ji korigovat, přeroste časem přes libovolnou mez. Pokud například robot zpracovává pouze odometrii ze svých kol, tak i odchylka v radu tisíciny procenta během otáčení vede k chybě při výpočtu natočení. To se projeví ve všech dalších výpočtech a i kdyby byla ostatní měření bez chyby, odhad pozice se stále více vzdaluje od skutečné pozice robota.

Pokud má být robot použitelný na plnění úkolu v reálném prostředí, je potřeba, aby se dokázal dlouhodobě úspěšně lokalizovat. Pro krátkodobou přesnou lokalizaci stačí relativní měření, pro udržení dlouhodobé lokalizace je potřeba tuto lokalizaci korigovat pomocí absolutních měření. Tato měření nemusí nutně přímo určovat úplnou pozici robota, ale přinášejí absolutní informaci podle které může robot korigovat chybu vzniklou akumulací relativních měření. Absolutní měření vznikají jako pozorování známých pevných objektů v prostředí robota. Objektem může být cokoliv co je robot schopen rozpoznávat pomocí svých senzorů. Například zeď změřena laserovým scannerem, nebo magnetický pól země změřený digitálním kompasem. Problém ale je s identifikací objektu. U magnetického pólu je to celkem jednoduché, ten je na zemi pouze jeden, takže až na pár výjimek kdy je kompas ovlivněn jiným magnetem, nebo odstíněn, můžeme předpokládat že kompas pozoruje úhel k magnetickému pólu země. U měření vzdálenosti ke zdi to už není tak jednoduché, pokud ale robot má k dispozici mapu svého prostředí a jeho chyba v lokalizaci není příliš velká, může určit o kterou zeď se jedná. Dalším možným přístupem je použít umělé majáčky, z jejichž pozorování dokážeme jednoznačně určit o který se jedná. Příkladem tohoto principu je například použití odrazivých značek s čárovým kódem, nebo již zmiňovaný systém GPS. Umísťování umělých značek může být dost náročné, obzvláště pokud robot se má robot pohybovat ve větším prostředí.

Proto se robotika zabývá algoritmy simultánní lokalizace a mapování (SLAM Simultaneous Localization and Mapping), kde robot vstupuje do předem neznámého prostředí a lokalizuje se pouze na základě

mapy, kterou si sám zároveň vytváří.

1.1 Definice problému

Simultánní lokalizace a mapování je postup, kdy se robot pohybuje v předem neznámém prostředí a lokalizuje se pouze pomocí svých senzorů prostřednictvím mapy prostředí. Tuto mapu si zároveň sám ze svých měření vytváří. Výstupem těchto algoritmů nemusí nutně být úplná mapa prostředí, ta může vzniknout jako vedlejší produkt. Cílem těchto algoritmů je udržet pokud možno přesnou pozici robota v neznámém prostředí.

Náročnost tohoto problému vzniká ze vzájemné propojenosti dvou podúkolů – lokalizace a mapování. Robot vlivem chyb měření zná svoji pozici jenom s určitou přesností. Z této pozice s chybou provádí měření okolních objektů v prostředí, toto měření v sobě opět skrývá nenulovou chybu. Takže robot dokáže pozici objektu ve své mapě určit jenom nepřesně, na základě této mapy se ovšem musí lokalizovat. Neboli zde vzniká smyčka, kde robot potřebuje mít přesnou mapu, aby mohl určit svoji přesnou pozici a zároveň potřebuje znát svoji přesnou pozici, aby mohl vytvořit přesnou mapu.

1.2 Možnosti přístupu

Všechny metody předpokládají, že robot dokáže ve svém prostředí detekovat objekty. K detekci objektů je několik možných přístupů. Rozdíl je, jestli se robot snaží určit objekt jednoznačně, nebo pouze nalezne objekt bez jakékoliv další informace. Jednoznačné, nebo alespoň částečné určení objektu přináší do výpočtu výrazně více informace. Díky tomu mohou metody založené na tomto principy identifikace objektů fungovat jednodušeji, nebo s méně objekty. Jednoznačná identifikace objektu se provádí buď na umělých majácích (nemusíme předem znát jejich polohu) viz například [1], nebo jako detekce určitých specifických vlastností objektu. Například pokud detekujeme rohy v laserových scannech, může se tyto rohy pokusit klasifikovat podle jejich úhlu. Je i několik metod převzatých ze zpracování obrazu pro hledání jednoznačných objektů v obraze. Důležitou vlastnosti vybraných objektů ale musí být invariance na otočení, změnu osvětlení a další vlastnosti, tak aby nedocházelo k špatné klasifikaci objektu. V této práci budeme používat jenom obecné objekty bez klasifikace. Detailnější rozbor výběru objektu je například na [2].

Speciální metodou, která mezi SLAM problémy zapádá jenom okrajově, je metoda Active vision [3]. Tato metoda používá aktivní kameru, kterou si vybere pevný objekt a během pohybu robota tento objekt stále aktivně sleduje. Stejný přístup lze aplikovat i při použití jiných senzorů. Aktivním sledováním objektu je zaručena jeho stálá a jednoznačná identifikace. Pokud se objekt dostane mimo dohled kamery, je pro sledování vybrán další objekt.

Nejrozšířenějším postupem pro řešení SLAM problému je aplikace Kalmanova filtru. Kalmanův filtr používá model chování robota a model

měření k vytvoření odhadu pozice robota a mapy. Pro SLAM problém se pak používá rozšířený Kalmanův filtr aplikovaný na stav robota zároveň se stavem všech objektů v mapě. Další možnou metodou je přímý přístup podle geometrických vlastností, kdy robot bere v potaz svoji chybu a chybu měření, snaží se geometricky z nových měření odhadnout co nejmenší oblast, ve které se objekt musí nacházet s ohledem na maximální možnou chybu. Příkladem tohoto přístupu je BOR filtr [4]. Poslední zde ukázanou metodou je pravděpodobnostní metoda založena na takzvaných Monte Carlo metodách, kdy se pro odhad pozice používá vážená množina vzorků simulující možné konfigurace robota.

Cílem této práce je podrobně prozkoumat problém simultánní lokalizace a mapování. Prostřednictvím důkladného rozboru řešení pomocí Kalmanova filtru ukážeme, v čem je tento úkol složitý a proč ho nelze rozdělit na jednodušší části. Ukážeme si jak funguje rozšířený Kalmanův a ukážeme že skutečně poskytuje optimální řešení problému. Poté ukážeme dvě suboptimální metody, které respektují vazby při simultánní lokalizaci a mapování a díky toho dosahují dobrých výsledků při menší výpočetní náročnosti než rozšířený Kalmanův filtr.

Následující text má toto členění: v kapitole 2 popíšeme jak funguje Kalmanův filtr. Ve třetí kapitole si pak vytvoříme formální matematický model pro robota a měření. Ve čtvrté kapitole potom po krocích ukážeme jak aplikovat Kalmanův filtr na řešení našeho problému. Pátá kapitola ukáže dva příklady suboptimálních algoritmů pro řešení SLAM problému. V šesté lehce popíšeme konstrukci robota použitého pro ověření těchto algoritmů. V sedmé kapitole je ukázána struktura jeho ovládacího softwaru. V osmé kapitole potom ukážeme dosažené výsledky. V poslední kapitole shrneme přínos práce a naznačíme možné navazující kroky.

2 Kalmanův filtr

Kalmanův filtr je nejběžněji používaný nastroj pro odhadovaní průběhu děje při navigaci a lokalizaci.

2.1 Model děje

Pro definici Kalmanova filtru je nejdříve potřeba formalizovat model děje. Model popisuje vývoj veličiny x v čase t. Pro spojitý případ ho definujme takto:

$$\dot{x}(t)=f^c[x(t),u(t),v(t),t]$$

kde t je čas, u(t) je řídící vstup a v(t) je šum. Šum také zahrnuje všechny jevy, které nejsou explicitně vyjádřeny v modelu.

Protože výpočty provádíme v krocích, je třeba definici upravit pro diskrétní případ. V diskrétním případě uvažujeme čas pouze v krocích t_0, t_1, .. Pro zjednodušení zápisu budeme t_k zapisovat přímo jako k:

$$x(k+1)=f[x(k),u(k),v(k),k]$$

Šum je náhodná veličina s Gaussiánským rozložením, nulovou střední hodnotou a kovariancí Q(k), $v(k)=N(0,Q(k))$. Nulovou střední hodnotu můžeme předpokládat, protože jiná než nulová by znamenala chybně vytvořený model. Tento model musí jít upravit tak, aby šum měl nulovou střední hodnotu [4].

2.2 Model pozorování

Pozorování stavu x(k) probíhá podle modelu pozorování:

$$z(k)=h[x(k),w(k),k] \quad ,$$

kde w(k) je šum pozorování, který funguje stejně jako šum děje. Zahrnuje všechny vlivy, které nejsou explicitně zahrnuty v modelu. Má nulovou střední hodnotu a Gaussianské rozložení s variancí R(k)

$$w(k)=N(0,R(k))$$

Předpoklad nulové střední hodnoty je opět podložen použitím vhodného modelu pozorování, který postihuje všechny deterministické vlastnosti pozorování. Šum pozorování vzniká obdobně jako šum děje jako kombinace mnoha nezávislých zdrojů šumu. Předpokládejme že šum je v každém okamžiku nezávislý na stavu systému a na předešlém šumu, také předpokládejme nezávislost šumu pozorování a šumu děje:

$$E[w(i)x^T(j)]=0\,\forall i,j$$
$$E[w(i)w^T(j)]=R(k)\delta_{i,j}$$
$$E[w(i)v^T(j)]=0\,\forall i,j$$

2.3 Odhadovaní

Kalmanův filtr odhaduje stav za pomocí modelu děje a modelu pozorování. Odhad skutečného stavu x(i) za předpokladu posloupnosti pozorování Z_j = {z(1),...,z(j)} označíme jako $\hat{x}$(i|j). Kalmanův filtr hledá odhad který minimalizuje střední kvadratickou odchylku v odhadu a je tedy roven očekávané hodnotě za předpokladu posloupnosti pozorování viz [5],

$$\hat{x}(i|j)=E[x|Z^j]$$

Odchylku odhadu $\hat{x}$(i|j) od skutečného stavu x(i) označíme jako $\tilde{x}$(i|j). Kovariancí $\hat{x}$(i|j) označíme jako P(i|j),

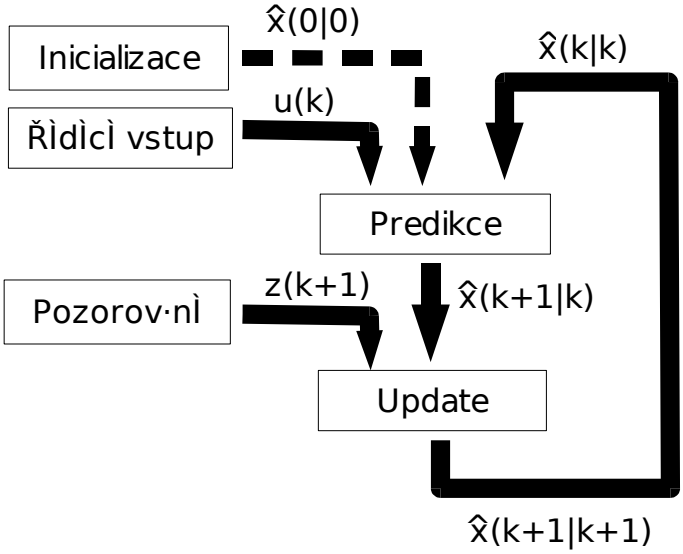
$$\tilde{x}(i|j) \simeq x(i)-\hat{x}(i|j),$$
$$P(i|j) = E[\tilde{x}(i|j)\tilde{x}^T(i|j)]$$

Kalmanův filtr počítá odhad $\hat{x}$(i|j) a zároveň počítá příslušnou kovariancí P(i|j). Tyto rovnice nemusí vždy platit, například kvůli špatné implementaci Kalmanova filtru.

Definice: Říkáme že odhad $\hat{x}$(i|j) je **konzistentní** *s kovarianční maticí P(i|j) tehdy a jen tehdy když* $P(i|j)-E[\tilde{x}(i|j)\tilde{x}^T(i|j)]$ *je pozitivně semidefinitní (viz A1). V ostatních případech říkáme že je* ***ne*konzistentní** [4]

Nyní se zaměříme na průběh výpočtu Kalmanova filtru. Chceme formulovat odhad $\hat{x}$ stavu x v čase t_{k+1}, pouze na základě odhadu z času t_k a pozorování v čase t_{k+1}.

Průběh výpočtu si můžeme rozložit na dvě části, jak je vidět na obrázku 2.1. První část je predikce, která vytvoří odhad stavu v kroku k +1 pouze na základě odhadu z kroku k. V druhé části – updatu připočítáme k odhadu stavu k+1 pozorování v kroku k+1. Tento postup rekurzivně opakujeme.



Obrázek 2.1: Kalmanův filtr

V další kapitole předpokládáme linearitu modelů pozorování a děje, což vede k jednodušší predikční a updatovací části. Toto omezení odstraníme později v kapitole 2.5

2.4 Lineární případ

V lineárním případě jsou modely děje a pozorování definovány následovně:

$$x(k+1) = F(k)x(k)+G(k)u(k)+v(k) \tag{2.5}$$

$$z(k) = H(k)x(k)+w(k) \tag{2.6}$$

F(k) je matice přechodu a G(k) mapuje ovládací vstup do stavového prostoru. Matice H(k) popisuje pozorování. Předpoklady na šum zůstávají stejné jako v kapitole 2.1 a 2.2

Odhad v kroku k+1 můžeme spočítat pomocí rovnice 2.5 za předpokladu prvních k pozorování

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k)+G(k)u(k)$$

Díky linearitě modelu můžeme predikci kovarianční matice vyjádřit jako:

$$\begin{aligned}\tilde{x}(k+1|k) &= x(k+1)-\hat{x}(k+1|k)\\ &=F(k)x(k)+G(k)u(k)+v(k)-[F(k)\hat{x}(k|k)+G(k)u(k)]\\ &=F(k)\tilde{x}(k|k)+v(k)\end{aligned}$$

$$\begin{aligned} P(k+1|k) &= E[\tilde{x}(k+1|k)\tilde{x}^T(k+1|k)] \\ &= E[F(k)\tilde{x}(k|k)\tilde{x}^T(k|k)F^T(k)]+E[v(k)v^T(k)] \\ &\quad +E[F(k)\tilde{x}(k|k)v^T(k)]+E[v(k)\tilde{x}^T(k|k)F^T(k)] \\ &= F(k)P(k|k)F^T(k)+Q(k) \end{aligned}$$

$\tilde{x}$(k|k) a v(k) předpokládáme nekorelované. Predikovaný stav umožňuje odhad pozorování v kroku k+1. Definujeme inovační vektor jako rozdíl mezi skutečným pozorováním v kroku k+1 a pozorováním predikovaným.

$$\begin{aligned} \hat{z}(k+1|k) &= E[z(k+1)|Z^k] \\ &= E[H(k+1)x(k+1)|Z^k] \\ &= H(k+1)\hat{x}(k+1|k) \end{aligned} \tag{2.7}$$

$$\begin{aligned} v(k+1) &= z(k+1)-\hat{z}(k+1|k) \\ &= z(k+1)-H(k+1)\hat{x}(k+1|k) \end{aligned}$$

Změna odhadu je rovna váženému součtu pozorování a predikce

$$\hat{x}(k+1|k+1)=\hat{x}(k+1|k)+W(k+1)v(k+1) \tag{2.8}$$

Kde W(k) je Kalmanova váhová matice, která určuje vliv inovace na změnu odhadu. Chybu odhadu můžeme vyjádřit jako:

$$\begin{aligned} \tilde{x}(k+1|k+1) &= x(k+1)-\hat{x}(k+1|k+1) \\ &= x(k+1)-[\hat{x}(k+1|k)+W(k+1)v(k+1)] \\ &= \tilde{x}(k+1|k)-W(k+1)v(k+1) \end{aligned}$$

$$\begin{aligned} P(k+1|k+1) &= E[\tilde{x}(k+1|k+1)\tilde{x}^T(k+1|k+1)] \\ &= E\left[[\tilde{x}(k+1|k)-W(k+1)v(k+1)][\tilde{x}^T(k+1|k)-W(k+1)v(k+1)]^T\right] \\ &= E[\tilde{x}(k+1|k)\tilde{x}^T(k+1|k)]+E[W(k+1)v(k+1)v^T(k+1)W^T(k+1)] \\ &\quad -E[\tilde{x}(k+1|k)v^T(k+1)W^T(k+1)]-E[W(k+1)v(k+1)\tilde{x}^T(k+1|k)] \\ &= P(k+1|k)+W(k+1)S_{vv}(k+1|k)W^T(k+1) \\ &\quad -S_{xv}(k+1|k)W^T(k+1)-W(k+1)S_{vx}(k+1|k) \end{aligned} \tag{2.9}$$

$$S_{vv}(k+1|k) = E[v(k+1)v^T(k+1)] \tag{2.10}$$

$$\begin{aligned} S_{xv}(k+1|k) &= E[\tilde{x}(k+1|k)v^T(k+1)] \\ &= E[v(k+1)\tilde{x}^T(k+1|k)]^T \\ &= S_{xv}^T(k+1|k) \end{aligned} \tag{2.11}$$

kde S_{vv}(k+1|k) je kovariance inovace a S_{xv}(k+1|k) je kovariance chyby predikce a inovace. Váhová matice je taková, aby minimalizovala střední kvadratickou odchylku $E\left[\tilde{x}^T(k+1|k+1)\tilde{x}(k+1|k+1)\right]$ což se rovná definici P(k+1|k+1).

W(k+1) se tedy rovná [4]:

$$W(k+1) = S_{xv}(k+1|k)S_{vv}^{-1}(k+1|k) \tag{2.12}$$

Kalmanův filtr počítá optimální změnu podle metody nejmenších čtverců pro odhad stavu založený na kovarianci chyby odhadu, inovaci a kovarianci inovace

Rovnice pro výpočet změny stavu a kovariance tedy jsou:

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k)W(k+1)v(k+1) \tag{2.13}$$

$$P(k+1|k+1) = P(k+1|k)-W(k+1)S_{vv}(k+1|k)W^T(k+1) \tag{2.14}$$

Z rovnic 2.8 – 2.14 vyplývá že pozorování ovlivňuje Kalmanův filtr více prostřednictvím přírůstku a kovariance přírůstku, než přímo.

V lineárním případě tedy můžeme změnu zapsat jako:

$$\begin{aligned} v(k+1) &= z(k+1)-\hat{z}(k+1|k) \\ &= H(k+1)x(k+1)+w(k+1)-H(k+1)\hat{x}(k+1|k) \\ &= H(k+1)\tilde{x}(k+1|k)+w(k+1) \end{aligned} \tag{2.15}$$

$$\begin{aligned} S_{vv}(k+1|k) &= E[v(k+1)v^T(k+1)] \\ &= H(k+1)(k)P(k+1|k)H^T(k+1)+R(k+1) \end{aligned}$$

$$\begin{aligned} S_{xv}(k+1|k) &= E[\tilde{x}(k+1|k)v^T(k+1)] \\ &= E\left[\tilde{x}(k+1|k)[H(k+1)\tilde{x}(k+1|k)+w(k+1)]^T\right] \\ &= P(k+1|k)H^T(k+1) \end{aligned}$$

$\tilde{x}$(k+1|k) a w(k+1) předpokládáme nekorelované.

Přírůstek v(k+1)v rovnici 2.15 má speciální tvar. Je závislý pouze na dvou proměnných – šumu pozorování w(k+1) a chybě odhadu $\tilde{x}$(k+1|k). Tyto dvě neznámé jsou ale nekorelované, E[$\tilde{x}$(k+1|k)w^T(k+1)]=0.

Definice: Pozorování které vede k přírůstku menšímu než chyba odhadu a šum pozorování budeme nazývat COIN pozorováním. (Constrained Innovation) [4]

Důležitým výsledkem COIN pozorování je zjištění, že potřebujeme pouze dvě kovariance pro počítání změny: kovarianci odhadu P(k+1|k) a kovarianci šumu pozorování R(k+1).

Pro lineární případ tedy máme:

Kalmanův filtr

Lineární systém

$$x(k+1) = F(k)x(k)+G(k)u(k)+v(k)$$
$$z(k) = H(k)x(k)+w(k)(k)$$

Predikce

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k)+G(k)u(k)$$
$$P(k+1|k) = F(k)P(k|k)F^T(k)+Q(k)$$

Update

$$\hat{x}(k+1|k+1) = x(k+1|k)+W(k+1)v(k+1)$$
$$P(k+1|k+1) = P(k+1|k)-W(k+1)S_{vv}(k+1|k)W^T(k+1)$$

kde

$$v(k+1) = z(k+1)-H(k+1)\hat{x}(k+1|k)$$
$$S_{vv}(k+1|k) = H(k+1)P(k+1|k)H^T(k+1)+R(k+1)$$
$$W(k+1) = P(k+1|k)H^T(k+1)S_{vv}^{-1}(k+1|k)$$

2.5 Nelineární případ

V nelineárním případě si modely děje a pozorování vyjádříme jako:

$$x(k+1) = f[x(k),u(k),v(k),k] \tag{2.16}$$

$$z(k) = h[x(k),w(k),k] \tag{2.17}$$

Při odhadech nelineárních systémů se nejčastěji používá rozšířený Kalmanův filtr (dále jen EKF – Extended Kalman Filter). EKF předpokládá že modely děje a pozorování jsou lokálně lineární, šumy děje a pozorování jsou malé a že

$$\hat{x}(k|k) \approx E[x(k)|Z^k]$$

Model děje linearizujeme pomocí Taylorova rozvoje $\hat{x}$(k|k)

$$x(k+1) = f[\hat{x}(k|k),u(k),0,k]+\nabla f_x\tilde{x}(k|k)+\nabla f_v v(k)+\textit{členy vyšších řádů}$$

kde ∇f_x je Jakobián f podle x v bodě $\hat{x}$(k|k) a ∇f_v je Jakobián podle v. Za předpokladu že vyšší řády Taylorova rozvoje můžeme zanedbat a že $\tilde{x}$(k|k), v(k) jsou malé veličiny s nulovou střední hodnotou, můžeme vyjádřit následující predikční rovnice:

$$\begin{aligned}\hat{x}(k+1|k) &= E[x(k+1)|Z^k]\\ &\approx E\left[f[\hat{x}(k|k),u(k),0,k]+\nabla f_x\tilde{x}(k|k)+\nabla f_v v(k)\right]\\ &= f[\hat{x}(k|k),u(k),0,k]\end{aligned}$$

$$\begin{aligned}\tilde{x}(k+1|k) &= x(k+1)-\hat{x}(k+1|k)\\ &\approx \nabla f_x\tilde{x}(k|k)+\nabla f_v v(k)\end{aligned}$$

$$P(k+1|k) = E[\tilde{x}(k+1|k)\tilde{x}^T(k+1|k)] \approx \nabla f_x P(k|k) \nabla f_x^T + \nabla f_v Q(k) \nabla f_v^T$$

Update fázi dostaneme obdobně pomocí Taylorova rozvoje modelu pozorování $\hat{x}$(k+1|k),

$$z(k+1) = h[\hat{x}(k+1|k),0,k] + \nabla h_x \tilde{x}(k+1|k) + \nabla h_w w(k+1) + \textit{členy vyšších řádů}$$

kde ∇h_x je Jakobián h podle x v bodě $\hat{x}$(k+1|k) a ∇h_w je Jakobián podle w. Předpokládáme že

$$\hat{x}(k+1|k) \approx E[x(k+1)|Z^k] \quad ,$$

že $\tilde{x}$(k+1|k) a w(k+1) jsou malé náhodné veličiny s nulovou střední hodnotou a že vyšší členy Taylorova rozvoje můžeme zanedbat pak dostáváme:

$$\hat{z}(k+1|k) \approx E\left[h[\hat{x}(k+1|k),0,k] + \nabla h_x \tilde{x}(k+1|k) + \nabla h_w w(k+1)\right] = h[\hat{x}(k+1|k),0,k]$$

$$v(k+1) = z(k+1) - \hat{z}(k+1|k) \approx \nabla h_x \tilde{x}(k+1|k) + \nabla h_w w(k+1) \tag{2.18}$$

$$S_{vv}(k+1|k) = E[v(k+1)v^T(k+1)] \approx \nabla h_x P(k+1|k) \nabla h_x^T + \nabla h_w R(k+1) \nabla h_w^T \tag{2.19}$$

$$S_{xv}(k+1|k) = E[\tilde{x}(k+1|k)v^T(k+1)] \approx E\left[\tilde{x}(k+1|k)[\nabla h_x \tilde{x}(k+1|k) + \nabla h_w w(k+1)]^T\right] = P(k+1|k) \nabla h_x^T \tag{2.20}$$

$\tilde{x}$(k+1|k) a w(k+1) jsou nekorelované.

Rovnice 2.8 až 2.14 stále platí, takže stejně jako v lineárním případě ovlivňuje model pozorování EKF hlavně skrze přírůstek a kovarianci přírůstku.

Rovnice přírůstku 2.18 ukazuje že model pozorování vede ke COIN pozorování. Stejně jako v lineárním případě potřebujeme pouze dvě kovariance pro spočtení updatu: P(k+1|k) a R(k+1).

Pro EKF tedy máme:

Rozšířený Kalmanův filtr EKF

Nelinearní system

$$x(k+1) = f[x(k), u(k), v(k), k]$$
$$z(k) = h[x(k), w(k), k]$$

Predikce

$$\hat{x}(k+1|k) = f[\hat{x}(k|k), u(k), 0, k]$$
$$P(k+1|k) = \nabla f_x P(k|k) \nabla f_x^T + \nabla f_v Q(k) \nabla f_v^T$$

Update

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + W(k+1) v(k+1)$$
$$P(k+1|k+1) = P(k+1|k) - W(k+1) S_{vv}(k+1|k) W^T(k+1)$$

kde:

$$v(k+1) = z(k+1) - h[\hat{x}(k+1|k), 0, k]$$
$$S_{vv}(k+1|k) = \nabla h_x P(k+1|k) \nabla h_x^T + \nabla h_w R(k+1) \nabla h_w^T$$
$$W(k+1) = P(k+1|k) \nabla h_x^T S_{vv}^{-1}(k+1|k)$$

2.6 Konzistence Kalmanova filtru

Řekneme že Kalmanův filtr funguje konzistentně jestliže výsledné odhady jsou konzistentní podle definice v sekci 2.3.

N·sledující rovnice Kalmanova filtru

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + W(k+1) v(k+1)$$
$$P(k+1|k+1) = P(k+1|k) - W(k+1) S_{vv}(k+1|k) W^T(k+1)$$

d·vají konzistentní výsledek tehdy a jen tehdy pokud platí

$$W(k+1) = S_{xv}(k+1|k) S_{vv}^{-1}(k+1|k)$$

3 Matematický model

Lokalizační problém je definován jako problém určení pozice robota pohybujícího se prostředím na základě pozorování prostředí z pozice robota. Pozice robota se skládá v našem příkladu z jeho posunutí v x,y a orientace, obecně stavu viz [6].

Prostředí modelujeme jako množinu objektů $P^N=\{p_1,...,p_N\}$, kde pi je stav i-tého objektu. Objekty předpokládáme jako pevné, tedy jejich pozice vůči referenčnímu počátku souřadnic se nemění.

Senzory pozorují pozici objektů relativně vůči pozici robota. Tedy pokud známe pozici objektu, můžeme přímo odhadnout pozici robota. V kapitole 4.1 ukážeme že pozorování objektů se známou pozicí je ekvivalentní přímo pozorování robota, takže můžeme Kalmanův filtr přímo aplikovat.

Pro naše potřeby si později vybudujeme různé modely pro robota, objekty, děj a pozorování, ale teorie funguje nezávisle na určitém modelu.

3.1 Stavový vektor

Definujeme si stavový vektor $x_r(t)$ robota v čase t jako ($x_r(t)$, $y_r(t)$, $\phi_r(t)$), tedy posun ve dvou osách a otočení vůči počátku souřadnic. Stavový vektor objektu $p_i(t)$ je definován jako ($x_i(t)$, $y_i(t)$), tedy pouze posunutí vůči stejnému počátku jako stavový vektor robota.

$x_r(t) = [\ (x_r(t)\ \ y_r(t)\ \phi_r(t)]^T$

$p_i(t) = [\ x_i(t)\ \ y_i(t)]^T$

3.2 Model robota a čidel

Fyzikální chování robota popisuje kinematicky model f_{Xr}^c. Obecný model může záviset na čase, ale námi použitý příklad na čase nezávisí.

$$\dot{x}_r(t) = f_{x_r}^c[x_r(t), u(t), v(t), t]$$

$$\begin{bmatrix} \dot{x}_r(t) \\ \dot{y}_r(t) \\ \dot{\phi}_r(t) \end{bmatrix} = \begin{bmatrix} [V(t)+v_r(t)]\cos(\phi_r(t)+[\gamma(t)+v_\gamma(t)]) \\ [V(t)+v_r(t)]\sin(\phi_r(t)+[\gamma(t)+v_\gamma(t)]) \\ \dfrac{[V(t)+v_r(t)]}{B}\sin(\gamma(t)+v_\gamma) \end{bmatrix}$$

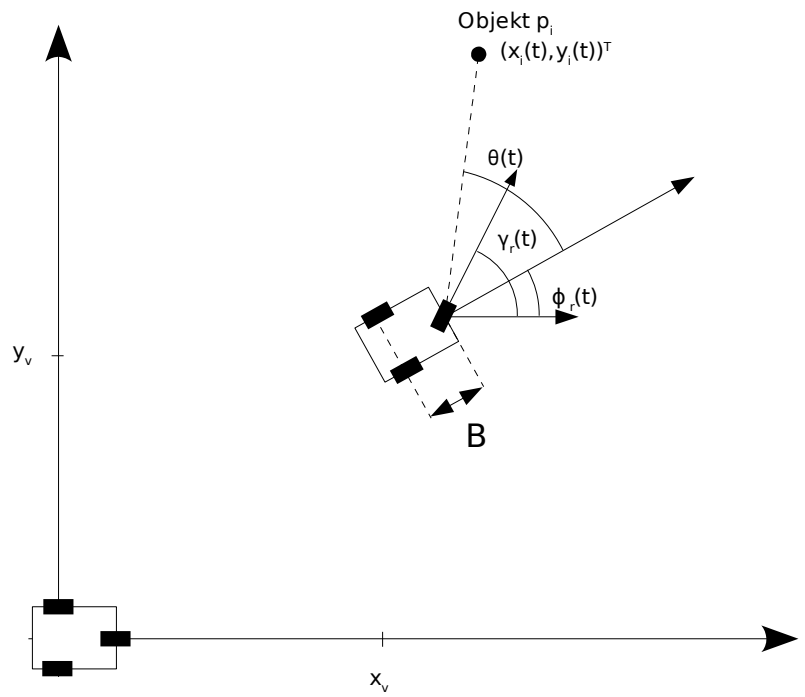
kde B je délka základny robota $u(t) = \begin{bmatrix} V(t) & \gamma(t) \end{bmatrix}^T$ je ovládací vstup v čase t. Skládá se z rychlosti V(t) a úhlu zatočení $\gamma(t)$. Šum $v(t) = \begin{bmatrix} v_r(t) & v_\gamma(t) \end{bmatrix}^T$ je přičítán k ovládacímu vstupu.

Předpokládáme že robot je vybaveno čidlem které provádí měření vzdálenosti $r_i(t)$ pod úhlem $\theta_i(t)$ od robota k objektu p_i. Stejně jako u modelu pohybu robota, může model pozorování záviset na času, ale v našem případě tomu tak není. Pozorování objektu p_i v čase t můžeme vyjádřit jako:

$$z_{x_r p_i}(t) = h_{x_r p}[x_r(t), p_i(t), w(t), t]$$

$$\begin{bmatrix} r_i(t) \\ \theta_i(t) \end{bmatrix} = \begin{bmatrix} \sqrt{(x_r(t)-x_i(t))^2+(y_r(t)-y_i(t))^2}+w_r(t) \\ \tan^{-1}\dfrac{y_r(t)-y_i(t)}{x_r(t)-x_i(t)}-\phi_r(t)+w_\theta(t) \end{bmatrix}$$

kde $w(t) = \begin{bmatrix} w_r(t) & w_\theta(t) \end{bmatrix}^T$ je šum pozorování.



3.3 Modely s diskrétním časem

Tyto modely je pro implementaci na počítači třeba upravit do diskrétní podoby. Budeme používat kroky konstantní délky, ale ukázané postupy by fungovaly i pro kroky proměnné délky. Délku periody označíme jako T. Čas v kroku k tedy značíme jako t_k=k*T.

Diskrétní model robota f_{Xr} je

$$
\begin{aligned}
x_r(k+1) &= f_{x_r}[x_r(k), u(k), v(k), k] \\
&= \begin{bmatrix} x_r(k)+T[V(k)+v_r(k)]\cos(\phi_r(k)+[\gamma(k)+v_\gamma(k)]) \\ x_r(k)+T[V(k)+v_r(k)]\sin(\phi_r(k)+[\gamma(k)+v_\gamma(k)]) \\ \phi_r(k)+\dfrac{T[V(k)+v_r(k)]}{B}\sin(\gamma(k)+v_\gamma(k)) \end{bmatrix} \\
&= x_r(k)+T\begin{bmatrix} [V(k)+v_r(k)]\cos(\phi_r(k)+[\gamma(k)+v_\gamma(k)]) \\ [V(k)+v_r(k)]\sin(\phi_r(k)+[\gamma(k)+v_\gamma(k)]) \\ \dfrac{[V(k)+v_r(k)]}{B}\sin(\gamma(k)+v_\gamma(k)) \end{bmatrix}
\end{aligned}
\tag{2.27}
$$

Diskrétní model pozorování je:

$$
\begin{aligned}
z_{x_r p_i}(k) &= h_{x_r p}[x_r(k), p_i(k), w(k), k] \\
&= \begin{bmatrix} r_i(k)+w_r(k) \\ \theta_i(k)+w_\theta(k) \end{bmatrix} \\
&= \begin{bmatrix} \sqrt{(x_r(k)-x_i(k))^2+(y_r(k)-y_i(k))^2}+w_r(k) \\ \tan^{-1}\dfrac{y_r(k)-y_i(k)}{x_r(k)-x_i(k)}-\phi_r(k)+w_\theta(k) \end{bmatrix}
\end{aligned}
\tag{2.29}
$$

stav objektů se v čase nemění tedy: $p_i(k+1) = p_i(k) = p_i$.

V následujících kapitolách budeme potřebovat lineární modely. Model robota můžeme linearizovat pomocí Taylorova rozvoje jako:

$$
x_r(k+1) \approx f_{x_r}[\widehat{x}_r(k|k), u(k), 0, k]+\nabla(f_{x_r})_{x_r}\widetilde{x}_r(k|k)+\nabla(f_{x_r})_v v(k) \tag{2.30}
$$

kde $\nabla(f_{x_r})_{x_r}$ a $\nabla(f_{x_r})_v$ jsou Jakobiány modelu robota podle stavu robota x_r a podle šumu v. Předpokládáme že vyšší řády rozvoje jsou zanedbatelné, $\hat{x}_r(k|k) \approx E[x_r(k)|Z^k]$ a že $\tilde{x}_r(k|k)$ a v(k) jsou malé.

Podobně můžeme linearizovat model pozorování pomocí Taylorova rozvoje $(\hat{x}_r(k+1|k), \hat{p}_i(k+1|k))$,

$$
\begin{aligned}
z_{x_r p_i}(k+1) \approx\ & h_{x_r p}[\hat{x}_r(k+1|k), \hat{p}_i(k+1|k), w(k+1), k+1]+\nabla(h_{x_r p})_{x_r}\tilde{x}_r(k+1|k) \\
& +\nabla(h_{x_r p})_{p_i}\tilde{p}_i(k+1|k)+\nabla(h_{x_r p})_w w(k+1)
\end{aligned}
\tag{2.31}
$$

kde $\nabla(h_{x_r p})_{x_r}$, $\nabla(h_{x_r p})_{p_i}$ a $\nabla(h_{x_r p})_w$ jsou Jakobiány pozorování podle stavu robota x_r, stavu objektu i p_i a šumu pozorování w. Předpokládáme zanedbatelnost vyšších řádu Taylorova rozvoje. Dále předpokládáme že

$$
\begin{aligned}
\hat{x}_r(k+1|k) &\approx E[x_r(k+1)|Z^j] \\
\hat{p}_i(k+1|k) &\approx E[p_i(k+1)|Z^j]
\end{aligned}
$$

a že $\tilde{x}_r(k+1|k), \tilde{p}_i(k+1|k)$ *a* $w(k)$ jsou malé.

4 Relativní pozorování a Kalmanův filtr

Lokalizační problém je problém určení pozice robota na základě pozorování prostředí z pozice robota. V této kapitole se zaměříme právě na použití Kalmanova filtru při relativním pozorování.

Z linearizovaného modelu pozorování můžeme vyjádřit přírůstek jako:

$$\begin{aligned} v(k+1) &= z_{x_r p_i}(k+1) - \hat{z}_{x_r p_i}(k+1|k) \\ &\approx \nabla(h_{x_r p})_{x_v} \tilde{x}_r(k+1|k) + \nabla(h_{x_r p})_{p_i} \tilde{p}_i(k+1|k) \\ &\quad + \nabla(h_{x_r p})_w w(k+1) \end{aligned} \tag{2.32}$$

přírůstek tedy není vázaný na šum pozorování ani na chybu odhadu.

Pozorování objektu relativně z pozice robota není COIN pozorování (viz 2.4)

Kovariance přírůstku je:

$$S_{vv}(k+1|k) = E[v(k+1)v^T(k+1)]$$

4.1 Lokalizace ve známém prostředí

Při lokalizaci ve známém prostředí známe v každém kroku stav objektu p_i pro všechna i a stačí odhadovat stav robota.

$$\begin{aligned} \hat{p}_i(k+1|.) &= p_i(k+1) \forall i \\ \tilde{p}_i(k+1|.) &= 0 \forall i \end{aligned}$$

$$\begin{aligned} \hat{x}_r(k+1|k) &= f_{x_r}[\hat{x}_r(k|k), u(k), 0, k] \\ P_{vv}(k+1|k) &= \nabla(f_{x_r})_{x_r} P_{vv}(k|k) \nabla(f_{x_r})_{x_r}^T + \nabla(f_{x_r})_v Q(k) \nabla(f_{x_r})_v^T \end{aligned}$$

$$\begin{aligned} z_{x_r p_i}(k+1) &= h_{x_r p}[\hat{x}_r(k+1), \hat{p}_i(k+1), w(k+1), k+1] + \nabla(h_{x_r p})_{x_r} \tilde{x}_r(k+1|k) \\ &\quad + \nabla(h_{x_r p})_w w(k+1) + \text{členy vyšších řádů} \\ v(k+1) &\approx \nabla(h_{x_r p})_{x_r} \tilde{x}_r(k+1|k) + \nabla(h_{x_r p})_w w(k+1) \end{aligned} \tag{2.38}$$

Tvar přírůstku v rovnici v 2.38 je stejný jako při přímém pozorování

$$\begin{aligned} z_{x_r}(k+1) &= h_{x_r}[x_r(k+1), w(k+1), k+1] \\ &\approx h_{x_r}[\hat{x}_r(k+1|k), 0, k+1] + \nabla(h_{x_r})_{x_r} \tilde{x}_r(k+1|k) + \nabla(h_{x_r})_w w(k+1) \\ v(k+1) &\approx \nabla(h_{x_r p})_{x_r} \tilde{x}_r(k+1|k) + \nabla(h_{x_r p})_w w(k+1) \end{aligned} \tag{2.39}$$

kde $z_{x_r}(k+1)$ je přímé pozorování robota podle modelu pozorování.

Pozorování přesně známého prostředí relativně z pozice robota je COIN pozorování robota, protože přírůstek v(k+1) závisí jenom na šumu pozorování w(k+1) a chybě odhadu $\tilde{x}_r(k+1|k)$. Z toho můžeme vyjádřit rovnici updatu jako:

Lokalizace ve zn·mém prostředí

$$S_{vv}(k+1|k) = \nabla(h_{x_r p})_{x_r} P_{vv}(k+1|k)\nabla(h_{x_r p})_{x_r}^T + \nabla(h_{x_r p})_w R(k+1)\nabla(h_{x_r p})_w^T$$
$$S_{x_r v}(k+1|k) = P_{vv}(k+1|k)\nabla(h_{x_r p})_{x_r}^T$$
$$W_{x_r}(k+1) = S_{x_r v}(k+1|k)S_{vv}^{-1}(k+1|k)$$

všechny kovariance které jsou potřeba pro update jsou tady uvnitř P_{vv}(k +1|k) a R(k+1).

Takže aplikace Kalmanova filtru na lokalizaci ve známém prostředí je relativně přímočará podle výše zmíněných rovnic. Protože odhadujeme pouze stav robota a jednu kovarianční matici, není to ani příliš výpočetně náročné. Přestože model robota může být relativně složitý, tak protože není výpočet závislý na počtu objektů je lokalizace ve známém prostředí relativně jednoduchá.

4.2 Vytváření mapy

Vytváření mapy je postup kdy pomocí měření z robota vytváříme mapu prostředí. Předpokládáme že v tomto případě je pozice robota přesně známá. V praxi tento stav málokdy nastává, ale je to důležitý krok při vytváření teorie.

$$\hat{x}_v(k+1|.) = x_r(k+1)$$
$$\tilde{x}(k+1|.) = 0$$

$$\hat{p}_i(k+1|k) = \hat{p}_i(k|k)\forall i$$
$$P_{ii}(k+1|k) = P_{ii}(k|k)\forall i$$

$$z_{x_r p_i}(k+1) = h_{x_r p}[\hat{x}_r(k+1),\hat{p}_i(k+1),w(k+1),k+1]+\nabla(h_{x_r p})_{p_i}\tilde{p}_i(k+1|k)$$
$$+\nabla(h_{x_r p})_w w(k+1)+\textit{členy vyšších řádů}$$
$$v(k+1) \approx \nabla(h_{x_r p})_{p_i}\tilde{p}_i(k+1|k)+\nabla(h_{x_r p})_w w(k+1)$$

Protože přírůstek závisí jenom na chybě pozorování w(k+1) a chybě v odhadu objektu $\tilde{p}_i$(k+1|k). Přírůstek chápeme jako přírůstek vzniklý z přímého pozorování objektu p_i

$$z_{p_i}(k+1) = h_p[p_i(k+1),w(k+1),k+1]$$
$$\approx h_p[\hat{p}_i(k+1|k),0,k+1]+\nabla(h_p)_{p_i}\tilde{p}_i(k+1|k)+\nabla(h_p)_w w(k+1)$$
$$v(k+1) \approx \nabla(h_p)_{p_i}\tilde{p}_i(k+1|k)+\nabla(h_p)_w w(k+1)$$

(2.40)

kde $z_{p_i}(k+1)$ je přímé pozorování objektu p_i podle modelu pozorování h_p a $\nabla(h_p)_{p_i}$, $\nabla(h_p)_w$ jsou Jakobiány modelu pozorování podle $p_i(k+1)$ a $w(k+1)$.

Pozorování objektu p_i relativně k přesně známé pozici robota je COIN pozorování.

Vytváření mapy

$$S_{vv}(k+1|k) = \nabla(h_{x_r}p)_{p_i}P_{ii}(k+1|k)\nabla(h_{x_r}p)^T_{p_i}+\nabla(h_{x_r}p)_w R(k+1)\nabla(h_{x_r}p)^T_w$$
$$S_{p_i v}(k+1|k) = P_{ii}(k+1|k)\nabla(h_{x_r}p)^T_{p_i}$$
$$W_{p_i}(k+1) = S_{p_i v}(k+1|k)S^{-1}_{vv}(k+1|k)$$

všechny kovariance potřebné k updatu jsou uvnitř $P_{ii}(k+1|k)$ a $R(k+1)$. Pokud víme k jakému objektu dané pozorování náleží (dokážeme objekt jednoznačně a bezchybně identifikovat), pak můžeme Kalmanův filtr přímo aplikovat. Každé pozorování je pozorováním jediného objektu a je COIN. Při použití posloupnosti pozorování jednoho objektu můžeme pomocí Kalmanova filtru odhadovat příští stav tohoto objektu. Pro každý objekt tedy můžeme použít nezávislý Kalmanův filtr který zpracovává jenom pozorování daného objektu. Dostaneme tak N nezávislých filtrů pro N objektů.

Při každém pozorování probíhá právě jeden update, právě jeden Kalmanův filtr provádí výpočet, takže vytváření mapy je řádově stejně složitý problém jako lokalizace ve známém prostředí. Prostorové nároky rostou lineárně s počtem objektů, ale výpočetní složitost se s počtem objektů nemění.

4.3 Simultánní lokalizace a vytváření mapy

Pokud ale řešíme oba předešlé problémy najednou, jde o řádově náročnější problém. Protože neznáme ani přesnou pozici robota ani pozici objektů, musíme najednou odhadovat všechny pozice. Můžeme předpokládat že iniciální pozice robota je známá, protože ji můžeme definovat jako počátek souřadnic. Na simultánní lokalizaci a vytváření mapy (dále jen SLAM-Simultaneous Localisation and Map building) se můžeme dívat jako na posloupnost kroků. Na začátku máme s určitou přesností definovaný počáteční stav robota. Na základě relativního pozorování objektu můžeme odhadnou počáteční stav pozorovaného objektu. Na základě dalších pozorování tohoto objektu můžeme provést update stavu robota a tak provádět jeho lokalizaci.

Pozorování jediného objektu nestačí pro pohyb v neznámém prostředí. Je potřeba aby robot přidával počáteční stavy dalších objektů ve všech oblastech prostředí ve kterém se pohybuje. Takže SLAM musí najednou odhadovat stav robota a stavy několika objektů. Složitost problému vzniká hlavně z toho že pozorování objektu závisí jak na stavu

robota, tak na stavu objektu.

$$z_{x_r p_i}(k) = h_{x_r p}[x_r(k), p_i(k), w(k), k] \tag{2.41}$$

Takže přírůstek závisí na chybě odhadu robota, chybě odhadu objektu i na šumu pozorování.

$$\begin{aligned} v(k+1) &= z_{x_r p_i}(k+1) - \hat{z}_{x_r p_i}(k+1|k) \\ &\approx \nabla(h_{x_r p})_{x_v} \tilde{x}_r(k+1|k) + \nabla(h_{x_r p})_{p_i} \tilde{p}_i(k+1|k) + \nabla(h_{x_r p})_w w(k+1) \end{aligned}$$

Kovariance přírůstku tedy vypadá:

$$\begin{aligned} S_{vv}(k+1|k) &= E[v(k+1)v^T(k+1)] \\ &\approx \nabla(h_{x_r p})_{x_r} P_{vv}(k+1|k) \nabla(h_{x_r p})_{x_r}^T + \nabla(h_{x_r p})_{p_i} P_{ii}(k+1|k) \nabla(h_{x_r p})_{p_i}^T \\ &\quad + \nabla(h_{x_r p})_{x_r} P_{vi}(k+1|k) \nabla(h_{x_r p})_{p_i}^T + \nabla(h_{x_r p})_{p_i} P_{iv}(k+1|k) \nabla(h_{x_r p})_{x_r}^T \\ &\quad + \nabla(h_{x_r p})_w R(k+1|k) \nabla(h_{x_r p})_w^T \end{aligned}$$

Je vidět že kovariance $S_{vv}(k+1|k)$ závisí na kovarianci odhadu stavu robota $P_{rr}(k+1|k)$, kovarianci odhadu stavu objektu $P_{ii}(k+1|k)$ a kovarianci odhadu stavu robota a odhadu stavu objektu $P_{ri}(k+1|k)$.

Pokud měníme $\hat{x}_r$ musíme zároveň přepočítat i P_{ri} kterou budeme potřebovat při příštím pozorování objektu p_i. $P_{ri} = E[\tilde{x}_r \tilde{x}_r^T]$, $kde\, \tilde{x}_r = x_r - \hat{x}_r$ takže se mění při každé změně $\hat{x}_r$ To platí pro všechna i.

4.3.1 Rozšířený stav

Abychom mohli na řešení SLAM aplikovat Kalmanův filtr, je potřeba sloučit stavové vektory. Naším cílem je z dvou samostatných řešení – lokalizace a mapovaní udělat jeden problém na který půjde aplikovat Kalmanův filtr.

Stavový vektor robota tedy rozšíříme o stavové vektory všech objektů v mapě, vznikne tak rozšířený vektor $x_m(k)$, odhad stavu robota a mapy bude $\hat{x}_m(k|k)$ a všechny příslušné korelace budou v kovarianční matici $P_m(k|k)$

$$x_m(k) = [\ x_r^T(k)\ \ p_1^T(k)\ \ \dots\ \ p_N^T(k)\ ]^T \tag{4.1}$$

$$
\begin{aligned}
\hat{x}_m(k|k) &= [\ \hat{x}_r^T(k|k)\ \ \hat{p}_1^T(k|k)\ \ \ldots\ \ \hat{p}_N^T(k|k)\ ]^T \\
P_m(k|k) &= E[\tilde{x}_m(k|k)\tilde{x}_m^T(k|k)] \\
&= E\left[\begin{bmatrix}\tilde{x}_r(k|k)\\ \tilde{p}_1(k|k)\\ \vdots \\ \tilde{p}_N(k|k)\end{bmatrix}\begin{bmatrix}\tilde{x}_r(k|k)\\ \tilde{p}_1(k|k)\\ \vdots \\ \tilde{p}_N(k|k)\end{bmatrix}^T\right] \\
&= \begin{bmatrix} P_{rr}(k|k) & P_{r1}(k|k) & \cdots & P_{rN}(k|k) \\ P_{1r}(k|k) & P_{11}(k|k) & \cdots & P_{1N}(k|k) \\ \vdots & \vdots & \ddots & \vdots \\ P_{Nr}(k|k) & P_{N1}(k|k) & \cdots & P_{NN}(k|k)\end{bmatrix}
\end{aligned}
\tag{4.2}
$$

Kde

$$
\begin{aligned}
P_{rr}(k|k) &= E[\tilde{x}_r(k|k)\tilde{x}_v^T(k|k)] \\
P_{ii}(k|k) &= E[\tilde{p}_i(k|k)\tilde{p}_i^T(k|k)] \\
P_{ri}(k|k) &= E[\tilde{x}_r(k|k)\tilde{p}_i^T(k|k)] \\
P_{ij}(k|k) &= E[\tilde{p}_i(k|k)\tilde{p}_j^T(k|k)]
\end{aligned}
$$

$P_{rr}(k|k)$ je kovarianční matice odhadu stavu robota, $P_{ii}(k|k)$ pro všechna i od 1 do N jsou kovarianční matice odhadu stavu objektu. $P_{ri}(k|k) = P_{ir}^T(k|k)$ jsou korelace mezi chybou odhadu robota a chybou odhadu i-tého objektu pro všechna i od 1 do N

$P_{ij}(k|k) = P_{ji}^T(k|k)$ jsou korelace mezi chybou odhadů dvou objektů, i a j.

Rozšířený stavový vektor umožňuje Kalmanovu filtru udržovat všechny korelace mezi chybami odhadů robota i objektů.

Model chování robota f_{Xm} je odvozen od původního modelu chování robota f_{Xr} díky předpokladu že pozice objektů se během času nemění:

$$
\begin{aligned}
x_v(k+1) &= f_{x_v}[x_v(k),u(k),v(k),k] \\
&\approx f_{x_v}[\hat{x}_v(k|k),u(k),0,k]+\nabla(f_{x_v})_{x_v}\tilde{x}_v(k|k)+\nabla(f_{x_v})_v v(k) \\
p_i(k+1) &= p_i(k)\,\forall i=1,2,\ldots,N
\end{aligned}
$$

$$
\begin{aligned}
x_m(k+1) &= f_{x_m}[x_m(k),u(k),v(k),k] \\
&= \begin{bmatrix} x_v^T(k+1) & p_1^T(k+1) & \ldots & p_N^T(k+1) \end{bmatrix}^T \\
&= \begin{bmatrix} f_{x_v}[x_v(k),u(k),v(k),k] \\ p_1(k) \\ \vdots \\ p_N(k) \end{bmatrix} \\
&= \begin{bmatrix} f_{x_v}[\hat{x}_v(k|k),u(k),0,k]+\nabla(f_{x_v})_{x_v}\tilde{x}_v(k|k)+\nabla(f_{x_v})_v v(k) \\ p_1(k) \\ \vdots \\ p_N(k) \end{bmatrix}
\end{aligned}
\tag{4.3}
$$

predikce $\hat{x}_m(k+1|k)$ pak vypadá:

$$
\begin{aligned}
\hat{x}_m(k+1|k) &= E[x_m(k+1)|Z^k] \\
&\approx \begin{bmatrix} f_{x_r}[\hat{x}_r(k|k),u(k),0,k] \\ \hat{p}_1(k|k) \\ \vdots \\ \hat{p}_N(k|k) \end{bmatrix} \\
&= f_{x_m}[\hat{x}_m(k|k),u(k),0,k]
\end{aligned}
\tag{4.4}
$$

$$
\begin{aligned}
\tilde{x}_m(k+1|k) &= x_m(k+1)-\hat{x}_m(k+1|k) \\
&= \begin{bmatrix} \nabla(f_{x_r})_{x_r}\tilde{x}_r(k|k)+\nabla(f_{x_r})_v v(k) \\ \tilde{p}_1(k|k) \\ \vdots \\ \tilde{p}_N(k|k) \end{bmatrix} \\
P_m(k+1|k) &= E[\tilde{x}_m(k+1|k)\tilde{x}_m^T(k+1|k)] \\
&= \begin{bmatrix} \nabla(f_{x_r})_{x_r}P_{rr}(k|k)\nabla(f_{x_r})_{x_r}^T & \nabla(f_{x_r})_{x_r}P_{r1}(k|k) & \cdots & \nabla(f_{x_r})_{x_r}P_{rN}(k|k) \\ P_{1r}(k|k)\nabla(f_{x_r})_{x_r}^T & P_{11}(k|k) & \cdots & P_{1N}(k|k) \\ \vdots & \vdots & \ddots & \vdots \\ P_{Nr}(k|k)\nabla(f_{x_r})_{x_r}^T & P_{N1}(k|k) & \cdots & P_{NN}(k|k) \end{bmatrix} \\
&\quad + \begin{bmatrix} \nabla(f_{x_r})_v Q(k)\nabla(f_{x_r})_v^T & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \\
&= \nabla(f_{x_m})_{x_m}P_m(k|k)\nabla(f_{x_m})_{x_m}^T+\nabla(f_{x_m})_v Q(k)\nabla(f_{x_m})_v^T
\end{aligned}
\tag{4.5}
$$

kde:

$$\nabla(f_{x_m})_{x_m} = \begin{bmatrix} \nabla(f_{x_r})_{x_r} & 0 & 0 & \cdots \\ 0 & I & 0 & \cdots \\ 0 & 0 & I & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$\nabla(f_{x_m})_v = \begin{bmatrix} \nabla(f_{x_r})_v^T & 0 & \dots \end{bmatrix}^T$$

Z toho můžeme již vytvořit predikční rovnice. Musíme ještě upravit model pozorování tak, aby popisoval pozorování vždy jen jednoho objektu.

$$\begin{aligned} z_{x_m i}(k) &= z_{x_r p_i}(k) \\ h_{x_r i}[x_m(k), i, w(k), k] &= h_{x_r p}[x_r(k), p_i(k), w(k), k] \end{aligned} \tag{4.6}$$

$$\begin{aligned} \nabla(h_{x_m i})_{x_r} &= \nabla(h_{x_r p})_{x_r} \\ \nabla(h_{x_m i})_{p_i} &= \nabla(h_{x_r p})_{p_i} \\ \nabla(h_{x_m i})_{p_j} &= 0 \, pro \, j \neq i \\ \nabla(h_{x_m i})_w &= \nabla(h_{x_r p})_w \end{aligned} \tag{4.7}$$

$$\begin{aligned} \nabla(h_{x_m i})_{x_m} &= \begin{bmatrix} \nabla(h_{x_m i})_{x_r} & \nabla(h_{x_m i})_{p_1} & \dots & \nabla(h_{x_m i})_{p_N} \end{bmatrix} \\ &= \begin{bmatrix} \nabla(h_{x_r p})_{x_r} & 0 & \dots & 0 & \nabla(h_{x_r p})_{p_i} & 0 & \dots \end{bmatrix} \end{aligned} \tag{4.8}$$

pro zjednodušení zavedeme následující zápis:

$$\begin{aligned} H_i &= \nabla(h_{x_m i})_{x_m} \\ H_r &= \nabla(h_{x_m i})_{x_r} \\ H_p &= \nabla(h_{x_m i})_{p_i} \\ H_w &= \nabla(h_{x_m i})_w \end{aligned}$$

Jakobiány $\nabla(h_{x_m i})_{x_m}$ pozorování objektů p_1,p_2 a p_3 pak vypadají následovně:

$$\begin{aligned} H_1(k) &= \begin{bmatrix} H_r(k) & H_p(k) & 0 & 0 & 0 & \dots \end{bmatrix} \\ H_2(k) &= \begin{bmatrix} H_r(k) & 0 & H_p(k) & 0 & 0 & \dots \end{bmatrix} \\ H_3(k) &= \begin{bmatrix} 0 & 0 & H_r(k) & H_p(k) & 0 & \dots \end{bmatrix} \end{aligned} \tag{4.9}$$

Řešení SLAM problému pomocí Kalmanova filtru nad rozšířeným stavovým vektorem budeme označovat jak MAK filtr (MAK-map augmented Kalman) [7].

4.3.2 Korelace

V této části ukážeme že všechny prvky kovarianční matice $P_m(k|k)$ jsou důležité pro řešení SLAM problému. Rozšířený stavový vektor umožňuje aby MAK filtr udržoval korelace mezi chybami odhadů stavu robota a objektu i objektů vzájemně. Důležitost těchto korelací můžeme

ukázat na updatu následujícím po pozorování objektu p_i.

$$\hat{x}_m(k+1|k) = \begin{bmatrix} \hat{x}_r^T & \dots & \hat{p}_i^T & \dots & \hat{p}_j^T & \dots & \hat{p}_N^T \end{bmatrix}^T (k|k)$$

$$P_m(k|k) = \begin{bmatrix} P_{rr} & \dots & P_{ri} & \dots & P_{rj} & \dots & P_{rN} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ P_{ir} & \dots & P_{ii} & \dots & P_{ij} & \dots & P_{iN} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ P_{jr} & \dots & P_{ji} & \dots & P_{jj} & \dots & P_{jN} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ P_{Nr} & \dots & P_{Ni} & \dots & P_{Nj} & \dots & P_{NN} \end{bmatrix} (k|k)$$

$$H_i(k+1) = \begin{bmatrix} H_r & 0 & \dots & 0 & H_p & 0 & \dots \end{bmatrix} (k+1)$$

kovariance přírůstku tedy je:

$$\begin{aligned} S_i(k+1|k) &= H_i(k+1)P_m(k+1|k)H_i^T(k+1)+H_w(k+1)R(k+1)H_w^T(k+1) \\ &= H_r(k+1)P_{rr}(k+1|k)H_r^T(k+1)+H_p(k+1)P_{ii}(k+1|k)H_p^T(k+1) \\ &\quad +H_r(k+1)P_{ri}(k+1|k)H_p^T(k+1)+H_p(k+1)P_{ir}(k+1|k)H_r^T(k+1) \\ &\quad +H_w(k+1)R(k+1)H_w^T(k+1) \\ &= \nabla(h_{x_r p})_{x_r} P_{rr}(k+1|k)\nabla(h_{x_r p})_{x_r}^T+\nabla(h_{x_r p})_{p_i} P_{ii}(k+1|k)\nabla(h_{x_r p})_{p_i}^T \\ &\quad +\nabla(h_{x_r p})_{x_r} P_{ri}(k+1|k)\nabla(h_{x_r p})_{p_i}^T+\nabla(h_{x_r p})_{p_i} P_{ir}(k+1|k)\nabla(h_{x_r p})_{x_r}^T \\ &\quad +\nabla(h_{x_r p})_w R(k+1)\nabla(h_{x_r p})_w^T \end{aligned}$$

Je vidět že jsme došli ke stejnému výsledku jako $S_{vv}(k+1|k)$ v rovnici 2.33. To znamená že MAK filtr správně počítá kovarianci přírůstku. Protože můžeme pozorovat libovolný objekt p_1, ..., p_N, musíme udržovat všechny kovariance $P_{ri}(k+1|k)$ abychom mohli spočítat kovarianci přírůstku. Kovarianční matice která obsahuje $P_{ri}(k+1|k)$ pro i=1..N musí také obsahovat $P_{ij}(k+1|k)$ i=1..N, j=1..N. Jinak by nebyla splněna podmínka $P(k+1|k)=E[\tilde{x}(k+1|k)\tilde{x}^T(k+1|k)]$ která je nutná pro správnou funkci Kalmanova filtru.

Váhová matice MAK filtru má následující tvar:

$$
\begin{aligned}
W_i(k+1) &= \begin{bmatrix} E\left[\tilde{x}_r(k+1|k)v^T(k+1)\right] \\ \vdots \\ E\left[\tilde{p}_i(k+1|k)v^T(k+1)\right] \\ \vdots \\ E\left[\tilde{p}_j(k+1|k)v^T(k+1)\right] \\ \vdots \\ E\left[\tilde{p}_N(k+1|k)v^T(k+1)\right] \end{bmatrix} S_{rr}^{-1}(k+1|k) \\
&= E\left[\tilde{x}_m(k+1|k)v^T(k+1)\right] S_{rr}^{-1}(k+1|k) \\
&= S_{x_m r}(k+1|k) S_{rr}^{-1}(k+1|k)
\end{aligned}
$$

Víme že:

$$
S_{p_i v}(k+1|k) = P_{jr}(k+1|k)\nabla(h_{x_r p})_{x_r}^T + P_{ij}(k+1|k)\nabla(h_{x_v p})_{p_i}^T
$$

Takže dostáváme:

$$
W_i(k+1) = \begin{bmatrix} S_{x_r v}(k+1|k)S_{vv}^{-1}(k+1|k) \\ \vdots \\ S_{p_i v}(k+1|k)S_{vv}^{-1}(k+1|k) \\ \vdots \\ S_{p_j v}(k+1|k)S_{vv}^{-1}(k+1|k) \\ \vdots \\ S_{p_N v}(k+1|k)S_{vv}^{-1}(k+1|k) \end{bmatrix} \tag{4.10}
$$

Submatice $S_{x_r v}(k+1|k)S_{vv}^{-1}(k+1|k)$ *a* $S_{p_i v}(k+1|k)S_{vv}^{-1}(k+1|k)$ jsou váhové matice pro update odhadu stavu robota a odhadu stavu pozorovaného objektu což zaručuje optimalitu MAK filtru.

Submatice $S_{p_j v}(k+1|k)S_{vv}^{-1}(k+1|k)$ zase zaručují že i nepozorované objekty projdou updatem podle korelací mezi přírůstkem a chybou odhadu nepozorovaného objektu.

$$
\begin{aligned}
W_i(k+1) &= \begin{bmatrix} E\left[\tilde{x}_r(k+1|k)v^T(k+1)\right] \\ \vdots \\ E\left[\tilde{p}_i(k+1|k)v^T(k+1)\right] \\ \vdots \\ E\left[\tilde{p}_j(k+1|k)v^T(k+1)\right] \\ \vdots \\ E\left[\tilde{p}_N(k+1|k)v^T(k+1)\right] \end{bmatrix} S_{vv}^{-1}(k+1|k) \\
&= E\left[\tilde{x}_m(k+1|k)v^T(k+1)\right] S_{vv}^{-1}(k+1|k) \\
&= S_{x_m v}(k+1|k) S_{vv}^{-1}(k+1|k)
\end{aligned}
\tag{4.11}
$$

Což znamená že $W_i(k+1)$ splňuje podmínku z kapitoly 2.6 postačující pro konzistentní update $\hat{x}_m(k+1|k)$ a $P_m(k+1|k)$

$$\hat{x}_m(k+1|k+1) = \hat{x}_m(k+1|k) + W_i(k+1)v(k+1) \tag{4.12}$$

$$P_m(k+1|k+1) = P_m(k+1|k) - W_i(k+1)S_i^T(k+1|k)W_i^T(k+1) \tag{4.13}$$

Váhová matice tedy zajišťuje konzistentní a ve smyslu nejmenších čtverců optimální update odhadu stavu robota a všech objektů.

4.3.3 Vznik korelací

Tato část popisuje vznik korelací mezi chybami odhadu v MAK filtru. Ukážeme proč rovnosti v P_m z části 4.3.2 musíme předpokládat jako nenulové.

Budeme pokračovat v příkladu z 4.3.2 Protože počáteční kovariance mezi všemi odhady bereme nulové, tak počáteční stav MAK filtru bude:

$$\hat{x}_m(0|0) = \begin{bmatrix} \hat{x}_r \\ \hat{p}_1 \\ \hat{p}_2 \end{bmatrix}(0|0) \tag{4.14}$$

$$P_m(0|0) = \begin{bmatrix} P_{rr} & 0 & 0 \\ 0 & P_{11} & 0 \\ 0 & 0 & P_{22} \end{bmatrix}(0|0) \tag{4.15}$$

Uvažujme dva následující updaty. První update následuje pozorování p_1 a druhý pozorování p_2.

$$H_1(1) \triangleq [H_r \;\; H_p \;\; 0](1) \tag{4.16}$$

$$H_1(2) \triangleq [H_r \;\; 0 \;\; H_p](2) \tag{4.17}$$

Z rovnice 4.5 vyplývá že predikce $P_m(1|0)$ nemění strukturu $P_m(0|0)$ a proto můžeme $P_m(1|0)$ definovat jako $P_m(1|0) = P_m(0|0)$.

První update spočítáme jako:

$$
\begin{aligned}
S_1(1|0) &= H_1(1)P_m(1|0)H_1^T(1)+H_w(1)R(1)H_w^T(1)\\
W_1(1) &= P_m(1|0)H_1^T(1)S_1^{-1}(1|0)\\
\hat{x}_m(1|1) &= \hat{x}_m(1|0)+W_1(1)v(1)\\
P_m(1|1) &= P_m(1|0)-W_1(1)S_1(1|0)W_1^T(1)
\end{aligned}
$$

kde v(1) je změna v k=1. Váhová matice MAK filtru je:

$$
\begin{aligned}
W_1(1) &= \begin{bmatrix} S_{x_r v}S_{vv}^{-1} \\ S_{p_1 v}S_{vv}^{-1} \\ S_{p_2 v}S_{vv}^{-1} \end{bmatrix}(1|0)\\
&= \begin{bmatrix} P_{rr}H_r^T+P_{r1}H_p^T \\ P_{1r}H_v^T+P_{11}H_p^T \\ P_{2r}H_v^T+P_{21}H_p^T \end{bmatrix}(1|0)S_1^{-1}(1|0) \qquad (4.18)\\
&= \begin{bmatrix} P_{rr}H_r^T \\ P_{11}H_p^T \\ 0 \end{bmatrix}(1|0)S_1^{-1}(1|0)
\end{aligned}
$$

Update odhadu můžeme zapsat jako:

$$\hat{x}_r(1|1) = \hat{x}_r(1|0)+P_{rr}(1|0)H_r^T(1)v(1) \qquad (4.19)$$

$$\hat{p}_1(1|1) = \hat{p}_1(1|0)+P_{11}(1|0)H_p^T(1)v(1) \qquad (4.20)$$

$$\hat{p}_2(1|1) = \hat{p}_2(1|0) \qquad (4.21)$$

Protože $H_1(1)$ representuje relativní pozorování objektu p_1 z pozice robota, $H_r(1)$ a $H_p(1)$ jsou nenulové. To způsobuje korelaci mezi přírůstkem v(1) a chybou odhadu $\tilde{x}(1|0)$ a $\tilde{p}_1(1|0)$. Nenulovost $P_{rr}(1|0)H_r^T(1) = S_{x_r v}(1|0)$ způsobuje korelaci mezi $\tilde{x}(1|0)$ a v(1). Nenulovost $P_{11}(1|0)H_p^T(1) = S_{p_1 v}(1|0)$ pak způsobuje korelaci mezi $\tilde{p}_1(1|0)$ a v(1).

Pokud by se $H_1(1)$ skládalo pouze z $H_p(1)$, $H_r(1)$ by tedy bylo nulové což by odpovídalo přímému pozorování objektu p_1, přírůstek by byl korelován pouze s chybou $\tilde{p}_1(1|0)$. Pak $P_{rr}(1|0)H_r^T(1) = S_{x_r v}(1|0) = 0$. Update by se týkal pouze v(1|0) a $\hat{x}_r(1|1) = \hat{x}_r(1|0)$.

Obdobně pokud by se $H_1(1)$ skládalo pouze z $H_r(1)$, $H_p(1)$ by teď tedy bylo nulové což by odpovídalo přímému pozorování robota, přírůstek by byl korelován pouze s chybou $\tilde{x}_r(1|0)$. Tedy $P_{11}(1|0)H_p^T(1) = S_{p_1 v}(1|0) = 0$. To by vedlo k updatu pouze $\hat{x}_r(1|0)$ a $\hat{p}_1(1|1) = \hat{p}_1(1|0)$.

Update kovariancí vypadá následovně:

$$\begin{aligned} P_{rr} &= P_{rr}(1|0) - \left[W_{x_r} S_1^{-1} W_{x_r}\right](1|0) \\ &= P_{rr}(1|0) - \left[P_{rr} H_r^T S_1^{-1} H_r P_{rr}\right](1|0) \end{aligned} \tag{4.22}$$

$$\begin{aligned} P_{r1}(1|1) &= P_{r1}(1|0) - [W_{x_r} S_1^{-1} W_{p_1}](1|0) \\ &= -[P_{rr} H_r^T S_1^{-1} H_p P_{11}](1|0) \end{aligned} \tag{4.23}$$

$$\begin{aligned} P_{r2}(1|1) &= P_{r2}(1|0) - [W_{x_r} S_1^{-1} W_{p_2}](1|0) \\ &= 0 \end{aligned} \tag{4.24}$$

$$\begin{aligned} P_{11}(1|1) &= P_{11}(1|0) - [W_{p_1} S_1^{-1} W_{p_1}](1|0) \\ &= P_{11}(1|0) - [P_{11} H_p^T S_1^{-1} H_p P_{11}](1|0) \end{aligned} \tag{4.25}$$

$$\begin{aligned} P_{12}(1|1) &= P_{12}(1|0) - [W_{p_1} S_1^{-1} W_{p_2}](1|0) \\ &= 0 \end{aligned} \tag{4.26}$$

$$\begin{aligned} P_{22}(1|1) &= P_{22}(1|0) - [W_{p_2} S_1^{-1} W_{p_2}](1|0) \\ &= P_{22}(1|0) \end{aligned} \tag{4.27}$$

$$P_m(1|1) = \begin{bmatrix} P_{rr} & P_{r1} & 0 \\ P_{1r} & P_{11} & 0 \\ 0 & 0 & P_{22} \end{bmatrix}(1|1) \tag{4.28}$$

Chyby v odhadu stavu robota a v odhadu stavu objektu p_1 jsou nyní korelované díky nenulovosti $P_{r1}(1|1)$. Tato korelace vzniká protože se opíráme o pozorování měřené relativně k pozici robota.

> Update po relativním pozorování koreluje chybu v odhadu stavu robota s chybou v odhadu stavu pozorovaného objektu protože se stejné pozorování používá k updatu stavu robota i pozorovaného objektu.

Podle rovnice 4.5 vidíme že predikce $P_m(2|1)$ opět nemění strukturu $P_m(1|1)$ Takže opět můžeme zadefinovat že $P_m(2|1)=P_m(1|1)$.

Druhý update následuje po pozorování objektu p_2 takže ho spočteme jako:

$$
\begin{aligned}
S_2(2|1) &= H_2(2)P_m(2|2)H_2^T(2)+H_w(2)R(2)H_w^T(2)\\
W_2(2) &= P_m(2|1)H_2^T(2)S_2^{-1}(2|1)\\
\hat{x}_m(2|2) &= \hat{x}_m(2|2)+W_2(2)v(2)\\
P_m(2|2) &= P_m(2|1)-W_2(2)S_2(2|1)W_2^T(2)
\end{aligned}
$$

kde v(2) je přírůstek v kroku k=2. Váhová matice pro druhý update má tvar:

$$
\begin{aligned}
W_2(2) &= \begin{bmatrix} S_{x_r v}S_{vv}^{-1} \\ S_{p_1 v}S_{vv}^{-1} \\ S_{p_2 v}S_{vv}^{-1} \end{bmatrix}(2|1)\\
&= \begin{bmatrix} P_{rr}H_r^T+P_{r2}H_p^T \\ P_{1r}H_r^T+P_{12}H_p^T \\ P_{2r}H_r^T+P_{22}H_p^T \end{bmatrix}(2|1)S_2^{-1}(2|1)\\
&= \begin{bmatrix} P_{rr}H_r^T \\ P_{1r}H_r^T \\ P_{22}H_p^T \end{bmatrix}(2|1)S_2^{-1}(2|1)
\end{aligned}
\tag{4.29}
$$

Update odhadu můžeme zapsat jako:

$$\hat{x}_r(2|2) = \hat{x}_r(2|1)+P_{rr}(2|1)H_r^T(2)v(2) \tag{4.30}$$

$$\hat{p}_1(2|2) = \hat{p}_1(2|1)+P_{1r}(2|1)H_r^T(2)v(2) \tag{4.31}$$

$$\hat{p}_2(2|2) = \hat{p}_2(2|1)+P_{22}(2|1)H_p^T(2)v(2) \tag{4.32}$$

Stejně jako u prvního updatu, relativní pozorování koreluje přírůstek v(2) s chybou odhadu stavu robota $\tilde{x}_r(2|1)$ a také s chybou odhadu pozorovaného objektu $\tilde{p}_2(2|1)$. Tentokrát je ale ještě navíc korelován i s chybou $\tilde{p}_1(2|1)$ protože $\tilde{p}_1(2|1)$ je korelován s $\tilde{x}_v(2|1)$. Proto bude všechny $S_{x_r v}(2|1),\ S_{p_1 v}(2|1)\ a\ S_{p_2 v}(2|1)$ nenulové a update se bude týkat $\hat{x}_r(2|1)$, $\hat{p}_1(2|1)$ i $\hat{p}_2(2|1)$.

Rozepsaný druhý update vypadá:

$$
\begin{aligned}
P_{rr}(2|2) &= P_{rr}(2|1)-[W_{x_r}S_2^{-1}W_{x_r}](2|1)\\
&= P_{rr}(2|1)-[P_{rr}H_r^T S_2^{-1}H_r P_{rr}](2|1)
\end{aligned}
\tag{4.33}
$$

$$
\begin{aligned}
P_{r1}(2|2) &= P_{r1}(2|1)-[W_{x_r}S_2^{-1}W_{p_1}](2|1)\\
&= P_{r2}(2|1)-[P_{rr}H_r^T S_2^{-1}H_r P_{r1}](2|1)
\end{aligned}
\tag{4.34}
$$

$$
\begin{aligned}
P_{r2}(2|2) &= P_{r2}(2|1)-[W_{x_r}S_2^{-1}W_{p_2}](2|1)\\
&= -[P_{rr}H_r^T S_2^{-1}H_p P_{22}](2|1)
\end{aligned}
\tag{4.35}
$$

$$\begin{aligned} P_{11}(2|2) &= P_{11}(2|1) - [W_{p_1} S_2^{-1} W_{p_1}](2|1) \\ &= P_{11}(2|1) - [P_{1r} H_r^T S_2^{-1} H_r P_{r1}](2|1) \end{aligned} \tag{4.36}$$

$$\begin{aligned} P_{12}(2|2) &= P_{12}(1|0) - [W_{p_1} S_2^{-1} W_{p_2}](2|1) \\ &= -[P_{1r} H_r^T S_2^{-1} H_p P_{22}](2|1) \end{aligned} \tag{4.37}$$

$$\begin{aligned} P_{22}(2|2) &= P_{22}(2|1) - [W_{p_2} S_2^{-1} W_{p_2}](2|1) \\ &= P_{22}(2|1) - [P_{22} H_p^T S_2^{-1} H_p P_{22}](2|1) \end{aligned} \tag{4.38}$$

$$P_m(2|2) = \begin{bmatrix} P_{rr} & P_{r1} & P_{r2} \\ P_{1r} & P_{11} & P_{12} \\ P_{2r} & P_{21} & P_{22} \end{bmatrix}(2|2) \tag{4.39}$$

všechny prvky $P_m(2|2)$ jsou teď nenulové, což znamená že všechny chyby

Přírůstek způsobený relativním pozorováním objektu p_i je korelovaný s chybou odhadu stavu robota. Pokud je chyba odhadu stavu robota korelovaná s chybou odhadu objektu p_j, pak je přírůstek také korelován s chybou odhadu objektu p_j. To znamená že chyba odhadu stavu objektu p_i po updatu bude korelována s chybou odhadu stavu objektu p_j. To jest pokud k updatu používáme relativní pozorování, tak jsou všechny chyby odhadu korelovány.

odhadů jsou teď korelované.

V sekci 4.3.3 jsme ukázali že korelace mezi chybami odhadů musíme udržovat aby MAK filtr zaručil konzistentní update. V této kapitole jsme ukázali že korelace jsou v SLAM problému nevyhnutelné.

SLAM problém pro N objektů nelze řešit jako N samostatných problémů. Pro řešení SLAMU je nutné použít filtr na všechny stavy najednou v podobě složeného stavu x_m. S tím je také vázána kovarianční matice P_m. Během každého updatu je nutné přepočítat každý člen x_m i P_m aby všechny odhady zůstaly konzistentní se svými kovariancemi.

Velikost stavového vektoru x_m roste lineárně s počtem objektů. Z

toho vyplývá že velikost kovarianční matice roste s druhou mocninou.

V prostředí s N objekty je pro řešení SLAM problému potřeba alespoň o(N^2) prostoru a výpočetní složitost je také alespoň o(N^2)

4.3.4 Vývoj mapy

Zaměříme se na vývoj jednotlivých prvků mapy za běhu MAK filtru. Rovnice pro udpdate kovarianční matice Kalmanova filtru vypadá:

$$P(k+1|k+1) = P(k+1|k) - W(k+1)S(k+1|k)W^T(k+1) \qquad (4.40)$$

Z teorie matic vyplývá že determinant positivně semidefinitní matice se přičtením jiné positivně definitní matice nesníží. Protože matice P(k+1|1) a W(k+1)S(k+1|k)W^T(k+1) jsou PSD tak je jasné že

$$det(P(k+1|k+1)) \leq det(P(k+1|k))$$

To znamená že determinant kovarianční matice se při updatu nemůže zvětšit.

Protože každá submatice PSD matice je také PSD, platí tato nerovnost také pro všechny submatice P(k+1|k+1). Tedy i pro P_{vv} , P_{ii} a všechny jejich kombinace tvaru $\begin{bmatrix} P_{nn} & P_{nm} \\ P_{mn} & P_{mm} \end{bmatrix}$ kde n,m=v, 1, 2, .., N. Protože predikce neovlivňuje kovarianční matici, můžeme det(P(k+1|k)) nahradit det(P(k|K)) viz [8].

Determinanty kovariančních matic jsou v čase nerostoucí funkce.

4.3.5 Relativní vzdálenosti

Když vezmeme dva objekty p_i a p_j, zapíšeme jejich relativní vzdálenost jako vektor Δp_{ij}

$$\Delta p_{ij} = \begin{bmatrix} p_{j_1} - p_{i_1} \\ \vdots \\ p_{j_n} - p_{i_n} \end{bmatrix} = \begin{bmatrix} -I & I \end{bmatrix} \begin{bmatrix} p_i \\ p_j \end{bmatrix}$$

K tomuto vektoru můžeme přiřadit kovarianční matici

$$\begin{aligned} \Delta P_{ij} &= E[(\Delta \tilde{p}_{ij})(\Delta \tilde{p}_{ij})^T] \\ &= E\left[\begin{bmatrix} \tilde{p}_{j_1} - \tilde{p}_{i_1} \\ \vdots \\ \tilde{p}_{j_n} - \tilde{p}_{i_n} \end{bmatrix} \begin{bmatrix} \tilde{p}_{j_1} - \tilde{p}_{i_1} \\ \vdots \\ \tilde{p}_{j_n} - \tilde{p}_{i_n} \end{bmatrix}^T\right] \\ &= E\left[\begin{bmatrix} -I & I \end{bmatrix} \begin{bmatrix} \tilde{p}_i \\ \tilde{p}_j \end{bmatrix} \begin{bmatrix} \tilde{p}_i \\ \tilde{p}_j \end{bmatrix}^T \begin{bmatrix} -I \\ I \end{bmatrix}\right] \\ &= \begin{bmatrix} -I & I \end{bmatrix} \begin{bmatrix} P_{ii} & P_{ij} \\ P_{ji} & P_{jj} \end{bmatrix} \begin{bmatrix} -I \\ I \end{bmatrix} \end{aligned}$$

a tedy:

$$det(\Delta P_{ij}) = det\left(\begin{bmatrix} -I & I \end{bmatrix} \begin{bmatrix} P_{ii} & P_{ij} \\ P_{ji} & P_{jj} \end{bmatrix} \begin{bmatrix} -I \\ I \end{bmatrix}\right)$$

Protože ΔP_{ij} je PSD tak takže víme že

$$det(\Delta P_{ij}(k+1|k+1)) \leq det(\Delta P_{ij}(k|k))$$

Protože kovarianční matice je prostorovým vyjádřením jistoty v náš odhad, determinant je tedy skalárním vyjádřením jistoty. Můžeme tedy udělat závěr že:

Jistota mapy vytvářené MAK filtrem neklesá.

Protože kovarianci odhadu stavu robota a kovariance mezi odhady stavu robota a stavů objektů jsou ovlivněny predikční fází proto pro determinanty těchto kovariancí monotónnost neplatí. To znamená že mapa a robot se vyvíjí jako samostatné jednotky.

4.4 Shrnutí

V kapitole 4 jsme ukázali jak použít Kalmanův filtr na řešení SLAM problému. Ukázali jsme že SLAM nelze řešit jako N samostatných úloh pro N objektů, protože to by vedlo k nekonzistencím v mapě kvůli zanedbání korelací. Dále jsme ukázali ze kovariance jsou nerostoucí a

tedy jistota během vytváření mapy roste.

Také jsme určili že paměťové nároky a výpočetní složitost MAK filtru jsou řádu N^2. Proto se MAK filtr nehodí k implementaci v reálném čase a je potřeba použít jiná suboptimální řešení.

5 Suboptimální algoritmy

5.1 Úvod do suboptimálních algoritmů

Výpočet kompletního MAK filtru je ve větších prostředí příliš náročný, proto je potřeba volit jiné přístupy. Je více možných způsobů jak zjednodušit výpočet za cenu ztráty informace. Je vidět že náročnost MAK filtru je hlavně při udržování korelací.

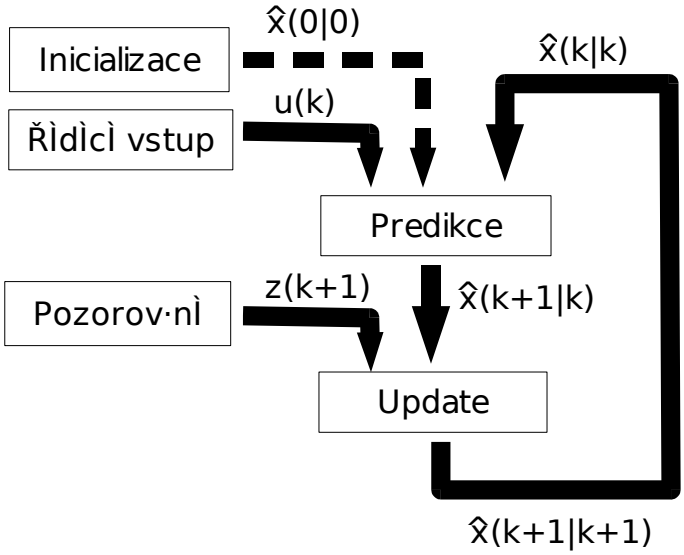
Nejdříve se zaměříme na BOR filtr. BOR filtr nepočítá korelace mezi objekty, pouze udržuje oblasti obalující skutečnou pozici objektu. Jinými přístupem je například odhadování relativních úhlů a vzdáleností mezi objekty. Dalším možným přístupem je používání MAK filtru lokálně, tedy s menším počtem objektů a pak tyto lokální oblasti, neboli skupiny objektů sdružovat a spojovat pomocí dalšího MAK filtru. [4]

V dalších kapitolách se potom zaměříme na Monte Carlo metody. Tyto metody používají váženou množinu vzorků pro simulaci možných stavů robota. Pro řešení SLAM problému je třeba ještě zahrnout i stav objektů. Pokud bychom použili Monte Carlo metodu na rozšířený stav robota se všemi objekty, museli bychom použít množinu s příliš mnoho vzorky, abychom dokázali simulovat stavy s takto velkou dimenzí. My si zde ukážeme jak se dá použít několik samostatných množin a jak tyto množiny udržovat tak ,abychom nepřišli o vzájemné vazby mezi objekty.

Existuji i další možné metody. Například J.E. Guivant s E.M. Nebotem v [9] ukázali, že optimálních výsledků lze dosáhnout i při aplikaci EKF pouze na lokální podmnožinu všech objektů.

5.2 BOR filtr

BOR filtr na rozdíl od Kalmanova filtru předpokládá že odhad je v určitých mezích. Filtr postupně mění tyto meze. Výhodou tohoto postupu je že nepočítá s korelacemi. Odhad stavu a s nim spojenou nejistotu tady zaznamenáváme jako oblast v stavovém prostoru které se odhad může nacházet. Kombinace dvou odhadů je pak průnik těchto oblastí.



Obrázek 5.1:BOR filtr

5.2.1 BOR filtr a SLAM

Aby šlo aplikovat BOR filtr na řešení SLAM problému je třeba, aby pozorování šlo obrátit relativně z pozice objektu.

Oblast robota se predikuje na základě oblasti ovládacího vstupu a šumu děje. Oblasti objektů se během predikce nemění.

Pozorování se přepočítá do souřadnic mapy a také do relativních souřadnic robota.

Kalmanův filtr hledá důležitou informaci v pozorování pomocí kovariancí, BOR filtr tuto informaci získává přímo průnikem oblasti pozorování s odhadem oblasti.

Paměťové nároky BOR filtru při řešení SLAM prostoru rostou pouze lineárně s počtem objektů a výpočetní složitost se nemění viz [4].

5.2.2 Implementace BOR filtru

Pro implementaci BOR filtru se voli buď elipsoidy nebo kvádry. Kvádry mají výrazně jednodušší implementaci v několika bodech. Hledání průsečíku dvou oblastí se provádí za pomocí dvou porovnání v každé ose. Definice oblasti obsahuje pouze dva parametry pro každou osu stavového prostoru.

Odhad stavu můžeme zapsat takto:

$$\hat{x}|_b = \{x | x_i^{min} \leq x_i \leq x_i^{max}\}$$

kde x_i je i-tý rozměr x a x^{min}, x^{max} jsou dolní a horní hranice intervalu v daném rozměru. Meze šumu lze jednoduše definovat pro každou osu zvlášť, tedy lze jednoduše oddělit různé zdroje šumu.

5.2.3 Definice odhadů

Odhad stavu robota a odhady stavů objektů zapíšeme následovně:

$$\hat{x}_r|_b(k|k) = \begin{bmatrix} \hat{x}_r|_{min}(k|k) \\ \hat{x}_r|_{max}(k|k) \\ \hat{y}_r|_{min}(k|k) \\ \hat{y}_r|_{max}(k|k) \\ \hat{\phi}_r|_{left}(k|k) \\ \hat{\phi}_r|_{right}(k|k) \end{bmatrix}$$

$$\hat{p}_i|_b(k|k) = \begin{bmatrix} \hat{x}_i|_{min}(k|k) \\ \hat{x}_i|_{max}(k|k) \\ \hat{y}_i|_{min}(k|k) \\ \hat{y}_i|_{max}(k|k) \end{bmatrix}$$

Kde min a max značí dolní a horní hranici u pozice. Left a right značí levý a pravý okraj u úhlů. Musí platit že skutečná pozice se nachází v kladném směru, tedy proti směru hodinových ručiček, od pravého okraje, blíže než levý okraj.

5.2.4 Predikce stavu robota

Predikce stavu robota je oblast ve stavovém prostoru obsahující skutečný stav:

$$\hat{x}_r|_b(k+1|k) = \begin{bmatrix} \hat{x}_r|_{min}(k+1|k) \\ \hat{x}_r|_{max}(k+1|k) \\ \hat{y}_r|_{min}(k+1|k) \\ \hat{y}_r|_{max}(k+1|k) \\ \hat{\phi}_r|_{\text{right}}(k+1|k) \\ \hat{\phi}_r|_{\text{left}}(k+1|k) \end{bmatrix}$$

kde

$$\hat{x}_r|_{min}(k+1|k) = \hat{x}_r|_{min}(k|k)+min_{\hat{x}_r|_b(k|k),V(k|k)}(T[V(k)+v_r]\cos(\phi_r+\gamma(k)+v_\gamma))$$
$$\hat{x}_r|_{max}(k+1|k) = \hat{x}_r|_{max}(k|k)+max_{\hat{x}_r|_b(k|k),V(k|k)}(T[V(k)+v_r]\cos(\phi_r+\gamma(k)+v_\gamma))$$
$$\hat{y}_r|_{min}(k+1|k) = \hat{y}_r|_{min}(k|k)+min_{\hat{x}_r|_b(k|k),V(k|k)}(T[V(k)+v_r]\sin(\phi_v+\gamma(k)+v_\gamma))$$
$$\hat{y}_r|_{max}(k+1|k) = \hat{y}_r|_{max}(k|k)+max_{\hat{x}_r|_b(k|k),V(k|k)}(T[V(k)+v_r]\sin(\phi_r+\gamma(k)+v_\gamma))$$
$$\hat{\phi}_r|_{\text{right}}(k+1|k) = \text{right}_{\hat{x}_r|_b(k|k),V(k|k)}\left(\phi_r+\frac{T[V(k)+v_r]}{B}\sin(\gamma(k)+v_\gamma)\right)$$
$$\hat{\phi}_r|_{\text{left}}(k+1|k) = \text{left}_{\hat{x}_r|_b(k|k),V(k|k)}\left(\phi_r+\frac{T[V(k)+v_r]}{B}\sin(\gamma(k)+v_\gamma)\right)$$

Funkce min a max vracejí minimum a maximum přes všechna x_r z $\hat{x}_r|_v(k|k)$ a v z V(k). Funkce left a right pak vracejí levý a pravý okraj všech možných orientací kterých lze dosáhnou přes všechna x_r z $\hat{x}_r|_v(k|k)$ a v z V(k).

5.2.5 Pozorování

Stejně jako u Kalmanova filtru, tak i BOR filtr používá pozorování pro update odhadu stavu robota a odhady stavů objektu. Update probíhá jako průnik dvou oblastí stavového prostoru. Jedna oblast je současný odhad stavu. Druhou oblast dostaneme na základě pozorování.

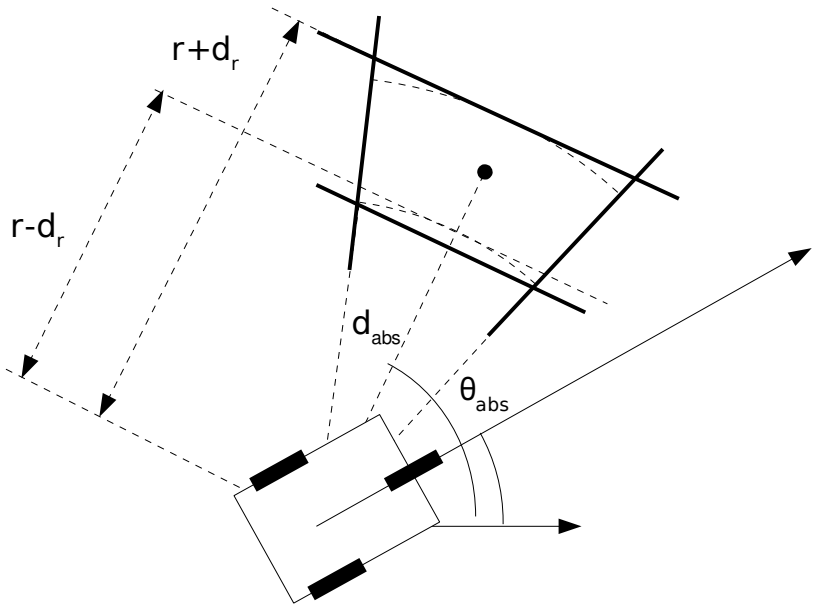
Pozorování probíhá relativně k pozici robota, pro vytvoření oblasti ho musíme převést do absolutních souřadnic stavového prostoru. Nejdříve určíme absolutní úhel pozorování. Úhel robota je zapsán jako rozmezí ϕ-d_ϕ až ϕ+d_ϕ. Pozorování je pak zapsáno jako relativní úhel θ-d_θ až θ+d_θ. Absolutní úhel pozorování je pak

$$\theta_{\text{abs}} = \theta+\phi$$
$$\theta_{\text{abs|left}} = \theta+\phi+(d_\theta+d_\phi)$$
$$\theta_{\text{abs|right}} = \theta+\phi-(d_\theta+d_\phi)$$
$$d_{\text{abs}} = d_\theta+d_\phi$$

Pokud k absolutním hranicím přičteme π radiánů dostaneme rozsah obráceného pozorování, tedy pozorování robota z objektu. Hranice vzdálenosti robota od pozorovaného objektu zůstává u obráceného pozorování stejná

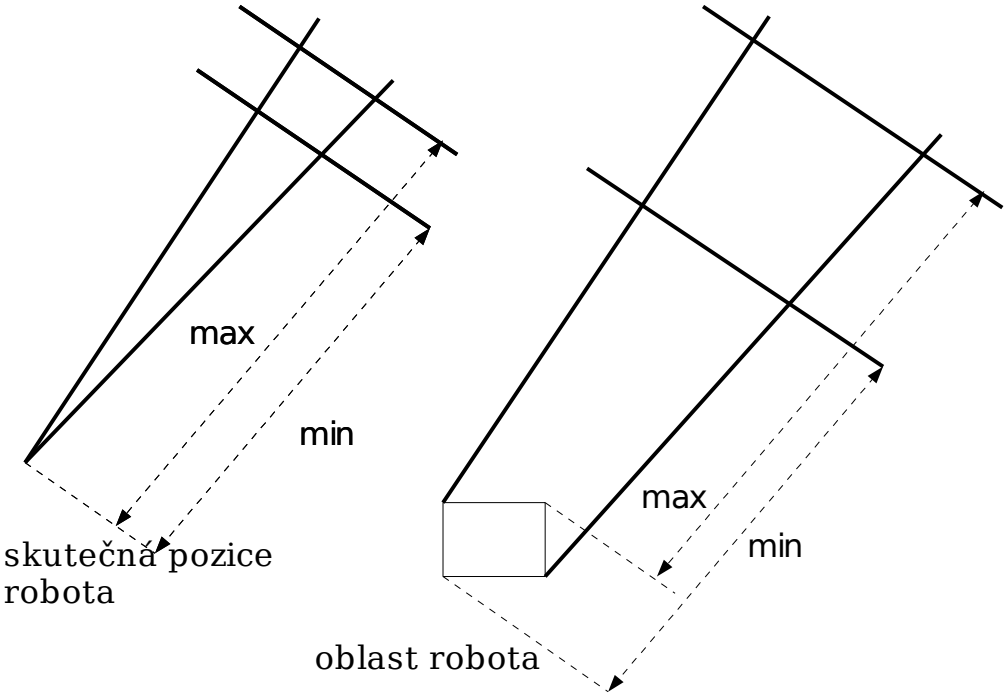
5.2.6 Update pozice objektu

Když máme oblast odhadu stavu robota, hranice vzdálenosti a hranice absolutního úhlu pozorování, můžeme sestrojit oblast kde se podle pozorování může objekt nacházet. Oblast pozorování budeme definovat pomocí hran jak je vidět na obrázku 5.1. Hrana je přímka která dělí rovinu na platnou a neplatnou polorovinu. Platnou oblastí pak je průnik všech čtyr polorovin. Protože skutečná oblast pozorování je část mezikruží, tak při použití hran se naše oblast pozorování zvětší i malé oblasti kde se pozorovaný objekt nemůže nacházet. Díky tomu ale dosáhneme výrazně jednoduššího výpočtu updatu.



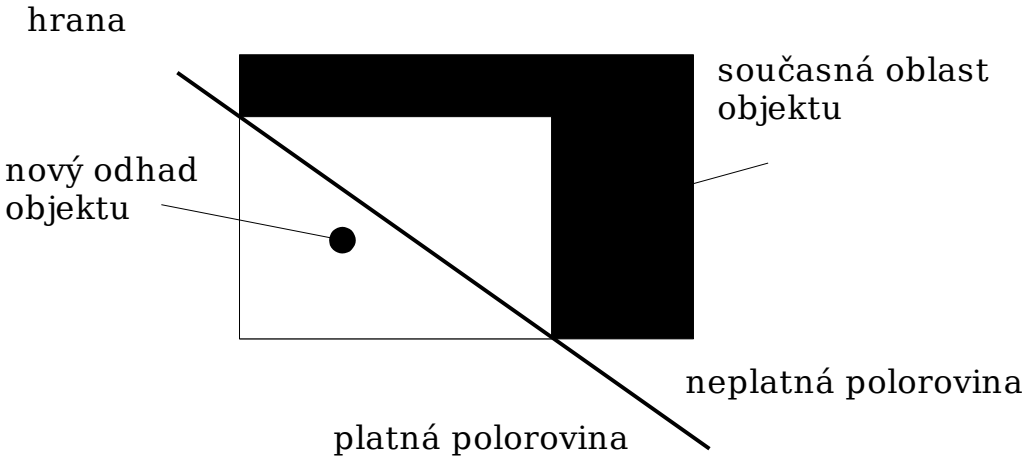
Obrázek 5.1: Platná oblast pozorování

Hrany které vznikly na základě vzdálenosti jsou kolmé na úhel pozorování. Vzdálenější je $r+d_r$, bližší je potom ($r-d_r$) cos(d_{abs}). Protože pozice robota je oblast, musíme tuto platnou oblast pozorování ještě rozšířit tak aby v absolutních pozicích pokrývala pozorování ze všech možných pozic robota.



Obrázek 5.2: rozšíření pozorování podle oblasti robota

Potom co každou hranu posuneme podle oblasti robota, použijeme ji na upřesnění nové oblasti pozorovaného objektu. Každá hrana rozděluje rovinu na platnou a neplatnou část. Protože oblast je pouze obdélníková a rovnoběžná s osami prostoru musíme vzniklou oblast obalit do nejmenší obdélníkové oblasti jako na obrázku 5.3.



Obrázek 5.3: update oblasti objektu

5.2.7 Update pozice robota

Update stavu robota probíhá stejným způsobem. Použijeme obráceného pozorování, tedy pozorování sestrojené na základě skutečného pozorování, které odpovídá pozorování robota z pozice objektu. Toto pozorování dostaneme tak že k hranicím úhlu pozorování přičteme π radiánů. Hranice vzdálenosti zůstane stejná. Pro rozšíření použijeme oblast objektu a získané hrany použijeme pro update oblasti robota.

5.2.8 Update orientace robota

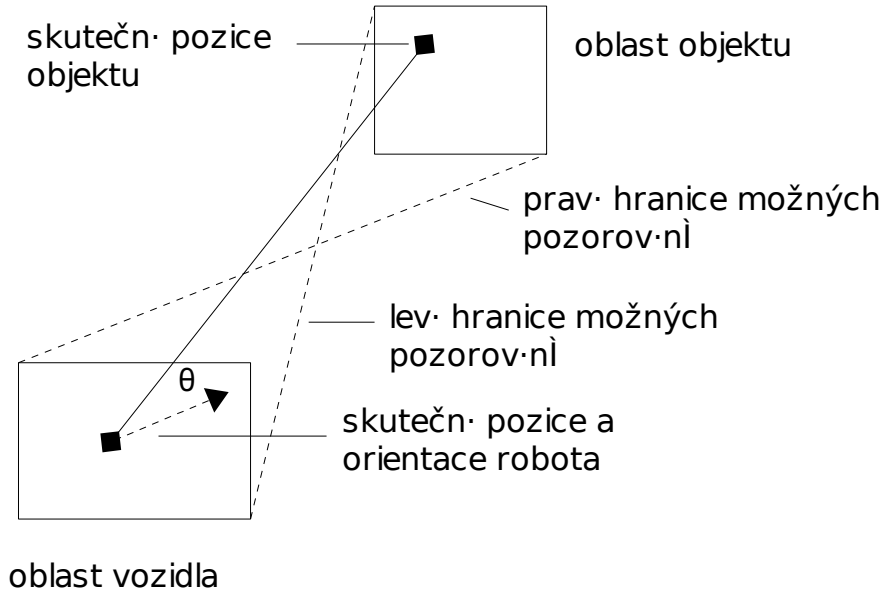
Update orientace robota se provádí pomocí průniku stávající orientace robota s orientací vypočítanou na základě pozorování.

Abychom dostali tento rozsah vezmeme oblast pozorovaného objektu a oblast robota.

5.2.9 Výběr pozorovaného objektu

Po provedení měření je potřeba určit který objekt je pozorován. Po pozorování si sestrojíme platnou oblast pozorování podle kapitoly 5.2.7. Zjistíme jestli se překrývá s některými z objektů. Pokud se překrývá právě s jedním z objektů, řekneme že se jedná o pozorování tohoto objektu. Podle něho si uděláme update pozice tohoto objektu, update pozice robota a update orientace robota.

Pokud se pozorovaná oblast nepřekrývá s žádným z objektů, vytvoříme z tohoto pozorování nový objekt. Potřebujeme zvážit všechny možné případy pozorování tedy všechny možné pozice robota uvnitř oblasti robota a zároveň všechny možné pozice objektu uvnitř pozice objektu. Ze všech těchto pozic chceme určit krajní hranice úhlu pod kterým pozorování mohlo proběhnout. Pokud se oblasti překrývají, update orientace robota nepřináší žádnou informaci, protože rozmezí úhlů je neomezené. Takto spočítaná hranice možných úhlů pozorování se otočí o změřený úhel relativního pozorování. Rozsah je ještě třeba rozšířit o úhel d_θ což je nepřesnost měření. Tím dostaneme rozsah možných orientací robota podle pozorování. Tento rozsah použijeme pro update orientace a to jednoduchým průnikem se stávajícím odhadem.



Obrázek 5.4: hranice možných orientací robota

5.2.10 Funkce BOR filtru

- **Začátek** výpočet začínáme po prvním pozorování objektu, odhad stavu robota se inicializuje do počátečního stavu. Pozorování se porovná s objekty které už na začátku jsou v mapě, pokud se jedná o pozorování existujícího objektu proběhne update, jinak se do mapy přidá nový objekt.
- **Predikce** predikce nových hranic odhadu pozice a orientace robota na základě předešlého stavu, ovládacího vstupu a šumu
- **Určení objektu** spočteme absolutní oblast pozorování, rozšíříme ji podle oblasti robota, pokud se tato oblast protíná právě s jedním z objektu, provedeme update, jinak vytvoříme nový objekt
- **Update pozice objektu** vezmu jednotlivé hrany rozšířené oblasti pozorování a ořezáváme s nimi současnou oblast objektu, po každém oříznutí vezmeme nejmenší obdélník obsahující platnou oblast
- **Update pozice robota** spočítáme rozšířenou oblast obráceného pozorování. vezmu jednotlivé hrany této oblasti pozorování a ořezáváme s nimi současnou oblast robota, po každém oříznutí vezmeme nejmenší obdélník obsahující platnou oblast

- **Update orientace robota** spočítáme hranice možného úhlu pozorování podle oblasti robota a oblasti pozorovaného objektu, otočíme tyto hranice podle úhlu pozorování a rozšíříme o možnou chybu pozorování, tuto oblast proniknu se stávající oblastí orientace robota
- **Konec** opakujeme update orientace a pozice dokud se oblasti znatelně zmenšují

5.3 Monte Carlo lokalizace

Monte Carlo metody (dále jen MCL Monte Carlo Localization[10]) se úspěšně používají pro lokalizaci robotů. V následující kapitole ukážeme jak se dají tyto metody upravit pro řešení SLAM problému.

5.3.1 Úvod do MCL metod

Sekvenční Monte Carlo metody používají simulační přístup pro odhad pravděpodobnostního rozložení. Jejich výhodou je relativně jednoduchá implementace a také to, že nevyžadují Gaussiánské rozložení. Jako MCL se označuje implementace Monte Carlo metod pro lokalizaci robota. Podrobněji v [10]

5.3.2 Průběh MCL algoritmu

Pro použití MCL je vhodné si odhadování vyjádřit jako hledání pravděpodobnostního rozložení. Skutečný stav označíme jako $\{x_t; t \in \mathbb{N}\}$ hledáme tedy pravděpodobnostní rozložení $p(x_{0:t}|y_{1:t})$ kde $\{y_t; t \in \mathbb{N}\}$ jsou pozorování. Tuto pravděpodobnost můžeme vypočítat pomocí Bayesovi věty: $p(x_{0:t+1}|y_{1:t-1}) = p(x_{0:t}|y_{1:t}) \frac{p(y_{t+1}|x_{t+1}) p(x_{t+1}|x_t)}{p(y_{t+1}|y_{1:t})}$

Pravděpodobnost můžeme zapsat jako víru (belief) v daný stav. $Bel(x_t) = p(x_t|y_t, u_{t-1}, y_{t-1}, \ldots, u_{0,} y_0)$

Bayesova věta vyjádřena jako belief:

$$Bel(x_t) = k \; p(y_t|x_t) \int p(x_t|x_{t-1}, u_{t-1}) Bel(x_{t-1}) \; dx_{t-1} \quad (5.1)$$

kde k je váhová konstanta.

Množinu vážených vzorků pak označíme jako $\{x^i, f^i\}_{i=1,..,m}$ kde f je váha příslušného vzorku. Počáteční hodnota f^i je 1/m pro všechna i. Výpočet pak probíhá podle následujícího algoritmu:

algoritmus 1:

---

for všechny vzorky do:

1) vyber vzorek s největším f_{t-1}

2) vypočítej x_t^i *a podmíněnou pravděpodobnost* $p(x_t|x_{t-1}, u_{t-1})$

3) vypočítej pravděpodobnost $p(y_t|x_t^i)$

4) vypočítej Bel(x_t*)*

done

znormalizuj váhy f_t *z Bel(*x_t*) tak aby* $\sum_{i=1}^{m} f^i = 1$

kroky 1-4 vlastně implementují výpočet rovnice (5.1) zprava doleva. Nejdříve vyberu vzorek i podle jeho váhy z předešlé iterace, poté vypočítám nový vzorek a jeho pravděpodobnost ze stavu vzorku z předešlé iterace a předešlého vstupu u_{t-1}. V 3. kroku vypočítáme pravděpodobnost nového pozorování na základě nově vypočítaného vzorku. Z těchto výsledků pak vychází váha nového vzorku. Na konci iterace je ještě třeba normalizovat váhy.

5.3.3 MCL a SLAM

SLAM musí odhadovat zároveň pozici robota a objektů. Jedním z možných přístupů je, stejně jako u MAK filtru, rozšířit stav robota i o stav všech objektů. potom je ale potřeba výrazně více vzorků, právě kvůli zásadnímu zvětšení dimenze stavového prostoru. Také by bylo obtížné sestrojit funkci pro výpočet vah. Proto zde ukážeme jiný přístup. Sestrojíme si několik množin vzorků, které budeme udržovat stejně jako v algoritmu 1.

Množinu vzorků pro stav robota označíme jako $V_r = \{x^i, f^i\}_{i=1,..,m}$. Tyto vzoky na začátku náhodně rozprostřeme okolo startovní pozice robota. Protože při řešení SLAM problému na začátku nemáme žádné informace o mapě, můžeme startovní pozici robota označit za počátek souřadnic. Výpočet pak probýhá podle algoritmu 2. Na začátku každého kroku zpracujeme naměřená data. Pokud najdeme nový objekt vytvoříme novou množinu vzorků V_L, s n vzorky $V_L = \{x^i, f^i\}_{i=1,..,n}$, kde L číslo nového objektu. Všechny tyto množiny V_0 - V_L se upravují podle algoritmu 1.

algoritmus 2:

Vytvoř množinu V_r pro stav robota

L = 0

repeat

if (našel jsi nový objekt) then

L=L+1

Vytvoř novou množinu V_L pro stav objektu

endif

Přepočítej všechny V_r, V_1, …,V_L podle algoritmu 1

until konec

Protože jak pozice robota, tak pozice objektů je neznámá, musíme při updatu množiny V_j použít jako pozici ostatních objektů a robota vždy náhodný vzorek z příslušných množin.

5.3.4 Stabilizace objektů

Během výpočtu se pozice objektů postupně upřesňuje. Pokud se všechny vzorky v příslušné množině dostatečně přiblíží, mohu tuto pozici prohlásit za správnou a dále už počítat pouze s touto pevnou pozicí. Tím se jednak zjednoduší výpočet, protože nemusíme upravovovat příslušnou množinu. Také tím odpadá problém s odhady objektů, které jsou dlouho mimo dohled senzorů. Odhady těchto objekty totiž mají tendenci se vzdalovat od skutečné pozice.

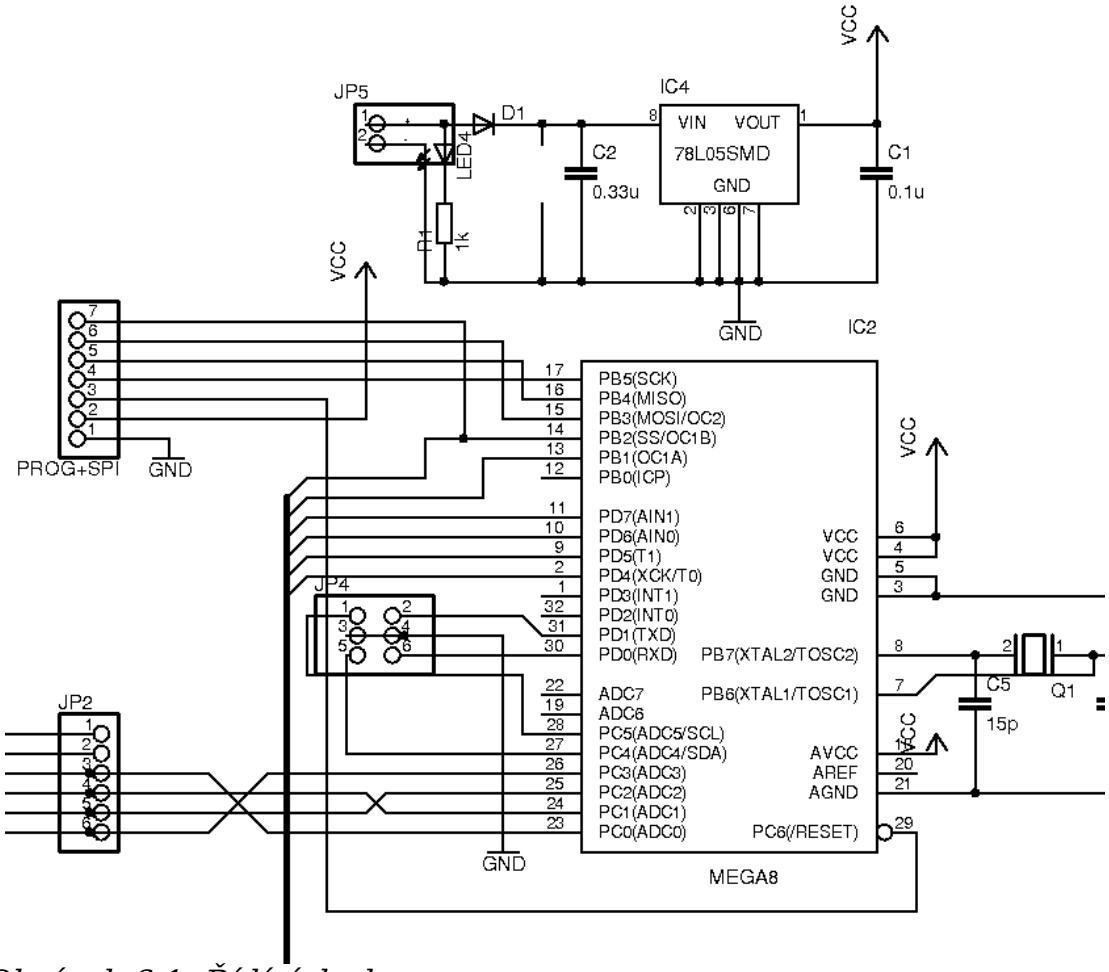
6 Konstrukce

6.1 Robot

Pro ověření funkčnosti suboptimálních algoritmů v reálném prostředí bylo potřeba vytvořit základní platformu. Z důvodu ušetření financí a času jsem se rozhodl použít jako základ podvozek z robotické hračky čínské výroby. Úspora času se nakonec sice ukázala jako sporná, protože bylo potřeba udělat mnoho úprav a oprav, ale přesto tento podvozek považuji za vhodný základ. Robot má čtvercovou základnu přibližně 12x12cm. Pohon je řešen jako diferenciální s dvěma hnanýma koly v zadní části a dvěma kluznýma koly vepředu. Původní napájení bylo z 4 kusů AA baterii.

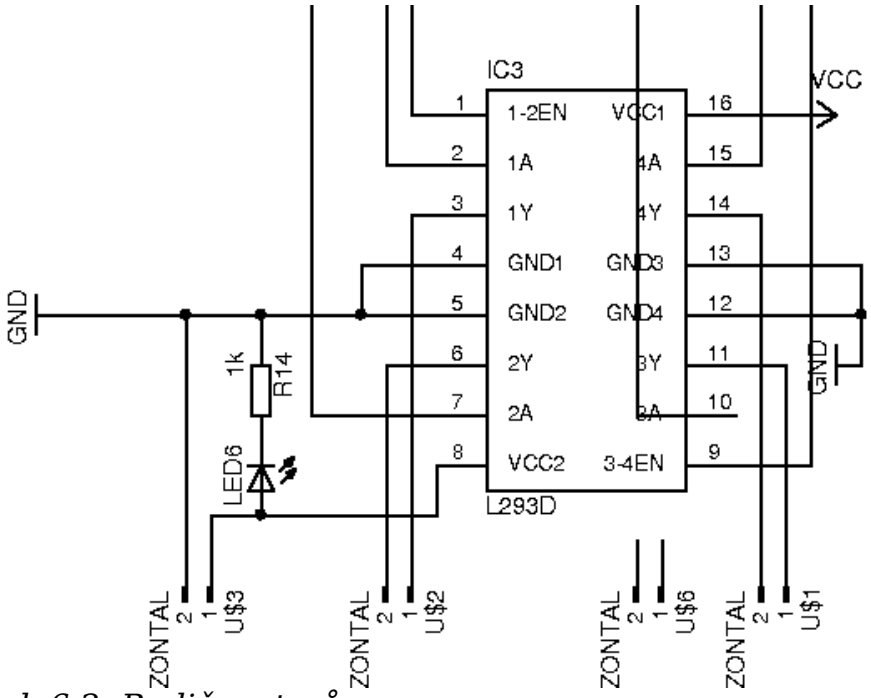
6.2 Řídící elektronika

Pro řízení motorů bylo třeba vyrobit řídící elektroniku. Jako základ jsem se rozhodl použít mikrokontrolér ATMega8 od firmy Atmel. Jedná se o RISCový mikrokontrolér z řady AVR. Kvůli jednoduchosti vývoje jsem od začátku počítal se zpracováním naměřených dat na pc. Proto na úrovni mikrokontroléru není potřeba téměř žádný výkon, mikrokontrolér se stará pouze o přímé ovládání hardwaru, veškerá data posílá po sériové do PC odkud zpětně přijímá příkazy.



Obrázek 6.1: Řídící deska

Jako budič motorů jsem pak použil integrovaný H-můstek L293D. Jde o klasický tranzistorový můstek, takže na něm je poměrně velká ztráta. Vzhledem k tomu, že jsem použil napájecí napětí o něco vetší než původní napájení motorů, tak mě tato ztráta nijak nevadila. Zatížení motorů bylo natolik malé, že se můstek jenom lehce zahříval. Pokud se motory zastavily úplně, tak při maximálním použitém 75% plnění se můstek silně zahříval, ale jeho přehřátí nehrozilo.

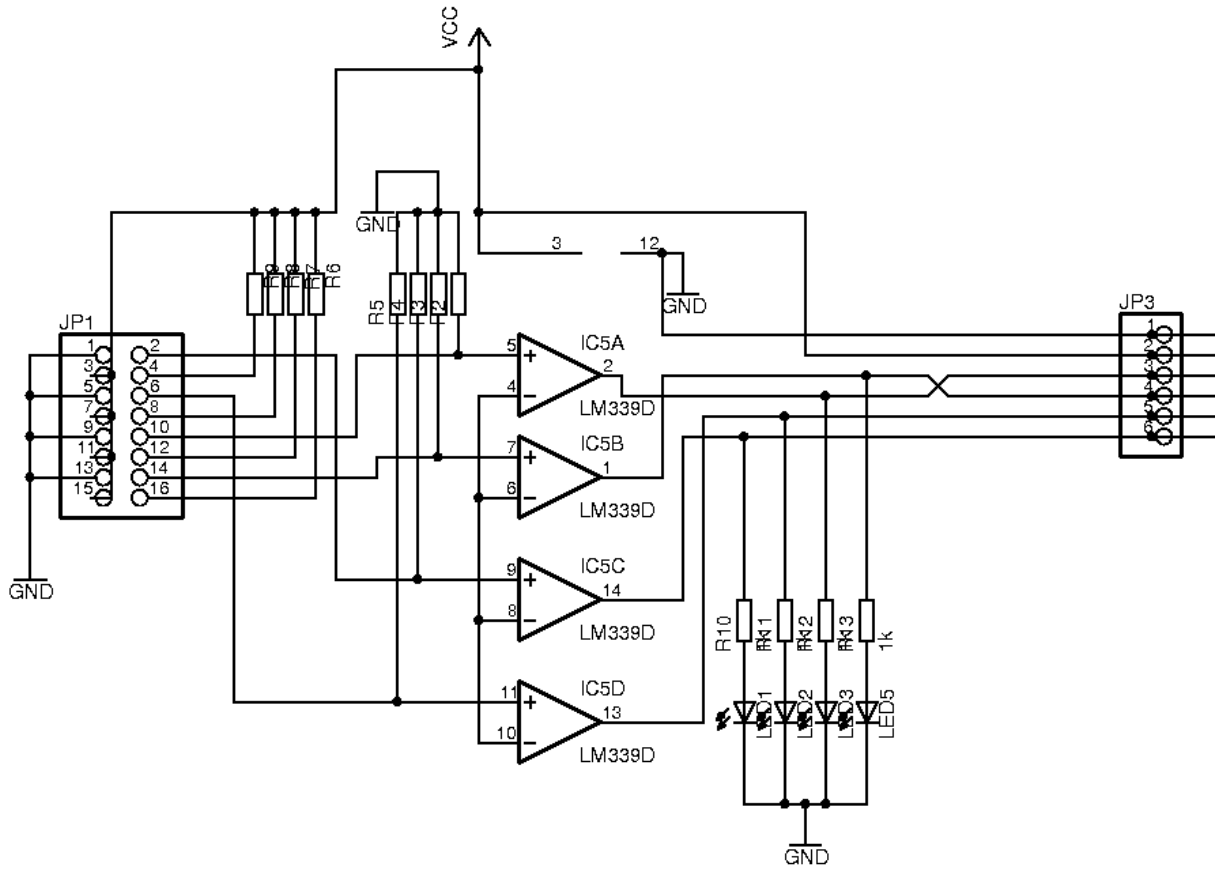


Obrázek 6.2: Budič motorů

6.3 Enkodéry

Pro zpětnou vazbu z motorů a pro měření ujeté vzdálenosti bylo třeba robota vybavit enkodéry. V první verzi jsem použil jednoduché optoreflexivní čidla CNY70 a na hnaná kolečka jsem nalepil proužky z hliníkové folie. Toto řešení se sice ukázalo funkční, ale poskytovalo poměrně malé maximální rozlišení asi 7mm. Proto jsem se rozhodl umístit enkodéry přímo na osu motoru. Použité motory měly malý přesah osy na druhou stranu od převodovky. Na tento přesah jsem uchytil enkodérová kolečka ze staré kuličkové myši. Toto uchycení se ukázalo jako poměrně nespolehlivé a přineslo poměrně dost starostí a stalo se nejčastěji opravovanou součástí robota. Enkodérová kolečka z myši totiž přinášejí veliké rozlišení, ale jsou poměrně citlivá na přesnost uchycení koleček a umístění optosenzorů. Senzory jsem použil přímo z rozebrané myši. Díku tomu jsem měl zaručené správné fázové posunutí fototranzistorů pro kvadraturní signál a tím informaci o rychlosti i směru točení motoru. Signál všech kanálů jsem pak komparátorem LM339D porovnával s napětím nastaveném ladícím potenciometrem. Tady jsem narazil na nečekaný problém, kdy výstupní signál enkodéru měl tvar přibližně sinusoidy s amplitudou 1 až 3V. S rostoucí rychlostí amplituda klesala k 1V, to by ale nebyl problém. Problém vznikal tím že s rostoucí rychlostí se měnil offset signálu, tedy při maximální rychlosti byl rozsah napětí 3,5V-4,1V, zatímco nižší rychlosti měly rozsah 2,4V-3,9V. Tento problém zřejmě vzniká kvůli nedostatečné rychlosti fototranzistorů. Možné řešení by bylo sehnat další stejnou sadu fototranzistorů a tu umístit do protifáze k těm současným, potom by stačilo komparátorem porovnávat napětí na

dvou tranzistorech v protifázi. To by ale vyžadovalo velikou přesnost uchycení. Velikost jednoho proužku enkodérového kolečka je přibližně 0,5mm. Pokoušel jsem se to na jednom motoru implementovat, ale nakonec jsem od toho musel ustoupit, bylo by třeba úplně předělat uchycení tranzistorů a také už to hrál větší roli nepřesný chod kolečka vzhledem k jeho uchycení. pro měření na jednom místě není tato vlastnost tak zásadní, mezi fototranzistorem a diodou stále přecházejí proužky správné velikosti, ale když se měří na dvou místech, není zachováno správné posunutí fází. Nakonec jsem se s funkčností enkodérů z myši spokojil s tím že jsem softwarově omezil maximální rychlost na přibližně 75 procent. V tomto rozsahu se ukázalo, že enkodéry fungují dobře jenom s malým procentem výpadků. Jejich fyzická odolnost byla bohužel nižší a při delším testování bylo otázkou času kdy jeden z nich přestane fungovat.



Obrázek 6.3: Zpracování enkodérů

6.4 Napájení

Vzhledem k tomu, že jsem od začátku plánoval použití robota připojeného na kabel pro komunikaci s počítačem, mohl jsem počítat i pouze s externím napájením. Kvůli napěťovým špičkám při rozjezdu motorů jsem ale rozhodl že použiji palubní baterie, externí napájení pak

použiji pro průběžné dobíjení baterií. Toto řešení zajišťuje celkem spolehlivé napájení všech komponent robota a umožňuje i budoucí další využití robota pro zcela autonomní pokusy. Jako baterie jsem použil čtyři Li Ion články o kapacitě 2000 mAh v zapojení 2P2S tedy dvě paralelně dvě sériově. Tím jsem dostal kapacitu 4000Ah při rozsahu napájení přibližně 6-8,4V. Zapojení nabíječe jsem převzal z internetu od pana Zajíce [11]. Jedná se o jednoduché zapojení zdroje konstantního napětí s omezením proudu. Toto řešení je levné a hlavně již mnohokrát ověřené. Já jsem použil zapojení pro 2 sériově zapojené články s omezením proudu přepínatelným mezi 0,3A a 1A. Kde 0,3A se používalo standardně za běhu, aby nedocházelo k přetížení dlouhého napájecího kabelu. 1A byla použita pouze při přímém připojení robota ke zdroji pro rychlejší dobíjeni. Napájení z baterií bylo pak po robotu rozvedeno přímo, pouze přes vratnou pojistku. Všechny moduly měly potřebný stabilizátor napětí přímo na sobě.

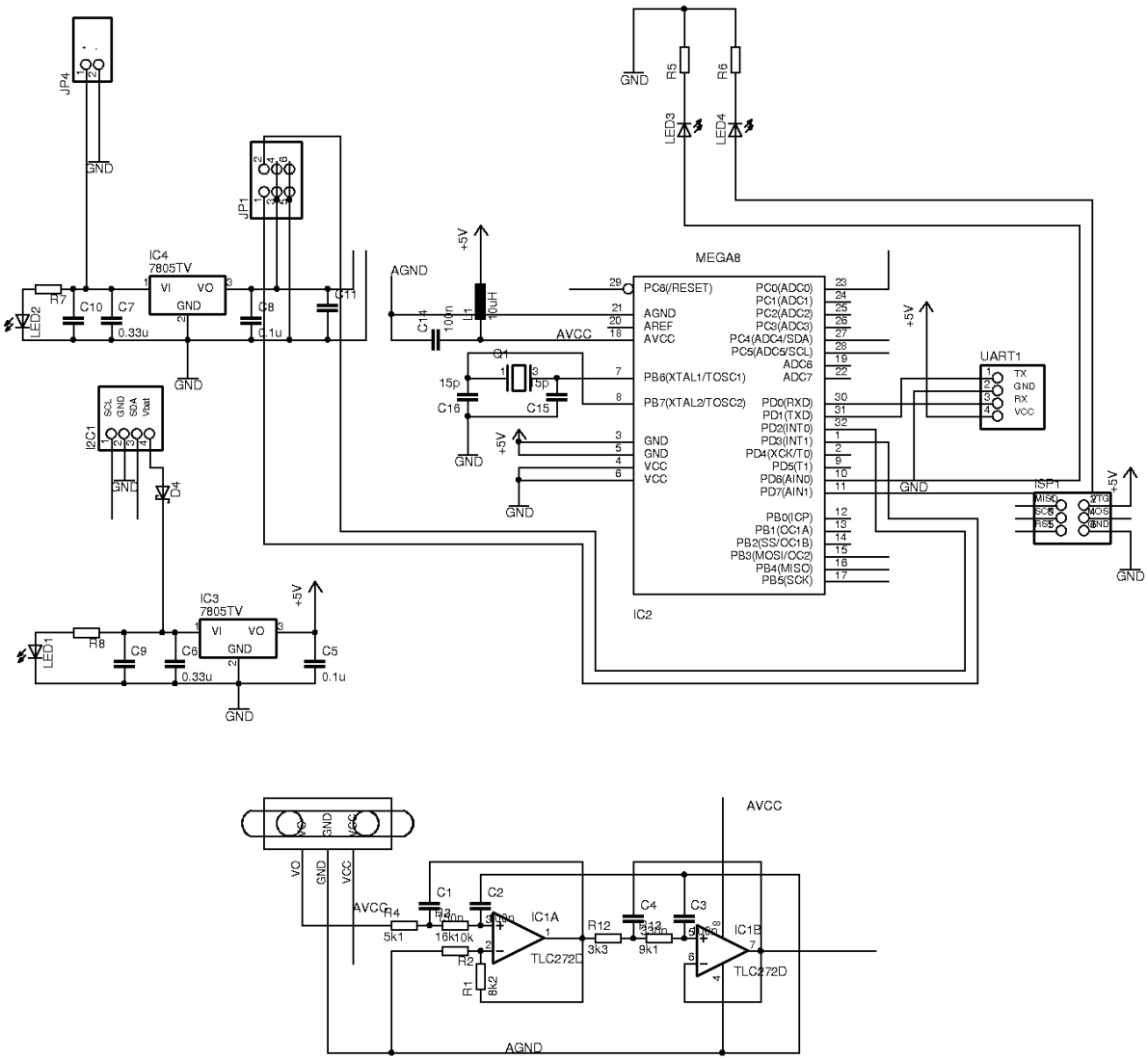
6.5 Čidla

Při výběru čidel jsem zvažoval jak jednoduše získat dostatek dat pro úspěšnou stavbu mapy. většina studií se zabývá tvorbou mapy z laserových scannerů. Tyto scannery ale stojí řadově několik stovek tisíc korun a jsou poměrně velké a těžké. Jednou z možností by bylo použít kameru. Obyčejná web kamera je dnes velice levná záležitost. Hledání objektů v obraze kamery a obecně zpracování obrazu je zajímavá oblast, ale je velice náročná a obsahově mimo zaměření této práce. Zajímavé výsledky z použití jediné kamery dosahuje Andrew Davison v [12] a navazujících pracích. Jeho práce se nazývá MonoSlam a dokáže dělat lokalizaci v neznámém prostředí pouze za použití dat z obyčejné kamery bez omezujících předpokladů na její pohyb.

Dalším možným řešením by bylo použít v prostředí umělé značky. Například pasivní majáčky z odrazivé folie, které se dají hledat pomocí laserového paprsku. Umělé značky ale značně omezují možnost dalšího použití, víceméně je to totiž pohyb robota ve známém prostředí. Majáčky by si robot teoreticky mohl do prostředí přidávat sám, ale to je stále velice omezující. Navíc by to bylo i mechanicky náročné. Více o navigaci umělými majáčky například v [13] nebo [1].

Já jsem se tedy zaměřil na jednoduchá čidla pro měření vzdálenosti. Z dostupných čidel připadaly v úvahu senzory založené na odrazu ultrazvuku a infračervených paprsků. Po několika pokusech s ultrazvukovým čidlem a prostudování materiálů jsem usoudil že infra čidla budou pro moje použití lepší. Za cenu okolo 400Kč se dají u nás sehnat čidla firmy SHARP z řady GP2D. Jsou to infra odrazová čidla fungující na principu měření vzdálenosti místa dopadu odraženého paprsku. Mají integrovanou logiku a jejich výstupem je analogové napětí odpovídající vzdálenosti od odrazové plochy. Já jsem zvolil čidlo GP2D120 s rozsahem měření 4-30cm. V první verzi bylo toto čidlo připojeno přímo na analogový vstup řídícího mikrokontroléru. Pro ověření funkce to bylo dostatečné, ale ukázalo se že čidlo je vrací dost zašuměné hodnoty. Na rady zkušenějších elektroniků jsem se tedy rozhodl vytvořit samostatný

modul pro zpracování čidla. Tento modul je umístěn přímo na tělo čidla a obsahuje analogový filtr z dvou operačních zesilovačů, samostatný mikrokontrolér pro měření, oddělené napájení pro analogovou část. Navíc obsahuje dva výstupy pro řízení serv. Přímým připojením serva, které senzor natáčí jsem dostal vlastně jakýsi jednoduchý 2D scanner. Druhé servo jsem zatím nevyužil, je tam pro případné další pokusy s naklápěním senzoru v 3d.



Obrázek 6.4: Modul IR čidla

Modul je možné k řídící desce robota připojit pomocí sběrnice I2C. Pro jednoduchost jsem ho ale měl připojen přímo na druhý sériový port počítače.

7 Software

7.1 Řídící deska

Řídící deska je řízena mikrokontrolérem ATMega8. Její software je psán v C a překládán pomocí GCC překladače, který je pro tyto procesory k dispozici. Program je velice jednoduchý, cyklicky vysílá do PC strukturu se svým stavem a přijímá strukturu s novým stavem. Pro tento účel deska posílá stav enkodérů a přijímá požadovanou rychlost motorů.

7.2 Čidlo

Program pro čtení čidla je opět velice jednoduchý. Cyklicky provádí A/D převod hodnoty na servu. Pokud se PC zeptá na hodnotu, program mu vrátí poslední naměřený údaj. Také si průběžně počítá klouzavý průměr z posledních 16 měření. Tento údaj se teď nikde nepoužívá a sloužil pro ověření stálosti měřených dat. Program umožňuje z PC nastavit natočení serva. PC také může požádat o vytvoření scanu. Scan se provádí tak že servo se postupně pootáčí o konstantní krok v jednom směru a vždy po chvilce na ustálení se po sériové lince posle do PC informace o změřené vzdálenosti.

7.3 PC

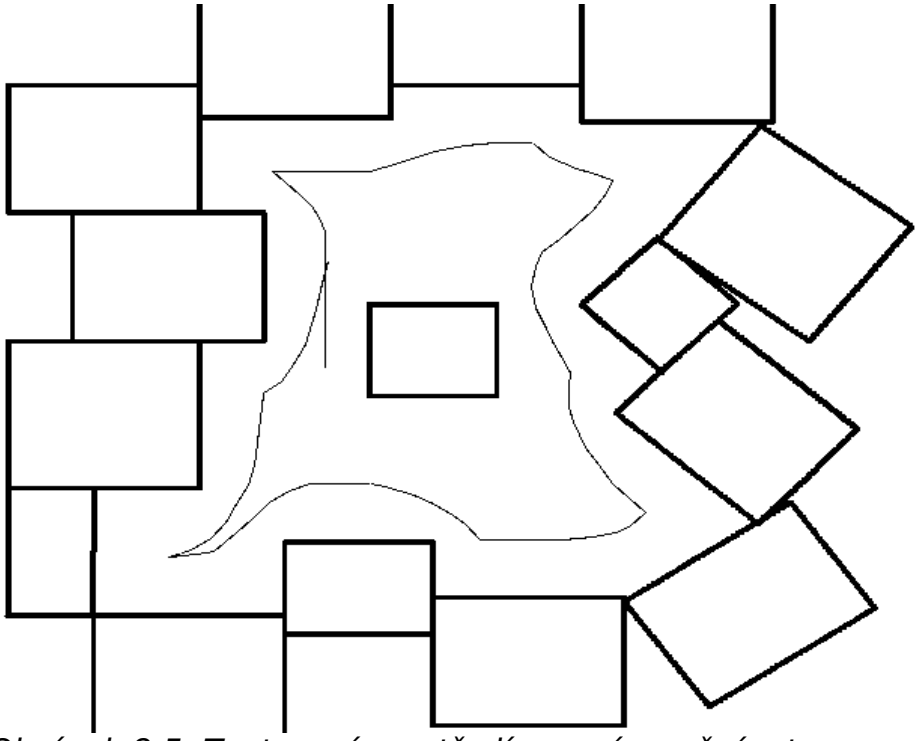
Kvůli jednoduchosti dalšího vývoje jsem řízení robota implementoval do robotickeho frameworku Player [14] Pro sběr dat jsem pak používal pouze velice jednoduchý prográmek, který jezdil po ploše a ukládal data o ujeté vzdálenosti a naměřených scanech do souboru. Vyšší algoritmy jsem pak pouštěl zvlášť na naměřených datech. Protože robot neprovádí v závislosti na mapování ani lokalizaci žádné akce, není potřeba provádět pokusné výpočty v reálném čase na skutečné robotu. Program pro sběr dat měl za úkol jezdit podél levé zdi. Pro jednoduchost jezdil pouze rovně a otáčel se na místě. Program vypadal následovně:

```
for(;;){
 pp.SetSpeed(0,0,0);//zastav pro měření
 robot.Read(); //synchronizace struktur s řídící deskou
 ir.scan(); //proveď scan
for (unsigned int i=0;i<sp.GetCount();i++) {//vypiš všechny hodnoty
   std::cout << sp[i] << ";";
 }
  std::cout << std::endl << std::endl;
 ir.setAngle(-45);//nastav servo na 45stupnu doleva

          pp.SetSpeed(0,turnrate); //nastav otáční doleva
while(ir.read()>0.25){//dokud jsi příliš daleko od stěny
   robot.Read();//synchronizace struktur
 }
pp.SetSpeed(0,-turnrate);// nastav otáčení doprava
while(ir.read()<0.18){//dokud jsi příliš blízko stěny
       robot.Read(); //synchronizace struktur
  }
 pp.SetSpeed(0,0);//zastav případné točení
 dist=0;//vynuluj ujetou vzdálenost
 while ((dist<0.04) &&(ir.read()>0.18) //ujeď vice než 4cm,drž se ode zdi
        robot.Read();
        dist = sqrt((target.px - pp.GetXPos())*
                (target.px - pp.GetXPos()) +
                (target.py - pp.GetYPos())*
                (target.py – pp.GetYPos())); //spočítej ujetou vzdálenost
  }
 }
```

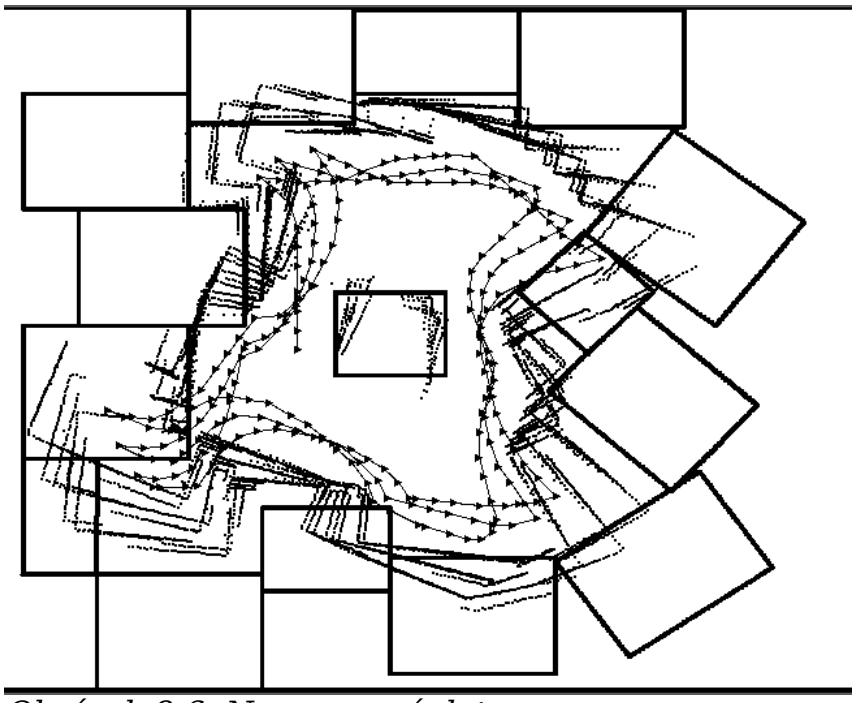
8 Výsledky

Pro porovnání BOR filtr algoritmu a MCL algoritmu jsem použil stejné měření. Testovací prostředí bylo postaveno z kartonových krabic na povrchu hřiště pro robotickou soutěž Eurobot. Krabice byly náhodně rozmístěny tak, aby vytvořily uzavřený prostor s dostatkem rohu, které robot hledal jako objekty. Skutečný prostor vypadal přibližně jako na obrázku 8.1. Pro výsledné testování byla použita data z průjezdu 3 kol algoritmem pro sledování stěny. Přibližná skutečná trasa jednoho kola je na obrázku 8.1.



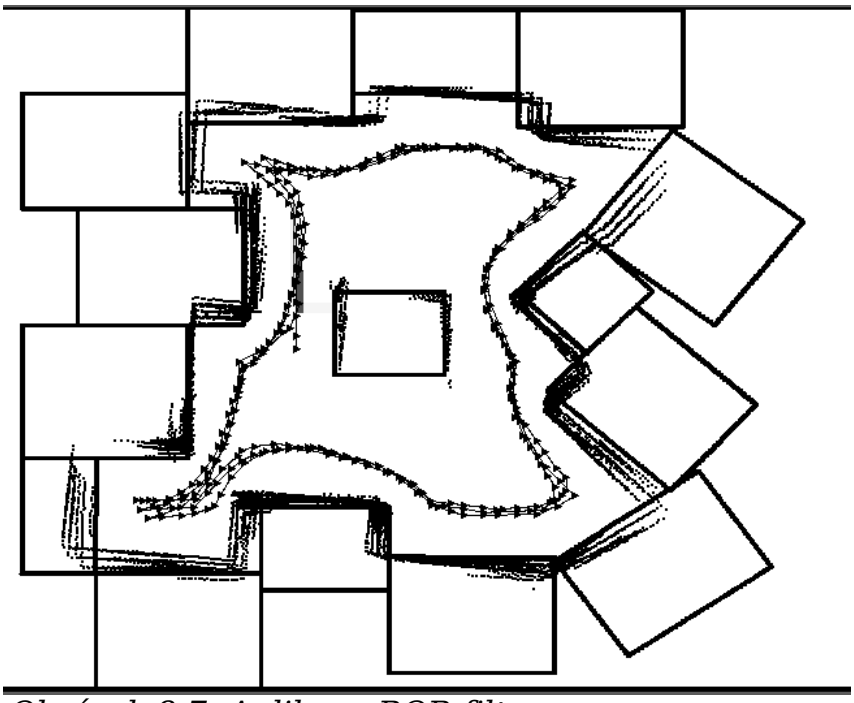
Obrázek 8.5: Testovací prostředí se znázorněním trasy

Na obrázku 8.2 je pak znázorněn nefiltrovaný výstup měření. Je vidět že se odhadovaná pozice robota stále více vzdalovala od skutečné pozice. Do obrázku jsou zakresleny všechny provedené scanny, které vratili hodnotu kratsi než 280mm. Scanny nad 280mm byly považovány za měření mimo dosah čidla. Robot během jednoho kola provedl asi 100 scanů. Každý scann obsahuje 90 měření vzdálenosti vzájemně pootočených vždy po 2 stupních.



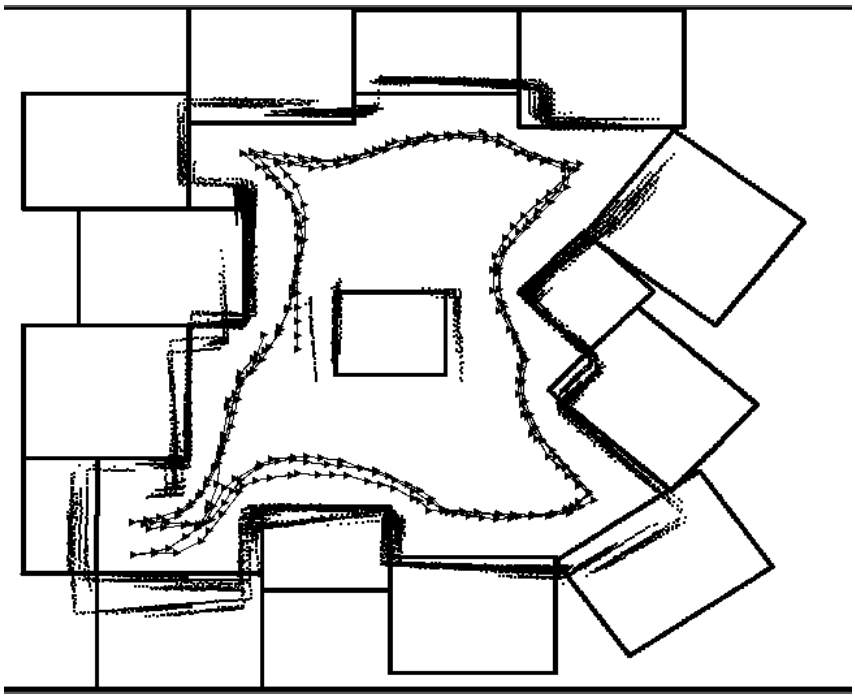
Obrázek 8.6: Neupravená data

Na dalším obrázku 8.3 je zobrazeno stejné měření po zpracování pomocí BOR filtru. Jako objekty byly použity rohy, kde za roh se považuje průsečík dvou přímek v jednom scannu. Bor filtr na těchto datech dokázal zpřesnit lokalizaci do té míry, že robot i po třech kolech chyba odhadu nerostla.



Obrázek 8.7: Aplikace BOR filtru

Na obrázku 8.4 je pak zobrazeno zpracování stejných dat pomoci MCL algoritmu. MCL algoritmus také na těchto datech udržel chybu dostatečně malou pro úspěšnou lokalizaci robota.



Obrázek 8.8: Aplikace MCL

Celkově jsem oba algoritmy ozkoušel na šesti sadách naměřených dat. Tento počet je příliš malý pro vyřčení ověřených výsledků. Z této malé skupiny se zdá že BOR filtr dosahuje při lokalizaci v takovýchto podmínkách lepší přesnosti, ale pokud se jeho chyba začne zvětšovat, nedokáže ji už napravit a velice rychle se úplně ztratí. Naproti tomu MCL algoritmus má v průběhu výpočtu většinou větší chybu, ale pouze v jednom případě došlo k úplnému ztracení. Byl to případ kdy robot udělal velikou chybou v odometrii když se zachytil ovládací kabel.

9 Závěr

Zaměřili jsem se v této práci na důkladné prostudování a vysvětlení postupů a algoritmů pro řešení SLAM problému. Protože použití Kalmanova filtru je nejrozšířenějším přístupem, bylo mu zde věnováno nejvíce prostoru. Jeho důsledný rozbor navíc vede k lepšímu pochopení vazeb, které přinášejí náročnost při řešení simultánní lokalizace a mapování. Použití rozšířeného Kalmanova filtr je řešením SLAM problému. Toto řešení je ovšem velice náročné na výpočetní výkon a pamětové nároky. Proto jsme pro skutečné pokusy použili méně náročné algoritmy. Našim cílem bylo ukázat, že se za použití těchto algoritmů dá řešit SLAM problém v reálném prostředí. Dále jsme ukázali, že pro úspěšnou lokalizaci v neznámém prostředí může robot využít i výrazně levnější a dostupnější čidla než je většinou používaný laserový scanner.

Velkým problémem při této práci se ukázala potřeba pracovat na skutečném robotu. Vývoj robota pro tuto zdánlivě nenáročnou aplikaci se ukázal být velice časově náročný. Pro další případné pokračování a rozšíření této práce se zdá být vhodná cesta použít dosud naměřená data, popřípadě průběžně ověřovat další pro odladění věrného chování modelu robota v simulátoru. Další pokusy by pak už mohly probíhat pouze v simulátoru, čímž by se práce na nich výrazně zjednodušila.

Použitá literatura

[1] : Mikael Bliksted Larsen (2006), The Autonomous Redundant Navigation System of an AUV for Mine Counter Measures.
[2]: Feature detection, http://en.wikipedia.org/wiki/Feature_detection
[3]: ActiveVision, http://www.robots.ox.ac.uk/ActiveVision/Publications/index.html
[4]: Michael Csorba (1997), Simultaneous Localisation and Map Building.
[5]: Jose E. Guivant (2002), Efficient Simultaneous Localization and Mapping in Large Enviroments.
[6]: H.R.Everet (1995), Sensors for Mobile Robots: Theory and Application. AK Peters, Ltd., 1568810482
[7] : M.W.M.G. Dissanayake, P.Newman, S. Clark, H.F. Durrant-Whyte, M. Csorba (2006), A Solution to the Simultaneous Localisation and Map Building (SLAM) Problem.
[8]: M. Zahradník, L. Motl (2002), Pěstujeme lineární algebru. Karolinum,
[9] : J.E. Guivant, E.M. Nebot (2001), Optimization of the simultaneous localization and map-building algorithm for real time implementation. 242-257
[10] : Dieter Fox, sebastian Thrun, Wolfram Burgard, Frank Dellaert (2000), Sequential Monte Carlo Methods in Practice.
[11]: Nabíječ Li Ion, http://www.zajic.cz/nablion/nablion.htm
[12] : Andrew J. Davison, Walterio Mayol and David W. Murray (2003), Real-Time Localisation and Mapping with Wearable Active Vision.
[13] : Jose Guivant, Eduardo Nebot, Stephan Baiker (2006), Autonomous Navigation and Map building Using Laser Range Sensors in Outdoor.
[14]: Player/Stage robotic framework, http://playerstage.sourceforge.net/