VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

NAVIGACE MOBILNÍCH ROBOTŮ

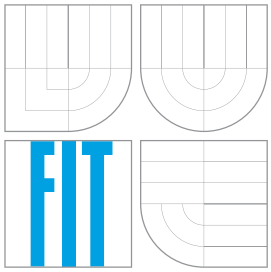
DISERTAČNÍ PRÁCE
PHD THESIS

AUTOR PRÁCE
AUTHOR

Ing. JAROSLAV ROZMAN

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

NAVIGACE MOBILNÍCH ROBOTŮ

MOBILE ROBOT NAVIGATION

DISERTAČNÍ PRÁCE
PHD THESIS

AUTOR PRÁCE — Ing. JAROSLAV ROZMAN
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. FRANTIŠEK V. ZBOŘIL, CSc.
SUPERVISOR

BRNO 2011

Abstrakt

Mobilní robotika je v posledních letech velice diskutované a rozšířené téma. Souvisí to především se stále se zdokonalující výpočetní technikou, která tak umožňuje vyvíjet stále složitější a dokonalejší roboty. Cílem tohoto snažení je vytvořit robota, schopného se autonomně pohybovat ve zvoleném prostředí. Pro tento úkol je nutné, aby si robot vytvořil mapu, ve které bude svůj pohyb plánovat. V současné době se za standard v mapování považují pravděpodobnostní algoritmy založené na metodě SLAM.
Tato disertační práce se zabývá návrhem plánovacího algoritmu právě pro metodu SLAM. Popisuje plánování pohybu pro robota vybaveného dvojicí kamer, tzv. stereokamerou, umístěnou na pohyblivé platformě. Plánování pohybu je navržené s ohledem na použití algoritmů, které budou v obraze ze stereokamery vyhledávat význačné body a z těch pak pomocí triangulace tvořit mapu, nebo také model prostředí.
Přínos práce by se dal rozdělit do tří částí. V první je popsán způsob vyznačování plochy, ve které pak bude robot plánovat svůj pohyb. Druhá část se zabývá samotným plánováním pohybu robota v této mapě. Bere při tom v úvahu vlastnosti algoritmu SLAM a snaží se tedy toto plánování navrhnout tak, aby vytvořená mapa byla co nejpřesnější. Ve třetí části je pak popsán pohyb platformy, která nese kamery. V této části se využívá toho, že robot může svými kamerami sledovat i jiná místa, než jsou ta ve směru jeho pohybu. To mu umožní prozkoumat mnohem větší prostor bez přílišné ztráty informace o své přesné poloze.

Abstract

Mobile robotics has been very discussed and wide spread topic recently. This due to the development in the computer technology that allows us to create better and more sophisticated robots. The goal of this effort is to create robots that will be able to autonomously move in the chosen environment. To achieve this goal, it is necessary for the robot to create the map of its environment, where the motion planning will occur. Nowadays, the probabilistic algorithms based on the SLAM algorithm are considered standard in the mapping in these times. This Phd. thesis deals with the proposal of the motion planning of the robot with stereocamera placed on the pan-and-tilt unit. The motion planning is designed with regard to the use of algorithms, which will look for the significant features in the pair of the images. With the use of the triangulation the map, or a model will be created.
The benefits of this work can be divided into three parts. In the first one the way of marking the free area, where the robot will plan its motion, is described. The second part describes the motion planning of the robot in this free area. It takes into account the properties of the SLAM algorithm and it tries to plan the exploration in order to create the most precise map. The motion of the pan-and-tilt unit is described in the third part. It takes advantage of the fact that the robot can observe places that are in the different directions than the robot moves. This allows us to observe much bigger space without losing the information about the precision of the movements.

Klíčová slova

robotika, SLAM, Delaunayho triangulace, částicový filtr, plánování pohybu, výběr cílů, aktivní sledování

Keywords

robotics, SLAM, Delaunay Triangulation, Constrained Delaunay Triangulation, particle filter, path-planning, goal point selection, active vision

Citace

Jaroslav Rozman: Navigace mobilních robotů, disertační práce, Brno, FIT VUT v Brně, 2011

Navigace mobilních robotů

Prohlášení

Prohlašuji, že jsem tuto disertační práci vypracoval samostatně pod vedením pana doc. Ing. Františka V. Zbořila. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jaroslav Rozman
1. července 2011

Poděkování

Chtěl bych poděkovat svému školiteli, panu docentu Zbořilovi, za jeho vedení při mé práci a hlavně během celého mého studia. Dále bych chtěl poděkovat Vladimírovi a Zdeňkovi za jejich neutuchající odpovědi na moje neutuchající otázky. A v neposlední řadě můj dík patří rodičům, za jejich podporu během mých celých studijních let.

Obsah

Kapitola 1

Úvod

Robotika je poměrně mladé vědní odvětví rozvíjející se v několika posledních desetiletích. Kombinuje v sobě mnoho jiných oborů, jako je teorie řízení, modelování a simulace, počítačové vidění a mnohé další. Její rozvoj v posledních letech souvisí především se stále dokonalejšími počítači a s miniaturizací veškerých elektronických zařízení, používaných v robotech.

Samotná robotika by se dala rozdělit do dvou oblastí [66]. První oblastí jsou průmysloví roboti, tzn. veškerá robotická ramena a manipulátory. Druhá oblast je oblast experimentální nebo také kognitivní robotiky. Tam patří robotické systémy, vybavené určitým stupněm inteligence. Jsou to především mobilní roboti, různá robotická vozítka od automatických vysavačů až po roboty zkoumající vzdálené planety. Patří sem také humanoidní roboti.

Zatímco roboti v průmyslu jsou široce rozšíření, a např. automobilový průmysl by bez nich mohl jen obtížně existovat, roboti mobilní, i přes jejich široké možnosti uplatnění, nedosahují ani zdaleka takových počtů. Je to dáno především tím, že roboti průmysloví jsou mnohem jednodušší a podstatně ulehčují lidem práci. Požadujeme po nich totiž většinou pouze velmi jednoduché úkoly, které jsou velice jednotvárné, jako vzít součástku a přemístit ji jinam, atd. Tyto úkony nevyžadují žádné komplikované myšlení a drahé senzory, bohatě si vystačí se senzory natočení ramen. Robotická práce je rychlá, přesná a na rozdíl od lidí se roboti neunaví.

Mobilní roboti představují komplexní zařízení, kde je nutná spolupráce odborníků z řady profesí, ať už se jedná o návrh a stavbu vlastní mechanické části robota, o vytvoření jeho softwarového vybavení, výběr vhodných procesorů, senzorů, kamer a mnoha dalších věcí, jako je už zmíněné počítačové vidění nebo zpracování dat ze senzorů. I průmysloví roboti obsahují mnoho takovýchto částí. Velký rozdíl je však v tom, že průmysloví roboti z velké části vykonávají pouze předem danou posloupnost pohybů, např. stříkání karoserie, zatímco mobilní roboti už musejí mít něco jako vlastní "inteligenci", aby se mohli volně pohybovat ve svém okolí, aniž by pro ně, nebo pro sebe, byli nebezpeční. A právě ona "inteligence"je důvod menšího rozšíření mobilních robotů. Reálný robot v reálném prostředí se totiž musí potýkat se spoustou problémů, které jejich průmysloví kolegové v prostředí jim přizpůsobeném nemají. Především jde o nedokonalost senzorů, stále malý výpočetní výkon počítačů, ale hlavně o to, že takový robot se pohybuje v dynamicky se měnícím prostředí, plném lidí, aut a třeba i dalších robotů a musí být schopen se v tomto prostředí nejenom sám orientovat, ale i plnit dané úkoly.

Pokud má robot plnit zadané úkoly, je nutné, aby se mohl samostatně pohybovat ve zvoleném prostředí. Potřebuje mapu, ve které se bude orientovat a ve které bude plánovat svůj pohyb. Mapu mu může poskytnout člověk, ale vhodnější je, aby si ji za pomoci svých senzorů vytvořil sám. Senzory, používané pro tvorbu mapy, jsou především ultrazvukové so-

nary, laserové scannery a kamery. Dále se pro určení pozice robota používá tzv. odometrie, což je zjištění polohy robota na základě překonané vzdálenosti. Tři nejčastější senzory pro zjišťování ujeté vzdálenosti jsou enkodéry, akcelerometry a gyroskopy. Ty robotovi umožní zjistit novou pozici a aktualizovat novými měřeními mapu. Vzhledem k různé výpočetní náročnosti při zpracování dat se nejdříve začaly používat sonary, pak laserové scannery a nakonec kamery. Každý z těchto senzorů má své specifické vlastnosti, výhody a nevýhody. Jednu věc ale mají všechny společnou a tou je šum. Ten se může projevovat nejen tak, že je správná hodnota porušena o nějakou malou hodnotu, ale i tak, že např. vlivem prostředí dojde k naprosto chybnému měření, nebo že se místo změření vzdálenosti od robota k překážce změří vzdálenost k právě procházejícímu člověku. V případě neexistence těchto chybných měření by všechna naměřená data byla přesná, což by vyústilo v tvorbu téměř dokonale přesné mapy a tak ve zbytečnost celého jednoho odvětví robotiky - simultánní lokalizace a mapování (SLAMu), viz kapitola 2.8.

1.1 Motivace a cíle disertační práce

Předložená práce se zabývá tvorbou mapy a plánováním pohybu robota. Jejím cílem je zbavit robota všech zmíněných senzorů a ponechat mu pouze dvojici kamer, tzv. stereokameru. Tím se robot svým vnímáním okolí přiblíží alespoň částečně člověku (máme na mysli pouze zrak, samozřejmě neuvažujeme ostatní smysly jako je sluch či hmat nebo dokonce čich a chuť, ty by se výše zmíněnými senzory nedaly nahradit).

Cílem disertační práce je navrhnout a otestovat takové metody, které robotovi umožní tvorbu plně 3D mapy za použití pouze stereokamery a zároveň mu dovolí se ve zmapovaném prostředí autonomně pohybovat. Senzory, potřebné pro určení robotovy pozice, tj. enkodéry a akcelerometry, mohou být totiž nahrazeny kamerou, stejně jako senzory pro určení vzdálenosti objektů. Běžná odometrie je tak nahrazena vizuální odometrií a vzdálenosti, které se běžně zjišťují ultrazvukovými nebo laserovými měřiči, se zjistí z rozdílné polohy objektů v obrazu z dvojice kamer.

Cíle disertační práce jsou následující:

1. Navrhnout vhodnou reprezentaci mapy pro 3D prostředí.
2. Navrhnout plánování pohybu robota v prostředí.
3. Navrhnout vhodný způsob prozkoumávání neznámého prostoru.

1.2 Struktura práce

Celá práce se skládá z pěti kapitol. V úvodu je čtenář uveden do problematiky a jsou vytyčeny cíle práce.

Druhá kapitola obsahuje seznámení s historií mapování v robotice a stručně jsou popsány nejpoužívanější senzory. Následuje přehled teorie z oblasti robotiky, na které jsou založeny algoritmy v současnosti používané v oblasti SLAMu a které jsou důležitým východiskem pro další práci.

Ve třetí kapitole je popsán model kamery a metody, důležité pro tvorbu modelu prostředí.

Čtvrtá kapitola obsahuje vlastní práci. Je v ní popsán způsob, jak v mapě vyznačovat průjezdnou oblast. Dále jak v této oblasti plánovat pohyb robota tak, aby jeho způsob pohybu byl co nejvhodnější pro algoritms SLAM. Poslední část je věnována plánování otáčení stereokamer umístěných na pohyblivé platformě. Díky tomu může robot prozkoumat větší prostor, aniž by se musel otočit celý. Na závěr kapitoly jsou prezentovány provedené experimenty.

Poslední, pátá kapitola, obsahuje shrnutí celé práce a její možný vývoj do budoucna.

Kapitola 2

Teoretický úvod - robotika

Prvním robotem s umělou inteligencí schopným plnit jednoduché úkoly, byl robot Shakey, vyvinutý v SRI International v letech 1966 až 1972. Dalšími významnými roboty je např. robotický pes AIBO vyráběný v letech 1999 až 2006 firmou SONY a humanoidní robot ASIMO vyrobený japonskou Hondou v roce 2000.

Pokud se obrátíme do vesmíru, nejznámějšími vesmírnými roboty (nazývanými rovery) byla vozítka Sojourner a Spirit a Oportunity, která přistála na Marsu v roce 1997, resp. v roce 2004. V případě Spiritu a Oportunity byla mise obzvláště úspěšná, protože trvala od jejich přistání v roce 2004 s přestávkami až do nedávné doby. Zatím posledním roverem, u kterého se plánuje přistání na Marsu v roce 2012, je Mars Science Laboratory.

2.1 Současný stav problematiky

Za počátek mapování v mobilní robotice bychom mohli považovat polovinu osmdesátých let a práce Elfese a Moravce [22], [50] , kteří jako první vytvořili matematický model pro sonary a pomocí jednoduchých algoritmů umožnili vytvářet mapy malých prostředí, např. místností. Z tvorby jednoduchých map se posléze vytvořila oblast tzv. SLAMu. Robot si vytváří mapu neznámého prostředí a zároveň se v této mapě lokalizuje, tj. určuje svou pozici v této právě vytvořené mapě.

Problematiku SLAMu bychom tak mohli rozčlenit na dvě části:

- lokalizace
- tvorba mapy

Postupem času se ukázalo, že vzhledem k nepřesnosti senzorů je vhodné tuto nepřesnost začlenit do mapovacích algoritmů jako pravděpodobnost správné funkce senzoru. Proto jsou současné nejlepší lokalizační a mapovací algoritmy pravděpodobnostní a vycházejí především z Bayesovy pravděpodobnosti a z Bayesových filtrů.

Jak už bylo uvedeno, dá se SLAM rozdělit na dvě části. Jednou z nich je lokalizace. V této oblasti se používají především dva pravděpodobnostní přístupy, event. jejich kombinace. Prvním z nich je Kalmanův filtr a jeho různé modifikace, jako např. verze pro nelineární systémy Rozšířený Kalmanův filtr (Extended Kalman Filter) a druhým je Grid-Based lokalizace a hlavně Monte-Carlo lokalizace (MCL), používající částicový filtr.

Druhou částí SLAMu je tvorba mapy. Vzhledem k nepřesnosti senzorů a malému výpočetnímu výkonu se dříve konstruovaly především mapy dvojrozměrné. V současné době

se s rozvojem kamer a 3D scannerů a hlavně se stále dokonalejšími algoritmy začíná přecházet od tvorby 2D map k mapám v plně 3D prostředí. Jedním z důvodů proč jsou 3D mapy, nebo spíše už 3D modely využívané, je možnost jejich prohlížení i lidmi. Využití nalézají např. při mapování nepřístupných oblastí, jako jsou doly [91] nebo jeskyně [26], v případě záchranných operací, ale také např. při automatické tvorbě 3D modelů měst v aplikacích jako je Google Earth a podobné. Pro samotné roboty je 3D mapa důležitá, protože díky ní jsou schopni autonomního pohybu a manipulace s 3D objekty. Z map statických interiérů budov se postupně přechází na mapování dynamicky se měnícího prostředí, rozsáhlých exteriérů a také na spolupráci více robotů při mapování.

Každý robot se skládá z několika základních částí:

- šasi robota - tvoří vlastní kostru robota, na něm jsou umístěny všechny ostatní části
- motory - slouží k pohonu robota, většinou pohání buď kola, nebo pásy
- baterie - napájí všechny elektronické části
- řídicí počítač - řídí pohyb robota, může být buď součást robota, nebo je umístěn mimo něj, pak je nutná ještě vzájemná komunikace
- senzory - slouží k získávání informací o okolí a o robotovi
- elektronika - slouží k propojení motorů, baterií, senzorů a řídicího počítače

Nás budou zajímat především senzory, proto si je v následující kapitole blíže popíšeme.

2.2 Senzory

Dá se říct, že jsou jednou z nejdůležitějších součástí robota. Používají se hlavně pro měření vzdálenosti objektů od robota, ale také pro zjišťování vlastního stavu robota, např. otočení koleček, natočení robota, stav baterií, atd.

Enkodéry

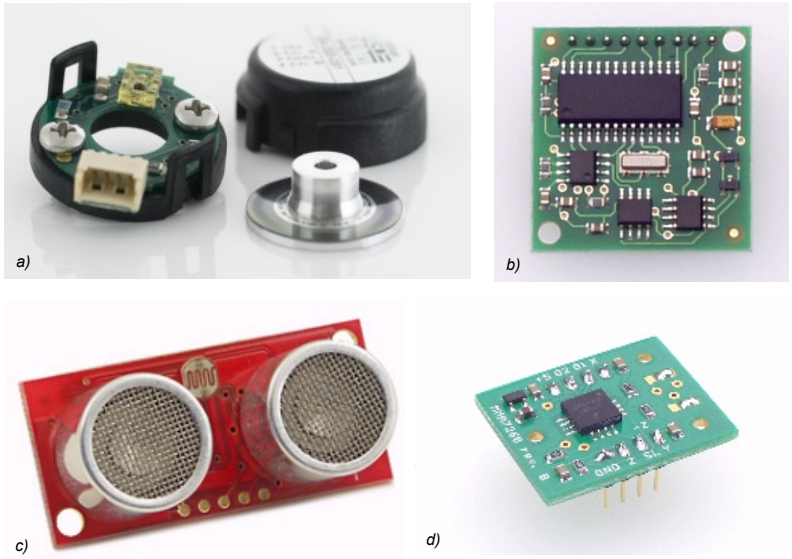
Enkodéry se na robotech používají především k měření počtu otočení koleček. Bývají umístěny buď na kolečku anebo přímo na hřídeli motorku. Fungují většinou na principu přerušování optického paprsku rotujícím diskem s miniaturními otvory. Jedná se o celkem jednoduchý způsob měření ujeté vzdálenosti. Nevýhoda však spočívá v tom, že pokud se robot pohybuje po kluzkém povrchu a jeho kolečka prokluzují, enkodér udává špatnou informaci o ujeté vzdálenosti.

Akcelerometry

Akcelerometry spolu s gyroskopy patří do tzv. inerciálních senzorů. Měří zrychlení a z něj pak počítají polohu objektu. Akcelerometry tak mohou být použity, stejně jako enkodéry, k měření polohy robota, přičemž na ně nepůsobí už zmíněný prokluz koleček.

Taktilní senzory

Jedním ze základních senzorů robota jsou tzv. taktilní senzory. Ty poskytují informaci o kolizi robota s překážkami, což znamená, že pro svou funkčnost potřebují přímý fyzický



Obrázek 2.1: Ukázka typických senzorů používaných nejčastěji na robotech. a) je digitální enkodér, b) kompas, c) ultrazvukový sonar, d) tříosý akcelerometr. Všechny uvedené senzory jsou použity na robotu, vzniklém na škole v rámci projektu FRVŠ.

kontakt s překážkou. Používají se především taktilní antény (ang. whiskers), taktilní nárazníky a pole spínačů. Vzhledem k nutnosti střetu robota s překážkou, což může být při vyšších rychlostech pro robota nebo i pro překážku nebezpečné, se používají spíše jako poslední záchrana při selhání ostatních senzorů. Kromě toho, např. whiskery umístěné kolem robota, znemožňují jeho položení na bok a také může často docházet k jejich špatné funkci. Přes všechny tyto nevýhody je vhodné, aby robot měl nějaké taktilní senzory, např. pole spínačů. To se používá např. u velmi jednoduchých robotů, jako jsou roboti pro robotické sumo.

Sonary

Sonary patří vzhledem ke své velikosti, hmotnosti a ceně k nejpoužívanějším senzorům už od počátku robotiky. Řadí se spolu s laserovými scannery a kamerami mezi proximitní senzory, tzn. pro měření vzdálenosti nepotřebují přímý kontakt s překážkou. Pracují, podobně jako lasery, na principu měření doby letu vyslaného signálu. Zde se ovšem jedná o ultrazvukové vlny. Ze známé rychlosti šíření zvuku pak vypočítají vzdálenost k překážce. Jejich největší nevýhoda vyplývá z podstaty jejich funkce. Ultrazvukový signál se totiž od svého zdroje šíří ve tvaru kužele. Navrácená hodnota tedy pouze říká, že překážka leží v udané vzdálenosti. Neříká však nic o její přesné poloze. Mezi další nevýhody sonaru patří jeho relativně velká nepřesnost a malá rychlost měření. I přes uvedené nevýhody je to jeden z nejrozšířenějších senzorů a to především pro svou jednoduchost a snadnou cenovou dostupnost.

Laserové scannery

Dalšími používanými senzory jsou laserové scannery. Stejně jako sonary fungují na principu měření doby letu, tentokrát laserového paprsku. Jejich hlavní výhoda je přesnost a rychlost, s jakou jsou schopny měřit. Protože používají laserový paprsek, který má téměř konstantní průřez, odpadá největší nevýhoda sonarů a tou je kuželová oblast měření. Jsou

tak schopny zjistit polohu předmětu s velkou přesností. Nevýhodou je jejich velikost, velká hmotnost a vyšší cena (třeba až 5kg a 100 tis. Kč oproti několika desítkám gramů a asi 1 tis. Kč za sonar). Vzhledem k těmto faktům se používají především u větších a dražších robotů.

Kamery

Se zvyšujícím se výkonem počítačů se do popředí zájmu začínají dostávat kamery. Jejich výhody jsou zřejmé - nízká cena a to i oproti sonarům, nízká hmotnost, malá velikost a také především to, že na rozdíl od sonaru a laseru je možné z jejich obrazu vytvořit 3D mapu prostředí, a to s velkým rozlišením. Jejich výhoda je však zároveň i jejich nevýhodou. Zatímco při použití sonaru máme max. několik desítek naměřených hodnot, při použití laseru několik set, u kamery jsou to statisíce až miliony hodnot/pixelů, které musíme zpracovat v reálném čase, což je i přes vzrůstající rychlost procesorů problém.

Pokud má být kamera užitečná, musí co nejpřesněji detekovat předměty. K tomu je potřeba použít co největší rozlišení, což už může být výpočetně náročné. Dále je nutné, aby kamera měla co nejlepší světelnost, protože pak budou jednotlivé snímky exponovány co nejrychleji a odstraní se tak rozmazání obrazu, které se objevuje při dynamickém pohybu robota, např. při jeho otáčení. Stejně jako lasery a sonary jsou i kamery ovlivněny šumem a to především v podobě nepřesného určení polohy význačných bodů. Pokud by totiž byly snímky rozmazané, nedala by se přesně určit poloha význačných bodů a nepomohlo by tak ani velké rozlišení kamery. Z nepřesné polohy bodů ve snímku by se tak vypočítala i nepřesná poloha bodů v 3D prostoru.

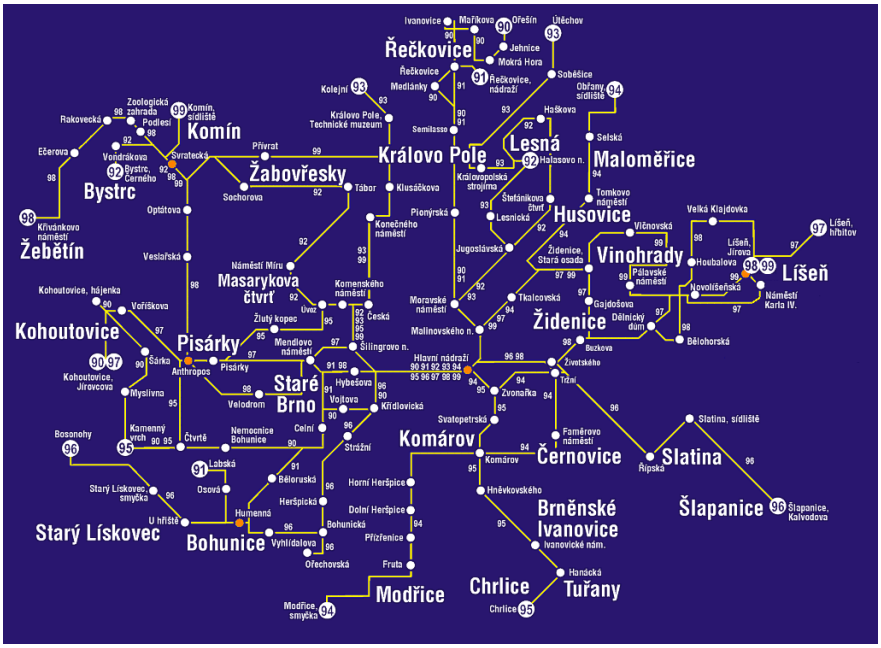
Je ovšem celkem přirozené, že se kamery začínají stávat hlavním senzorem na robotech. Roboti se tak přibližují lidem, pro něž je zrak nejdůležitějším smyslem, díky němuž získávají až 90% informací o okolí.

2.3 Vnitřní reprezentace prostředí - mapy

Pokud chceme, aby mobilní robot plnil složitější úkoly, je nutné ho vybavit nějakou reprezentací vnějšího prostředí, tzn. mapou, pomocí které se bude orientovat v reálném světě. Mapy pro mobilní, ale i pro stacionární roboty se ovšem částečně liší od map používaných lidmi. Mapy, používané roboty, se v podstatě dělí na dva druhy, na mapy topologické, které určují vzájemnou polohu objektů a na mapy metrické, které jsou bližší mapám používaným lidmi. Za zvláštní druh metrických map bychom také mohli považovat mapy *konfiguračního prostoru* robota (Configuration Space) [66], v nichž lze určit bod, který přesně odpovídá poloze robota v prostoru.

2.3.1 Topologické mapy

Topologické mapy (obr. 2.2) na rozdíl od metrických neurčují přesnou polohu objektů, ale vzájemné vztahy mezi jednotlivými objekty, jejich propojenost a sousednost. Pro lepší orientaci v mapě jsou jednotlivé uzly nebo objekty označeny symbolickými jmény. Pro použití v robotice je potřeba z prostředí extrahovat takové vlastnosti, které by mohly být použity v mapě coby uzly, tzn. algoritmus musí rozpoznat např. rohy, T-spoje chodeb, dveře atd. Jednou z nejznámějších topologických map je mapa londýnského metra.



Obrázek 2.2: Příklad topologické mapy - plán rozjezdů brněnské MHD. Důležitější než přesná poloha a tvar trasy jsou vzájemná propojení mezi jednotlivými zastávkami.

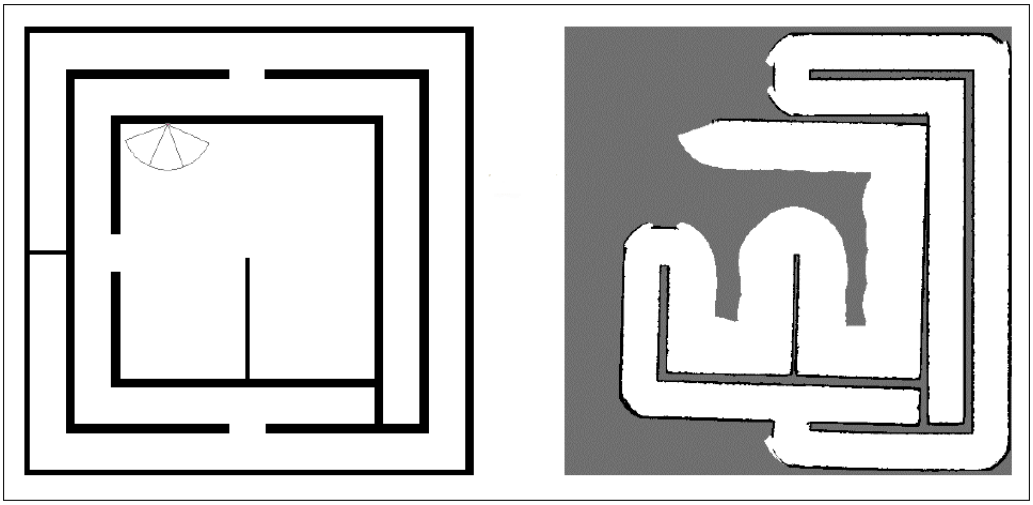
2.3.2 Metrické mapy

Metrické (nebo také senzorické) mapy jsou založeny na kartézské soustavě souřadnic, tzn. každý objekt je zadán souřadnicemi v daném souřadném systému. Metrické mapy obsahují buď přímá měření ze senzorů, nebo mohou být fúzí různých měření od různých senzorů. Nejčastějším představitelem těchto druhů map je tzv. mřížka obsazenosti (obr. 2.3). V tomto případě je prostor rozdělen mřížkou na jednotlivé buňky a každé buňce je přiřazena hodnota v intervalu $\langle 0, 1 \rangle$ určující její obsazenost či volnost. Nevýhodou tohoto přístupu je velká paměťová náročnost při ukládání mřížky v počítači a z toho plynoucí nemožnost vytvářet mřížky pro velká území. Jedním ze způsobů, jak tuto nevýhodu obejít, může být např. slučování buněk se stejnou hodnotou do větších celků nebo použití stromů, jako je třeba v [26]. I tak je ale velice nepraktické vytvářet touto metodou např. 3D modely větších oblastí.

V praxi se používá několik druhů přístupů, jak získat hodnoty obsazenosti pro jednotlivé buňky. Nejčastější je pravděpodobnostní přístup, používající Bayesův vzorec pro podmíněné pravděpodobnosti, který bude vysvětlen dále. Dalšími možnostmi jsou pak použití Dempster-Shaferovy teorie [55] a HIMM (Histogrammic in Motion Mapping) algoritmu [9].

2.4 Filtry

Filtry nám umožňují ze zašuměných dat vypočítat co nejpřesnější hodnoty a jak už bylo uvedeno, matematickým základem pro pravděpodobnostní algoritmy je Bayesův filtr. Pomocí jeho praktické implementace můžeme spočítat nejpravděpodobnější polohu robota a nejpravděpodobnější vzdálenosti k překážkám, z čehož můžeme následně vytvořit mapu okolí, která bude rovněž pravděpodobnostní. Bayesův filtr pracuje ve dvou fázích a to predikce a korekce. Ve fázi predikce požadovanou veličinu vypočítáme na základě modelu a



Obrázek 2.3: Simulace robota zkoumajícího neznámý prostor (vlevo). Robotova pozice je těsně vedle zdi, tři kruhové výseče reprezentují měření jeho třemi sonary. Na obrázku vpravo je doposud vytvořená metrická mapa. Šedá barva znázorňuje buňky, jejichž pravděpodobnost obsazenosti je 0,5, černou barvou jsou zobrazeny buňky, kde je pravděpodobnost obsazenosti blízká 1 a bílou ty buňky, kde je pravděpodobnost obsazenosti blízká 0. Velikost buňky v tomto příkladu odpovídá velikosti pixelu.

ve fázi korekce její hodnotu upřesníme na základě aktuálních měření.

Pro výpočet je potřeba zavést značení, které bude určovat polohu robota, jeho měření, řízení a vytvořenou mapu. Stav robota se bude značit x, tzn. v případě, že stavem robota je jeho poloha (souřadnice x, y a úhel natočení φ) bude $x = (x, y, \varphi)$. Kdybychom chtěli pohyb robota rozšířit do 3D prostoru, tzn. neuvažovali bychom jen jeho pohyb po vodorovné podlaze, ale i např. po nakloněné rovině či v terénu, nebo by robot byl schopen pohybu ve všech směrech, přidali bychom třetí kartézskou souřadnici z a dvě úhlové souřadnice ψ a ω.

Robot může měnit svůj stav pomocí svých akčních členů nebo může zjišťovat informace o prostředí pomocí sady měření. Ta se bude označovat z. Měření v čase t tedy bude z_t. Značení

$$z_{t_1:t_2} = z_{t_1}, z_{t_1+1}, z_{t_1+2}, \ldots, z_{t_2} \tag{2.1}$$

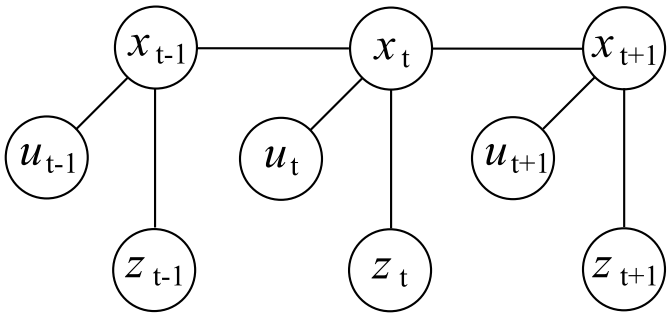
odpovídá sadě měření od času t_1 po čas t_2 ($t_1 < t_2$).

Řízením se bude označovat změna stavu robota v prostředí. Většinou se bude jednat o pouhou změnu polohy robota, protože nepředpokládáme, že by robot měl nějakou možnost měnit svoje okolí (např. byl vybaven ramenem). Změna polohy se zjišťuje čtením z robotových vnitřních senzorů, tzn. z enkodérů, akcelerometru nebo GPS. Řízení budeme označovat u. Znační u_t bude odpovídat řízení v intervalu $(t-1; t)$. Stejně jako v předchozím případě, značení

$$u_{t_1:t_2} = u_{t_1}, u_{t_1+1}, u_{t_1+2}, \ldots, u_{t_2} \tag{2.2}$$

odpovídá sadě řízení od času t_1 po čas t_2 ($t_1 < t_2$).

V podstatě všechny současné state-of-art systémy pro určování polohy, tvorbu mapy a řízení robota jsou založeny na pravděpodobnostech a ve velké míře se používá Bayesova podmíněná pravděpodobnost. Značení $p(z_t \mid x_t)$ značí pravděpodobnost, že robot v pozici x_t provedl měření z_t. Tato pravděpodobnost se nazývá *pravděpodobnost měření* (ang.



Obrázek 2.4: Schéma Bayesovy sítě, která reprezentuje změny stavu x v čase $t-1$ až $t+1$ s řízením u a měřením z.

measurement probability). Analogicky značení $p(x_t \mid x_{t-1}, u_t)$ značí pozici robota v čase t, když víme, že v čase $t-1$ byl robot v pozici x_{t-1} a bylo provedeno řízení u_t. Nazývá se *pravděpodobnost změny stavu* (state transition probability). V literatuře, např. [87], se pravděpodobnost měření, resp. změny stavu také nazývá *model měření*, resp. *model pohybu*.

Tyto dvě pravděpodobnosti popisují dynamický stochastický systém robota a jeho okolí. Stav v čase t závisí stochasticky na stavu v čase $t-1$ a na řízení u_t. Měření z_t v čase t závisí stochasticky na stavu v čase t.

Dalším důležitým pojmem je tzv. *míra důvěry* (ang. belief). Ta odpovídá robotově vnitřní znalosti o stavu prostředí. Robot totiž není schopen tento stav schopen zjistit přímo, ale pouze na základě dat ze svých měření.

$$\overline{bel}(x_t) = p(x_t \mid z_{1:t-1}, u_{1:t}) \tag{2.3}$$

$$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) \tag{2.4}$$

Zde $\overline{bel}(x_t)$ odpovídá víře, že robot je po provedení řízení u_t v pozici x_t ještě předtím, než provede měření z_t. Výpočet $bel(x_t)$ z $\overline{bel}(x_t)$ se nazývá aktualizace měření.

2.4.1 Bayesův filtr

Nejobecnější algoritmus pro výpočet míry důvěry (beliefu) je dán Bayesovým filtrem. Bayesův filtr počítá stav x dynamického systému podle dat ze senzorů. Např. podle dat ze sonarů a enkodérů určuje pozici robota. Podstatou filtru je odhadnout rozložení pravděpodobnosti přes stavový prostor v závislosti na datech ze senzorů.

Bayesův filtr se skládá ze dvou hlavních částí, v první se provede aktualizace řízení, této části se říká *predikce* a výsledkem je $\overline{bel}(x_t)$ a ve druhé části se provede aktualizace měření, takzvaná *korekce*, tzn. vypočte se $bel(x_t)$. Tyto dva kroky obsahují všechny algoritmy vycházející z Bayesova filtru. Pro výpočet je nutné znát data z řízení a z měření. Z těch se pomocí modelu řízení a měření vypočítají odpovídající pravděpodobnosti. Protože se jedná o rekurzívní algoritmus, je nutné znát i předchozí pravděpodobnost.

Odvození $p(x_t \mid z_{1:t}, u_{1:t})$ z $p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1})$ je podle [87] následující: Použijeme Bayesovo pravidlo

$$p(x_t \mid z_{1:t}, u_{1:t}) = \frac{p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) p(x_t \mid z_{1:t-1}, u_{1:t})}{p(z_t \mid z_{1:t-1}, u_{1:t})} \tag{2.5}$$

$$= \eta p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) p(x_t \mid z_{1:t-1}, u_{1:t}) \tag{2.6}$$

Protože řízení nemá na měření žádný vliv a samotné měření závisí jen na současné poloze robota, můžeme psát

$$p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) = p(z_t \mid x_t), \tag{2.7}$$

což předchozí výraz zjednoduší na

$$p(x_t \mid z_{1:t}, u_{1:t}) = \eta p(z_t \mid x_t) p(x_t \mid z_{1:t-1}, u_{1:t}) \tag{2.8}$$

Tuto rovnici můžeme přepsat na

$$bel(x_t) = \eta p(z_t \mid x_t) \overline{bel}(x_t) \tag{2.9}$$

$\overline{bel}(x_t)$ rozepíšeme jako

$$\overline{bel}(x_t) = \int p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) \mathrm{d}x_{t-1} \tag{2.10}$$

Protože nová poloha robota nezávisí na měření, ale jen na předchozí poloze a řízení, můžeme rovnici opět zjednodušit

$$\overline{bel}(x_t) = \int p(x_t \mid x_{t-1}, u_t) p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) \mathrm{d}x_{t-1} \tag{2.11}$$

Druhá část integrálu je podmíněná pravděpodobnost v minulém kroku

$$bel(x_{t-1}) = p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) \tag{2.12}$$

Rovnice Bayesova filtru tedy vypadá následovně:

$$bel(x_t) = \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t) bel(x_{t-1}) \mathrm{d}x_{t-1} \tag{2.13}$$

Pravděpodobnosti $p(x_t \mid x_{t-1}, u_t)$ resp. $p(z_t \mid x_t)$ jsou model pohybu, resp. model měření. Oba modely budou vysvětleny v dalších kapitolách.

Algoritmus 1 BayesFilter

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

1: **for** all x_t **do**
2: $\quad \overline{bel}(x_t) = \int p(x_t \mid x_{t-1}, u_t) bel(x_{t-1}) \mathrm{d}x_{t-1}$
3: $\quad bel(x_t) = \eta p(z_t \mid x_t) \overline{bel}(x_t)$
4: **end for**
5: **return** $bel(x_t)$

Bayesův filtr se dá implementovat mnoha různými způsoby, ale všechny se dají rozdělit do dvou kategorií.

První z nich jsou Gaussovy filtry, které reprezentují pravděpodobnost pomocí normálního, Gaussova, rozložení:

$$p(x) = det(2\pi\Sigma)^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \tag{2.14}$$

Hustota pravděpodobnosti $p(x)$ je charakterizována dvěma proměnnými, střední hodnotou μ a kovariancí Σ. Důležitou vlastností těchto filtrů je to, že jsou unimodální, tzn. mají jen jedno maximum. Tato vlastnost je charakteristická např. pro sledování pozice, kdy se robot s určitou pravděpodobností nachází na určitém místě a s klesající pravděpodobností v okolí tohoto místa. Příkladem těchto filtrů je např. Kalmanův filtr, v robotice velice populární.

Druhou kategorií jsou tzv. neparametrické filtry. Ty reprezentují pravděpodobnost pomocí konečného počtu hodnot, kde každá hodnota odpovídá místu ve stavovém prostoru. Kvalita reprezentace pravděpodobnosti závisí na počtu těchto hodnot. Existují dva druhy neparametrických filtrů. První dělí stavový prostor na konečný počet oblastí a reprezentuje pravděpodobnost histogramem. Druhý ji reprezentuje konečným počtem vzorků a nazývá se částicový filtr. Částicový filtr dosáhl v robotice, a nejen tam, obrovské popularity. Výhodou neparametrických filtrů oproti Gaussovým je to, že jsou multimodální, tzn. mají více maxim, takže mohou být použity nejenom ke sledování pozice, ale i na globální lokalizaci nebo problém "uneseného robota"(viz kapitola 2.6). Jejich další výhodou je snadná implementovatelnost.

2.4.2 Kalmanův filtr

Kalmanův filtr [35], pojmenovaný po svém autorovi Rudolfovi E. Kalmanovi, je jedním z nejznámějších a nejčastěji používaných nástrojů při filtrování zašuměných dat. Jeden z nejznámějších článků, kde je popsána i jeho rozšířená varianta (Extended Kalman Filter - EKF), je od autorů Welcha a Bishopa [94]. Další články na toto téma jsou například [57], [64]. Použití přímo v robotice, konkrétně na sledování pohybu robota je popsáno v [12]. Další článek na téma sledování pohybu robota, a to včetně zdrojových kódů pro matlab, je od Dana Simona [78].

V robotice je používání Kalmanova filtru velice rozšířeno a dá se říct, že žádný mobilní robot se bez něj neobejde. Proto bude vhodné, věnovat se mu trochu blíže.

Pro použití Kalmanova filtru je potřeba splnit čtyři podmínky [11]. Tou první je, že systém, který modelujeme je lineární (a samozřejmě, my musíme rovnice systému znát). To znamená, že nový stav systému získáme vynásobením starého stavu maticí přechodu. Dalšími dvěma vlastnostmi je to, že šum měření musí být bílý (tzn. nekoreluje v čase) a musí se dát namodelovat Gaussovou funkcí (tedy průměrem a kovariancí). Poslední, ne příliš často uváděnou podmínkou je to, že počáteční rozložení pravděpodobnosti systému musí být také Gaussovo. Tato podmínka se většinou neuvádí proto, že počáteční stav bývá většinou znám přesně. Pokud by ovšem mohl být systém na počátku ve více polohách, i když ty by byly přesně známy, nemohli bychom Kalmanův filtr použít. Jeho nevýhodou je totiž to, že může modelovat jen systémy s jedním pravděpodobnostním maximem a toto by byl případ, kdy bychom museli použít částicový filtr, který bude popsán dále.

Kalmanův filtr se používá ve dvou případech. Tím prvním je kombinování měření stejné veličiny, ale ze dvou senzorů. Druhým je upřesnění nepřesného odhadu stavu systému pomocí nepřesného měření tohoto stavu. Ve své podstatě je účelem Kalmanova filtru získat ze dvou zašuměných dat co nejpřesnější výslednou hodnotu.

Představme si, že máme systém, jehož jednorozměrný stav měříme jako posloupnost veličin $x_1, x_2, \ldots, x_n$. Naším úkolem je, zjistit co nejpřesnější hodnotu tohoto systému, spočítáme tedy průměr naměřených hodnot:

$$\mu_n = \frac{1}{n} \sum_{1}^{n} x_i \tag{2.15}$$

Pokud dostaneme další hodnotu x_{n+1} můžeme přepočítat x_n, ale mnohem výhodnější je použít původní hodnotu μ_n a opravit ji novou hodnotou x_{n+1}. Rovnice pro výpočet μ_{n+1} z původní hodnoty μ_n tedy bude následující

$$\mu_{n+1} = \frac{1}{n+1}\sum_{1}^{n+1} x_i = \frac{n}{n+1}\left(\frac{1}{n}\sum_{1}^{n} x_i + \frac{1}{n}x_{n+1}\right) \tag{2.16}$$

Abychom jako první člen v závorce dostali původní průměr, musíme celou sumu vydělit hodnotou n. Tím se nám zlomek změní na $n/(n+1)$ a druhý člen bude představovat aktualizaci původního průměru nově naměřenou hodnotou. Tento výraz pak můžeme přepsat jako

$$\mu_{n+1} = \frac{n}{n+1}\mu_n + \frac{1}{n+1}x_{n+1} = \mu_n + K(x_{n+1} - \mu_n), \tag{2.17}$$

kde $K = 1/(n+1)$. Takto dostaneme výraz, který nám v podstatě říká, že nová hodnota průměru μ_{n+1} je váhovaným součtem starého průměru μ_n a nové hodnoty x_{n+1} Protože původní průměr jsme spočítali z více hodnot, věříme mu samozřejmě mnohem více, než nové hodnotě x_{n+1}. Na rovnici můžeme také pohlížet tím způsobem, že starý průměr μ_n aktualizujeme rozdílem starého průměru μ_n a nové hodnoty x_{n+1}. Hodnota K nám pak říká, jak tato aktualizace bude velká. V literatuře se jí často říká *zesílení* (ang. gain).

Stejným způsobem můžeme vypočítat i rozptyl (ve vícerozměrném případě kovarianční matici). Jestliže máme n hodnot, rozptyl bude

$$\sigma_n^2 = \frac{1}{n}\sum_{1}^{n}(x_i - \mu_n)^2 \tag{2.18}$$

Pokud dostaneme další hodnotu x_{n+1}, nový rozptyl bude

$$\sigma_{n+1}^2 = \frac{1}{n+1}\sum_{1}^{n+1}(x_i - \mu_{n+1})^2 = \frac{1}{n+1}\sum_{1}^{n+1}(x_i - \mu_n - K(x_{n+1} - \mu_n))^2 \tag{2.19}$$

Tuto rovnici můžeme rozepsat, abychom ji následně mohli zjednodušit:

$$\sigma_{n+1}^2 = \frac{1}{n+1}\left[\sum_{1}^{n}(x_i - \mu_n)^2 + 2K\sum_{1}^{n}(x_i - \mu_n)(x_{n+1} - \mu_n) + nK^2(x_{n+1} - \mu_n)^2 + \right.$$
$$\left. + (1-K)^2(x_{n+1} - \mu_n)^2\right] \tag{2.20}$$

Protože druhý výraz v závorce je nulový, a po dalších úpravách, dostaneme

$$\sigma_{n+1}^2 = \frac{1}{n+1}\left[n\sigma_n^2 + (x_{n+1} - \mu_n)^2(nK^2 + (1-K)^2\right] \tag{2.21}$$

Vzhledem k tomu, že $nK^2 + (1-K)^2 = nK$, můžeme předchozí rovnici přepsat na

$$\sigma_{n+1}^2 = \frac{n}{n+1}(\sigma_n^2 + K(x_{n+1} - \mu_n)^2) = (1-K)(\sigma_n^2 + K(x_{n+1} - \mu_n)^2) \tag{2.22}$$

Podle [64] můžeme dále celý proces rozepsat do několika kroků. Nejprve předpokládejme, že máme n měření a máme spočítány hodnoty μ_n a σ_n.

1. Když přijde nová hodnota x_{n+1}, spočítáme zesílení $K = 1/(n+1)$.

2. Spočítáme nový průměr

$$\mu_{n+1} = \mu_n + K(x_{n+1} - \mu_n) \tag{2.23}$$

3. Spočítáme dočasnou hodnotu rozptylu

$$\sigma_n^{'2} = \sigma_n^2 + K(x_{n+1} - \mu_n)^2 \tag{2.24}$$

4. Konečný rozptyl spočítáme opravou dočasného rozptylu zesílením K

$$\sigma_{n+1}^2 = (1 - K)\sigma_n^{'2} \tag{2.25}$$

Tímto způsobem můžeme počítat průměr a rozptyl série hodnot, aniž bychom si museli pamatovat jednotlivé hodnoty.

Představme si nyní, že máme nějakou veličinu x, jejíž přesnou hodnotu neznáme, ale měříme ji pomocí dvou sad měření s průměry $\overline{x}_1$ a $\overline{x}_2$. Nyní z těchto dvou hodnot chceme vypočítat co nejpřesnější odhad $\hat{x}$ veličiny x. Pokud bychom věděli, že obě sady měření byly stejně přesné, spočítali bychom jednoduše jejich průměr $\overline{x}$. Pokud bychom ale na druhou stranu věděli, že jedna sada měření je o mnoho přesnější než druhá, vzali bychom do úvahy jen první z nich. V ostatních případech bychom museli jednotlivým měřením přiřadit nějaké váhy, které by nám určovaly, jak moc si myslíme, že jsou přesné. Pokud známe oba rozptyly, je přirozené, že jako váhy vezmeme tyto rozptyly. Otázkou je, jak je oba zkombinovat tak, aby nám dávaly co nejpřesnější výsledek. Protože obě měření mají Gaussovo rozložení pravděpodobnosti, můžeme pro ně psát

$$p_i(\hat{x}) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-(\hat{x}-\overline{x}_i)^2/2\sigma_i^2}, \tag{2.26}$$

kde $i = 1, 2$. Protože obě měření jsou nezávislá, výslednou pravděpodobnost dostaneme jejich vzájemným vynásobením

$$p(\hat{x}) = p_1(\hat{x})p_2(\hat{x}) = \frac{1}{2\sigma_1\sigma_2\pi} e^{-(\hat{x}-\overline{x}_1)^2/2\sigma_1^2 - (\hat{x}-\overline{x}_2)^2/2\sigma_2^2}, \tag{2.27}$$

Vzhledem k tomu, že chceme zjistit maximum této funkce, funkci zderivujeme podle $\hat{x}$ a položíme rovnu nule.

$$p'(\hat{x}) = p(\hat{x})(-(\hat{x} - \overline{x}_1)/\sigma_1^2 - (\hat{x} - \overline{x}_2)/\sigma_2^2) = 0, \tag{2.28}$$

Tato funkce se bude rovnat nule jen pokud se její druhá část bude rovnat nule

$$-(\hat{x} - \overline{x}_1)/\sigma_1^2 - (\hat{x} - \overline{x}_2)/\sigma_2^2 = 0. \tag{2.29}$$

Pokud teď vypočítáme $\hat{x}$, dostaneme

$$\hat{x} = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}\overline{x}_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\overline{x}_2 \tag{2.30}$$

Jestliže hodnotu zesílení K vypočítáme analogicky jako v předchozím případě, tzn. $K = \sigma_1^2/(\sigma_1^2 + \sigma_2^2)$ a $(1 - K) = \sigma_2^2/(\sigma_1^2 + \sigma_2^2)$, dostaneme

$$\hat{x} = \overline{x}_1 + K(\overline{x}_2 - \overline{x}_1) \tag{2.31}$$

a pro rozptyl

$$\sigma = (1 - K)\sigma_1^2 \tag{2.32}$$

Pro výpočet nemusí být obě hodnoty $\overline{x}_1$ a $\overline{x}_2$ průměry, mohou to být jednotlivá měření x_1 a x_2, pak ale musíme nějakým způsobem zjistit odpovídající rozptyly.

Předchozí rovnice představují nástin fungování Kalmanova filtru. V Kalmanově filtru ovšem hodnota $\overline{x}_1$ představuje předpověď stavu[1] systému v čase t a značí se $\hat{x}_t^-$ a hodnota $\overline{x}_2$ měření systému v čase t a značí se z_t. Hodnota $\hat{x}_t$ potom představuje stav systému v čase t aktualizovaný měřením z_t. Další změnou, která se objevuje v Kalmanově filtru je to, že hodnoty $\overline{x}_1$ a $\overline{x}_2$, resp. teď už $\hat{x}_t^-$ a z_t jsou vektory a tím pádem i z rozptylů σ_1 a σ_2 se stávají kovarianční matice. Kovarianční matice vzniklá z rozptylu σ_1 se značí P. Kovarianční matice vzniklá z rozptylu σ_2 se většinou značí R a odpovídá šumu měření. Poslední důležitou změnou v Kalmanově filtru je to, že výsledek měření může mít jiný rozměr, než je stav systému. Stav x_k pak tedy musíme vynásobit maticí H, která nám převede stav systému na rozměr měření. K tomuto je pak ještě třeba připočítat již zmíněnou kovarianční matici R, která odpovídá šumu měření.

Shrňme si tedy celou problematiku Kalmanova filtru:
Jak už bylo řečeno, pro použití Kalmanova filtru potřebujeme nejprve znát stavovou rovnici lineárního systému, jehož stav chceme vypočítat. Dále potřebujeme rovnici, která popisuje, jakým způobem budeme stav systému měřit.

$$x_k = Fx_{k-1} + Bu_k + w_k \tag{2.33}$$
$$z_k = Hx_k + v_k \tag{2.34}$$

- F je matice přechodu z minulého stavu do současného stavu
- x_{k-1} je minulý stav systému
- B převádí řídicí signály na změnu stavu systému
- u_k je řízení (neboli vstup) systému
- z_k je měření systému (někdy je také označováno jako výstup systému)
- H je matice, která převádí stav systému na měření
- k je časový index
- w a v jsou náhodné veličiny, které představují šum procesu resp. šum měření, jsou na sobě nezávislé a mají normální rozložení pravděpodobnosti, které má střed v nule a kovarianční matice Q a R:

$$p(w) \sim N(0, Q_k) \tag{2.35}$$
$$p(v) \sim N(0, R_k). \tag{2.36}$$

Kovarianční matice R už byla popsána výše, kovarianční matice Q pak analogicky odpovídá šumu procesu. Obě matice zde jsou doplněny o index k, neboť jejich hodnota se může po každém kroku systému změnit.

[1] Ve skutečnosti je to předpověď odhadu stavu systému, skutečný stav systému totiž nejsme schopni zjistit, a tak jej počítáme jako střední hodnotu, neboli vážený průměr. Skutečný stav systému tedy je x a jeho odhad je $\hat{x}$, kde $E[x] = \hat{x}$. Předpověď odhadu stavu systému je potom $\hat{x}^-$.

Stejně jako u Bayesova filtru se i rovnice Kalmanova filtru dělí na dvě části - *prediktor* a *korektor*. V části označované jako prediktor jsou rovnice předpovídající nový stav systému (respektive vypočítají ho ze stavové rovnice systému) a novou kovarianční matici systému. Ve druhé části, nazývané korektor, se provádí oprava odhadu nového stavu systému pomocí nového měření a výpočet nové kovarianční matice systému. V následujících rovnicích už je stav systému označen jako $\hat{x}_k$ a kovarianční matice jako P_k.

Prediktor

$$\hat{x}_k^- = F\hat{x}_{k-1} + Bu_k \tag{2.37}$$
$$P_k^- = FP_{k-1}F^T + Q_k \tag{2.38}$$
$$\tag{2.39}$$

Korektor

$$K_k = P_k^- H^T (HP_k^- H^T + R_k)^{-1} \tag{2.40}$$
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{2.41}$$
$$P_k = (I - K_k H)P_k^- \tag{2.42}$$

Kde I je jednotková matice.

Pro použití těchto vztahů je nutné, aby byl systém lineární. Tento stav však není ve skutečných systémech příliš častý, a proto se používá modifikace Kalmanova filtru nazývaná rozšířený Kalmanův filtr (Extended Kalman Filter, EKF).

Rozšířený Kalmanův filtr

V tomto případě je systém popsán nelineárními stochastickými diferenciálními rovnicemi

$$x_k = f(x_{k-1}, u_k, w_{k-1}) \tag{2.43}$$

a rovnice měření je potom

$$z_k = h(x_k, v_k) \tag{2.44}$$

V praxi samozřejmě neznáme hodnoty šumu w_k a v_k, ale i tak můžeme systém aproximovat

$$\tilde{x}_k = f(\hat{x}_{k-1}, u_k, 0) \tag{2.45}$$
$$\tilde{z}_k = h(\tilde{x}_k, 0) \tag{2.46}$$

Stejně jako obyčejný Kalmanův filtr má i EKF dvě fáze, predikci a korekci.

Prediktor

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k, 0) \tag{2.47}$$
$$P_k^- = F_k P_{k-1} F_k^T + W_k Q_{k-1} W_k^T \tag{2.48}$$

Korektor

$$K_k = P_k^- C_k^T (C_k P_k^- C_k^T + V_k R_k V_k^T)^{-1} \tag{2.49}$$
$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - h(\hat{x}_k^-, 0)) \tag{2.50}$$
$$P_k = (I - K_k C_k)P_k^- \tag{2.51}$$

Matice F a C jsou analogické jako v předchozím případě, matice W a V pak představují Jacobiho matice parciálních derivací.

2.4.3 Částicový filtr

Podstatou částicového filtru je reprezentovat hustotu pravděpodobnosti množinou vážených částic, tzn. čím vyšší je pravděpodobnost v určité oblasti, tím více částic a s větší vahou v tomto místě bude.

Jednotlivé částice se značí $x_t^{[n]}$ a označují stav v čase t. n odpovídá označení konkrétní částice a nabývá hodnot od 1 do N, kde N je počet částic v množině. Množina částic v čase t se označuje X_t. Váha částice je ω.

$$X_t = x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[N]} \tag{2.52}$$

Obecný algoritmus částicového filtru je ukázán na Algoritmu 2. Protože se v podstatě jedná o Bayesův filtr, skládá se ze dvou částí. První částí je první cyklus for, což odpovídá predikci, druhou částí je druhý cyklus for, což odpovídá korekci.

Algoritmus 2 ParticleFilter

Input: X_{t-1}, u_t, z_t
Output: X_t

1: **for** $n = 1$ to N **do**
2: sample $x_t^{[n]} \approx p(x_t \mid u_t, x_{t-1}^{[n]})$
3: $\omega_t^{[n]} = p(z_t \mid x_t^{[n]})$
4: $\overline{X_t} = \overline{X_t} + \langle x_t^{[n]}, \omega_t^{[n]} \rangle$
5: **end for**
6: **for** $n = 1$ to N **do**
7: draw i with probability $\approx \omega_t^{[i]}$
8: add $x_t^{[i]}$ to X_t
9: **end for**
10: **return** X_t

V části predikce vygenerujeme stav $x_t^{[n]}$ z předchozího stavu $x_{t-1}^{[n]}$ pomocí modelu pohybu a na základě řízení u_t. Na dalším řádku pomocí modelu měření spočítáme váhu částice $\omega_t^{[n]}$ a obojí přidáme do množiny částic na třetím řádku. Váha částice je v podstatě pravděpodobnost, podle které vybíráme částice do nové množiny částic X_t v druhé části algoritmu. Dá se říct, že algoritmus částicového filtru je podobný jako genetické algoritmy, jen s tím rozdílem, že zde se negenerují žádné nové vzorky, ale používají se pouze vzorky z minulé množiny, tzn. v množině může být po provedení korekce hodně vzorků naprosto stejných. Před převzorkováním rozložení částic odpovídá pravděpodobnosti $\overline{bel}(x_t)$ a po převzorkování $bel(x_t)$.

Převzorkování tvoří klíčovou část algoritmu. V případě, že by k němu nedocházelo, filtr by skončil v situaci, kdy by hodně vzorků mělo malé váhy a několik málo vzorků naopak váhy obrovské. V místech vzorků s malými vahami se robot pravděpodobně nevyskytuje, a tak je vcelku zbytečné se jimi zabývat, spíše by bylo vhodné, aby co nejvíce vzorků bylo v místech robotova nejpravděpodobnějšího výskytu. Na druhou stranu není ani příliš vhodné, aby převzorkovávání probíhalo příliš často, protože pak dochází k druhému extrému, tj. ke ztrátě různorodosti částic, což může mít za následek např. to, že v místě robota nebude žádná částice a algoritmus nebude pracovat správně. Proto se používají různé strategie pro řízení převzorkovávání částic, např. se doporučuje, aby se nepřevzorkovávalo, pokud se robot nepohybuje, nebo se počítají různé koeficienty z váhy částic a podle jejich hodnoty se

rozhodne, zda převzorkovat, či ne. Například v práci [42] jsou dva koeficienty, první je koeficient variace částic cv_t^2 a druhý efektivní velikost vzorku, ESS_t, M je počet částic a ω_i je váha i-té částice.

$$cv_t^2 = \frac{1}{M}\sum_{i=1}^{M}(M\omega_i - 1)^2 \tag{2.53}$$

$$ESS_t = \frac{M}{1 + cv_t^2} \tag{2.54}$$

Pokud ESS_t padne pod určitou hodnotu, většinou je touto hodnotou nějaké procento z počtu částic, je množina částic převzorkována.

K provedení samotného převzorkování existuje více algoritmů. Požadavek na převzorkování je takový, aby hustota pravděpodobnosti po převzorkování byla co nejvíce podobná hustotě pravděpodobnosti před převzorkováním.

Některé z používaných algoritmů jsou popsány v [63]. Nejjednodušší a asi také nejpoužívanější je metoda nazývaná Select with Replacement. Vstupem jsou normalizované váhy částic, které se postupně přičítají až do výsledku 1. Poté se náhodně vygeneruje N čísel v intervalu $\langle 0, 1\rangle$ a seřadí se podle velikosti. Pak se podle počtu vygenerovaných čísel, které padnou do intervalu postupně sečtených vah určí, které a kolik částic se objeví v nové množině. Používá se i varianta, kde se počítá s logaritmy a částice se vybírají v lineárním čase, nebo se z vah počítají odmocniny a na jejich základě se teprve vybírají do nové množiny.

Jiná metoda je popsána v [87]. Tam se náhodně vygeneruje číslo r v intervalu $\langle 0, 1/M\rangle$, kde M je počet požadovaných částic. Pak se v pravidelných intervalech $u = r + (m-1)/M$, kde $m = 1..M$ vybírají ty částice, do nichž tento interval padne.

Nevýhodou částicových filtrů je neefektivní reprezentace pravděpodobnosti v případě mnoharozměrných stavových prostorů. S tímto se částicové filtry vypořádávají redukcí stavového prostoru pomocí takzvané Rao-Blackwellizace. Tato metoda je popsaná v [20] a také v [54] a používá se i v případě VisualSLAMU, např. v sérii článků od R. Sima [72].

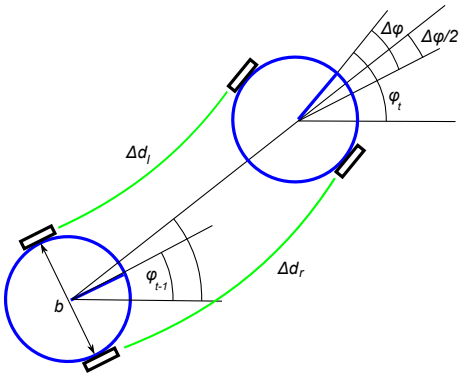
2.5 Model robota

Pro použití algoritmů, popsaných v předchozí kapitole, potřebujeme způsob, jak vypočítat pravděpodobnosti $p(x_t \mid x_{t-1}, u_t)$, která odpovídá pravděpodobnosti, že se robot po řízení u_t dostane z polohy x_{t-1} do polohy x_t a pravděpodobnost $p(z_t \mid x_t)$, tzn. pravděpodobnost měření z_t v místě x_t. První pravděpodobnost se vypočítá modelem pohybu robota, druhá pak modelem měření.

2.5.1 Model pohybu / Aktualizace stavu

Model pohybu robota nám říká, s jakou pravděpodobností se robot objeví v čase t na pozici x_t, jestliže víme, že v čase $t-1$ byl na pozici x_{t-1} při čemž jsme použili řízení u_t.

Protože nepřesnosti, které mají vliv na pohyb robota jsou ovlivněny Gaussovým šumem, můžeme i polohu robota reprezentovat pomocí Gaussova rozložení. Toto můžeme provést dvěma způsoby. Prvním způsobem je reprezentovat polohu robota pomocí pravděpodobnostní elipsy v případě 2D pohybu, nebo pomocí elipsoidu v případě 3D pohybu. V obou případech se bere v úvahu jen poloha robota bez jeho natočení. Tento způsob bychom



Obrázek 2.5: Ukázka robota určujícího svou polohu podle vzdáleností Δd_l a Δd_r ujetých levým a pravým kolečkem.

použili, pokud bychom chtěli pozici robota počítat pomocí Kalmanova filtru. Druhým způsobem by bylo z uvedeného Gaussova rozložení navzorkovat dostatečný počet hodnot a podle jejich rozmístění v prostoru pak určovat nejpravděpodobnější pozici robota. Tento druhý způsob se používá v případě, že chceme pozici robota počítat částicovým filtrem (viz kapitola 2.4.3).

Použijeme-li první nebo druhý způsob, vždy musíme spočítat pozici robota, do které by se dostal v případě provedení ideálního řízení u_t, tzn. bez vlivu šumu. Nyní máme několik možností, jak se bude robot pohybovat. Ty záleží na konstrukci robota.

V nejjednodušším případě bude robota, pohybující se v laboratorním prostředí, představovat vozítko, vybavené pásy nebo koly. V tomto případě nám plně postačí, pokud bude výpočet pozice robota pouze ve 2D (tzn. uvažujeme pouze souřadnice x, y a φ). Ve složitějším případě bude robotické vozítko vybaveno kamerou nebo kamerami umístěnými na pohyblivé hlavě (ang. pan-and-tilt unit). Požadovanou pozici robota v tomto případě budou tvořit kamery, takže musíme 2D pohyb rozšířit o další dvě souřadnice ψ a ω. Tímto způsobem budeme mít jednoznačně určenu pozici na vodorovné ploše[2].

V posledním, nejobecnějším, případě můžeme uvažovat, že se popsaný robot pohybuje nejenom po vodorovné rovině, ale i po šikmé, což nám zároveň dovolí tímto způsobem popsat jakýkoliv pohyb, tzn. můžeme spočítat pozici pro robota pohybujícího se nejenom na zemi, ale i ve vzduchu či ve vodě. V tomto případě musíme použít všech šest souřadnic nutných pro přesné určení pozice objektu ve 3D prostoru.

Pohyb ve 2D

Základní pohyb robota ve 2D případě popisují rovnice (2.55). Robot se v tomto případě otočí o úhel $\Delta\varphi$ a vykoná pohyb Δd.

$$\begin{aligned} x_t &= x_{t-1} + \Delta d \cos(\varphi_{t-1} + \Delta\varphi) \\ y_t &= y_{t-1} + \Delta d \sin(\varphi_{t-1} + \Delta\varphi) \\ \varphi_t &= \varphi_{t-1} + \Delta\varphi \end{aligned} \tag{2.55}$$

[2]Ve skutečnosti bychom ovšem potřebovali ještě dvě další souřadnice. Stav robota totiž není určen jen jeho pozicí v prostoru, ale v případě robota s pohyblivými částmi i vzájemnou polohou těchto částí. Stejně tak v případě 3D pohybu by nám nestačilo šest souřadnic, ale museli bychom jich mít osm. Pro zjednodušení však nebudeme uvažovat robota s pohyblivými částmi, proto si vystačíme jen se čtyřmi, resp. šesti souřadnicemi.

Pokud budeme uvažovat už poněkud reálnější situaci, např. podle [71], kdy je robot vybaven enkodéry na svých kolečkách a my tak víme, jakou vzdálenost urazila kolečka na levé a jakou kolečka na pravé straně, můžeme použít rovnice (2.56). Schéma takovéhoto pohybu je na obrázku 2.5.

$$\begin{aligned} x_t &= x_{t-1} + \Delta d \cos(\varphi_{t-1} + \Delta\varphi/2) \\ y_t &= y_{t-1} + \Delta d \sin(\varphi_{t-1} + \Delta\varphi/2) \\ \varphi_t &= \varphi_{t-1} + \Delta\varphi \\ \\ \Delta\varphi &= \frac{\Delta d_r - \Delta d_l}{b} \\ \Delta d &= \frac{\Delta d_r + \Delta d_l}{2} \end{aligned} \tag{2.56}$$

Rozdíl oproti předchozím rovnicím je ten, že zde máme pohyb robota rozdělen na pohyb pravého kolečka Δd_r a levého kolečka Δd_l, b je potom vzdálenost mezi oběma kolečky a Δd je průměr vzdáleností ujetých levým a pravým kolečkem. V podstatě Δd odpovídá vzdálenosti, ujeté středem robota.

Nejistota pozice ve 2D pohybu

Nejistota pozice má Gaussovo rozložení pravděpodobnosti se středem v bodě x_t a s kovarianční maticí Σ_t. Tu vypočítáme následujícím způsobem

$$\Sigma_t = F_{t-1}\Sigma_{t-1}F_{t-1}^T + F_{\Delta rl}\Sigma_\Delta F_{\Delta rl}^T \tag{2.57}$$

Matici Σ dostaneme jako součet předchozí hodnoty kovariance Σ_{t-1} a kovariance Σ_Δ, což je kovariance odpovídající pohybu z místa x_{t-1} do místa x_t. Tu vypočítáme například následujícím způsobem:

$$\Sigma_\Delta = \begin{bmatrix} k_r|\Delta d_r| & 0 \\ 0 & k_l|\Delta d_l| \end{bmatrix}, \tag{2.58}$$

kde Δd_l a Δd_r jsou vzdálenosti ujeté levým a pravým kolečkem a k_l a k_r jsou konstanty, které v sobě zahrnují chyby způsobené prokluzem koleček, chybami v měření enkodérů a dalšími. Matice F_{t-1} obsahuje parciální derivace rovnic (2.56) podle souřadnic x, y a φ.

$$F_{t-1} = \begin{bmatrix} \frac{\partial f}{\partial x} \frac{\partial f}{\partial y} \frac{\partial f}{\partial \varphi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta d \sin(\varphi + \Delta\varphi/2) \\ 0 & 1 & \Delta d \cos(\varphi + \Delta\varphi/2) \\ 0 & 0 & 1 \end{bmatrix} \tag{2.59}$$

Obdobně pak matice $F_{\Delta rl}$ obsahuje parciální derivace podle Δd_r a Δd_l.

$$F_{\Delta rl} = \begin{bmatrix} \frac{\partial f}{\partial \Delta d_r} \frac{\partial f}{\partial \Delta d_l} \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{1}{2}\cos\left(\varphi + \frac{\Delta\varphi}{2}\right) - \frac{\Delta d}{2b}\sin\left(\varphi + \frac{\Delta\varphi}{2}\right) & \frac{1}{2}\cos\left(\varphi + \frac{\Delta\varphi}{2}\right) + \frac{\Delta d}{2b}\sin\left(\varphi + \frac{\Delta\varphi}{2}\right) \\ \frac{1}{2}\sin\left(\varphi + \frac{\Delta\varphi}{2}\right) - \frac{\Delta d}{2b}\cos\left(\varphi + \frac{\Delta\varphi}{2}\right) & \frac{1}{2}\sin\left(\varphi + \frac{\Delta\varphi}{2}\right) - \frac{\Delta d}{2b}\cos\left(\varphi + \frac{\Delta\varphi}{2}\right) \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix} \tag{2.60}$$

Vizualizace chyb ve 2D

Počítáme-li i s nepřesností pohybu, je celkem zřejmé, že si pravděpodobné pozice robota budeme chtít nějak zobrazit [79], [80]. V jednorozměrném případě by se pravděpodobná poloha zobrazila jako úsečka se středem ve středu Gaussovy křivky a s délkou odpovídající požadované pravděpodobnosti. Ve dvourozměrném případě se z úsečky stane elipsa, jejíž velikost a natočení nám udávají oblast, ve které se s danou pravděpodobností robot nachází. Z kovarianční matice Σ vybereme tu její část, kterou budeme chtít zobrazit, tzn. tu se souřadnicemi x a y. To znamená, že z rovnice (2.61) (přepsaná rov. (2.57)) vypustíme třetí sloupec a třetí řádek.

$$\Sigma = \begin{bmatrix} \sigma_x^2 & \sigma_{yx} & \sigma_{\varphi x} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{\varphi y} \\ \sigma_{x\varphi} & \sigma_{y\varphi} & \sigma_\varphi^2 \end{bmatrix} \tag{2.61}$$

Úhel α mezi osou x a hlavní osou elipsy potom spočítáme jako

$$\alpha = 0.5 \arctan \frac{2\sigma_{xy}}{\sigma_x^2 - \sigma_y^2} \tag{2.62}$$

Délka hlavní a vedlejší osy elipsy se pak spočítá podle následujících rovnic,

$$\alpha_{maj} = \sqrt{\frac{k}{2}(\sigma_x^2 + \sigma_y^2 + \tau)} \tag{2.63}$$

$$\alpha_{min} = \sqrt{\frac{k}{2}(\sigma_x^2 + \sigma_y^2 - \tau)} \tag{2.64}$$

kde τ je

$$\tau = \sqrt{(\sigma_x^2)^2 + (\sigma_y^2)^2 - 2\sigma_x^2\sigma_y^2 + 4\sigma_{xy}^2} \tag{2.65}$$

a k je konstanta odpovídající požadované pravděpodobnosti Pr.

$$k = -2\ln(1 - Pr) \tag{2.66}$$

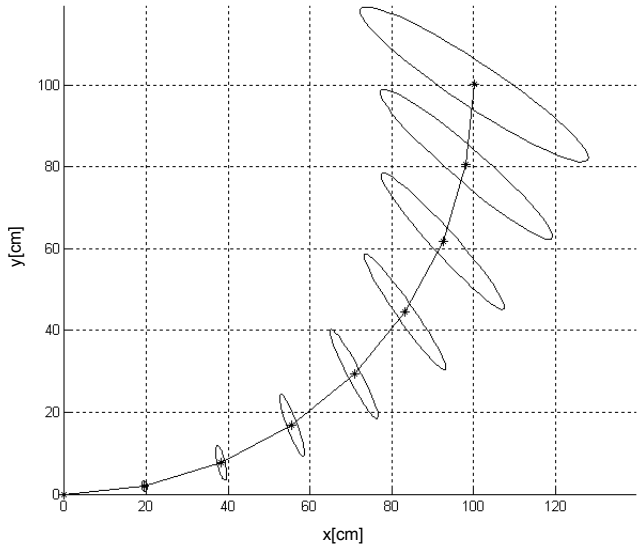
Výsledek pohybu robota a zobrazení nejistoty jeho pozice je vidět na obrázku 2.6, kde robot začal svůj pohyb na souřadnicích [0, 0], kde byla jeho pozice známá. V sérii osmi následných pohybů se ve tvaru čtvrtkruhu dostal až na pozici [100, 100]. Postupně se zvětšující elipsy zobrazují pravděpodobnou pozici robota po dokončení jednotlivých pohybů a symbolizují postupnou ztrátu informace o jeho přesné poloze.

Pohyb ve 3D

Při pohybu ve 3D je situace poněkud složitější, než v případě 2D pohybu. Tady totiž musíme použít už šest proměnných proto, abychom mohli popsat pohyb a pozici robota. Pro popis pohybu ve 3D prostoru se kvůli větší přehlednosti už většinou používá maticový zápis. Pozici robota nám bude udávat následující rovnice:

$$X = [x, y, z, \varphi, \theta, \psi]^T = \begin{bmatrix} T \\ A \end{bmatrix} \tag{2.67}$$

Pro větší přehlednost je zde oddělen translační a rotační pohyb robota.



Obrázek 2.6: Příklad robota pohybujícího se ve čtvrtkruhu a zobrazujícího postupně chybové elipsy. Na nich je vidět, jak se nejistota pozice robota zvyšuje s tím, jak se zvyšuje ujetá vzdálenost.

Pro to, abychom mohli počítat s maticovým zápisem, musíme rotaci vyjádřit jako matici a ne jako vektor. Tzn. úhly φ, θ, ψ musíme přepočítat do matice o velikosti 3×3. Pro tento přepočet se nejčastěji používají dva způsoby, popsané např. v [46]. Prvním je použití Eulerových úhlů a druhým jsou tzv. RPY úhly[3]. Rotace kolem jedné osy ve 3D prostoru jsou vyjádřeny maticemi, které se postupně přidávají tak, jak se těleso otáčí kolem jednotlivých os. Protože násobení matic není komutativní, záleží u otáčení na pořadí os, kolem kterých těleso rotuje. My budeme uvažovat pořadí, ve kterém se robot nejprve otočí kolem své svislé osy, protože tento pohyb bude vykonávat nejčastěji, pak kolem vodorovné osy nahoru nebo dolů a nakonec kolem vodorovné osy vpravo nebo vlevo, což bude pohyb nejméně častý. Posloupnost pohybů tedy bude následující:

$$F_0 \overset{R_z(\varphi)}{\longrightarrow} F_1 \overset{R_{y'}(\theta)}{\longrightarrow} F_2 \overset{R_{x''}(\psi)}{\longrightarrow} F_3 \tag{2.68}$$

Říká nám, že se z počáteční soustavy souřadnic F_0 dostaneme do F_1 tak, že se otočíme o úhel φ okolo osy z, poté se otočením o úhel θ okolo nové osy y' dostaneme do souřadného systému F_2 a nakonec otočením o úhel ψ okolo nové osy x'' do cílového souřadného systému F_3.

$$R_{RPY} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{13} & r_{23} & r_{33} \end{bmatrix} = R_{x''}(\psi) R_{y'}(\theta) R_z(\varphi) = \tag{2.69}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \tag{2.70}$$

[3]RPY je zkratka pro roll, pitch a yaw, což jsou rotace okolo vodorovné osy vlevo a vpravo, pak kolem vodorovné osy nahoru a dolů a nakonec kolem svislé osy.

$$= \begin{bmatrix} \cos\theta\cos\varphi & -\cos\theta\sin\varphi & \sin\theta \\ \cos\psi\sin\varphi + \sin\psi\sin\theta\cos\varphi & \cos\psi\cos\varphi - \sin\psi\sin\theta\sin\varphi & -\sin\psi\cos\theta \\ \sin\psi\sin\varphi - \cos\psi\sin\theta\cos\varphi & \sin\psi\cos\varphi + \cos\psi\sin\theta\sin\varphi & \cos\psi\cos\theta \end{bmatrix} \quad (2.71)$$

Na rovnicích (2.69) až (2.71) vidíme výpočet matice R_{RPY} - vynásobením jednotlivých rotačních matic. Když si pro lepší přehlednost přeznačíme R_{RPY} na ΔR, dostaneme výpočet nové rotační matice R_t jako součin změny rotace ΔR a původní rotace R_{t-1}, viz rovnice (2.72).

$$R_t = \Delta R R_{t-1} \quad (2.72)$$

Novou polohu objektu získáme vynásobením nové rotace a nového pohybu a přičtením k minulé poloze.

$$T_t = R_t[\Delta x, \Delta y, \Delta z]^T + [x_{t-1}, y_{t-1}, z_{t-1}]^T \quad (2.73)$$

Vyjádření rotace objektu maticově je výhodné pro výpočet, těžko ale z něj zjistíme, o jaké úhly máme objekt orotovat okolo jednotlivých os. Pro přepočet do rotačních úhlů můžeme použít následující vztah.

$$A = \begin{bmatrix} \arctan 2(-r_{12}, r_{11}) \\ \arctan 2(r_{13}, r_{11}\cos\varphi - r_{12}\sin\varphi) \\ \arctan 2(r_{31}\sin\varphi - r_{32}\cos\varphi, -r_{21}\sin\varphi + r_{22}\cos\varphi) \end{bmatrix} \quad (2.74)$$

Příklad algoritmu pro 2D pohyb

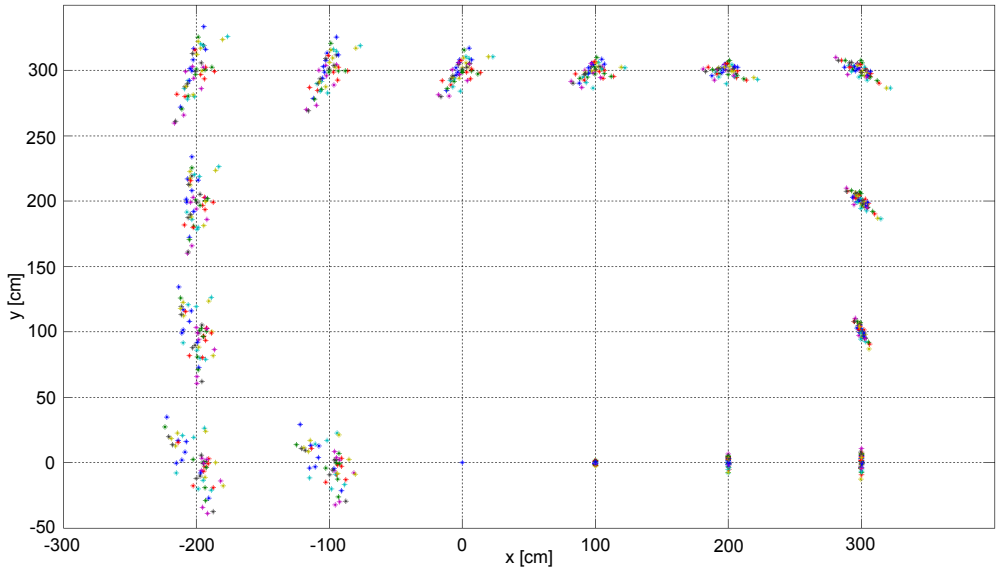
Při bližším pohledu na obecný algoritmus částicového filtru (Algoritmus 2) zjistíme, že jeho vstupem je sada částic a hodnoty řízení a měření. Nás budou v této chvíli zajímat právě částice a hodnoty řízení. Částici v tomto případě tvoří pozice robota v prostoru a její pravděpodobnost, tzn. množinu částic X_t budou tvořit dvojice hodnot $\langle x_t^{[n]}, \omega_t^{[n]} \rangle$. V předpisu $p(x_t \mid x_{t-1}, u_t)$ bude částicí $x_t^{[n]}$ hodnota x_t a hodnota této pravděpodobnosti bude tvořit pravděpodobnost (váhu) částice $\omega_t^{[n]}$, kde n je index částice, který nabývá hodnot od 1 do N. My do částicového filtru potřebujeme generovat částice s požadovaným, nejčastěji Gaussovým, rozložením, v závislosti na pohybu robota, který vypočítáme některými z předešlých rovnic. Celý proces je vidět na Algoritmu 3 .

Algoritmus 3 SampleMotionModelOdometry

Input: $x_{t-1}, u_t = (\delta_{rot1}, \delta_{trans}, \delta_{rot2})$
Output: $x_t = (x', y', \theta')^T$

1: $\tilde{\delta}_{rot1} = \delta_{rot1} - sample(\alpha_1\delta_{rot1}^2 + \alpha_2\delta_{trans}^2)$
2: $\tilde{\delta}_{trans} = \delta_{trans} - sample(\alpha_3\delta_{trans}^2 + \alpha_4\delta_{rot1}^2 + \alpha_4\delta_{rot2}^2)$
3: $\tilde{\delta}_{rot2} = \delta_{rot2} - sample(\alpha_1\delta_{rot2}^2 + \alpha_2\delta_{trans}^2)$
4:
5: $x' = x + \tilde{\delta}_{trans}\cos(\theta + \tilde{\delta}_{rot1})$
6: $y' = y + \tilde{\delta}_{trans}\sin(\theta + \tilde{\delta}_{rot1})$
7: $\theta' = \theta + \tilde{\delta}_{rot1} + \tilde{\delta}_{rot2}$
8: **return** $x_t = (x', y', \theta')^T$

Vstupem je pozice robota v čase $t-1$ a bezprostřední řízení u_t. Výstupem je nová pozice robota x_t. Algoritmus vychází z rozdělení pohybu robota do tří částí. První je natočení robota do požadovaného směru jízdy δ_{rot1}, druhá je překonání požadované vzdálenosti δ_{trans}

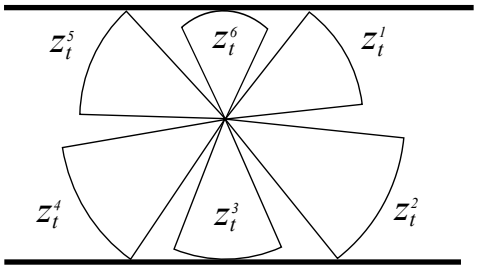


Obrázek 2.7: Ukázka pohybu skupiny částic podle algoritmu 3. Částice, představující robota, začaly svůj pohyb v bodě [0, 0]. Vykonaly 15 návazných pohybů po obdélníkové trajektorii proti směru hodinových ručiček. Na obrázku je dobře vidět, jak se zvětšuje plocha, kterou částice zabírají. To představuje zvětšující se nejistotu polohy robota.

a třetí je konečné natočení robota δ_{rot2}. Toto konečné natočení může být i nulové. V další části jsou vypočtené hodnoty "zašuměny"Gaussovým šumem (provádí funkce *sample*), v závislosti na tom, zda se jedná o pohyb translační nebo rotační. V poslední části jsou vypočtené hodnoty přičteny k původní pozici robota a my tak dostaneme jeho novou pozici. Velikost šumu se nastavuje pomocí konstant α_1 až α_4.

2.5.2 Model měření

Další pravděpodobností, kterou budeme potřebovat vypočítat, je pravděpodobnost měření $p(z_t \mid x_t)$. Ta v částicovém filtru bude odpovídat váze částice $\omega_t^{[m]}$. Robot zjišťuje informace o svém okolí různými senzory, v tomto případě to budou především sonary, laserový scanner, event. kamera. Robot v jednom okamžiku (na jednom místě) neprovádí pouze jedno měření,



Obrázek 2.8: Měření $z_t = \{z_t^1, z_t^2, \ldots, z_t^6\}$, robota detekujího zdi. Robot má šest sonarů, tzn. $K = 6$.

ale provádí jich celou řadu. Ve většině případů není vybaven pouze jedním sonarem, ale celým prstencem sonarů. Má sice většinou jen jeden laserový scanner, ale ten je schopen provést až několik set měření v úhlu většinou 180°. Jedna taková hodnota (měření jednoho sonaru, jedno změření paprskem laseru) se bude označovat z_t^k. Značení

$$z_t = \{z_t^1, z_t^2, \ldots, z_t^K\} \tag{2.75}$$

bude odpovídat sadě měření v čase t pro K sonarů, event. pro (jedno) měření, ale K paprsky u laseru. Pravděpodobnost $p(z_t \mid x_t, m)$ určuje pravděpodobnost měření z v čase t za předpokladu, že robot je v pozici x v mapě m. Tato pravděpodobnost se spočítá jako součin pravděpodobností jednotlivých měření

$$p(z_t \mid x_t, m) = \prod_{k=1}^{K} p(z_t^k \mid x_t, m) \tag{2.76}$$

Teď už zbývá jen spočítat pravděpodobnost jednoho měření $p(z_t^k \mid x_t, m)$. Tato pravděpodobnost se spočítá jako součet čtyř různých pravděpodobností [87]. Pravděpodobnosti, že senzor správně naměří hodnotu, pravděpodobnosti, že senzor naměří menší hodnotu (např. změří vzdálenost k právě procházejícímu člověku), pravděpodobnosti, že dojde k naměření náhodné vzdálenosti z neznámých důvodů a pravděpodobnosti, že senzor vlivem selhání naměří maximální hodnotu. Pravděpodobnost naměření skutečné vzdálenosti $p_{hit}(z_t^k \mid x_t, m)$ se modeluje Gaussovou křivkou s maximem v bodě skutečné vzdálenosti překážky, označované z_t^{k*}. Pravděpodobnost naměření menší hodnoty $p_{short}(z_t^k \mid x_t, m)$ je dána klesající exponenciálou začínající v bodě 0 a končící v z_t^{k*}. Pravděpodobnost naměření zcela náhodného výsledku $p_{rand}(z_t^k \mid x_t, m)$ je konstanta s počátkem v 0 a koncem v hodnotě Z_{max}, což je maximální vzdálenost naměřitelná sonarem. Pravděpodobnost $p_{max}(z_t^k \mid x_t, m)$ vrácení maximální hodnoty Z_{max} je určena funkcí podobnou Diracově impulsu v bodě Z_{max}. Hodnoty z_{hit}, z_{short}, z_{max} a z_{rand} se stanoví buď empiricky nebo výpočtem se série naměřených hodnot.

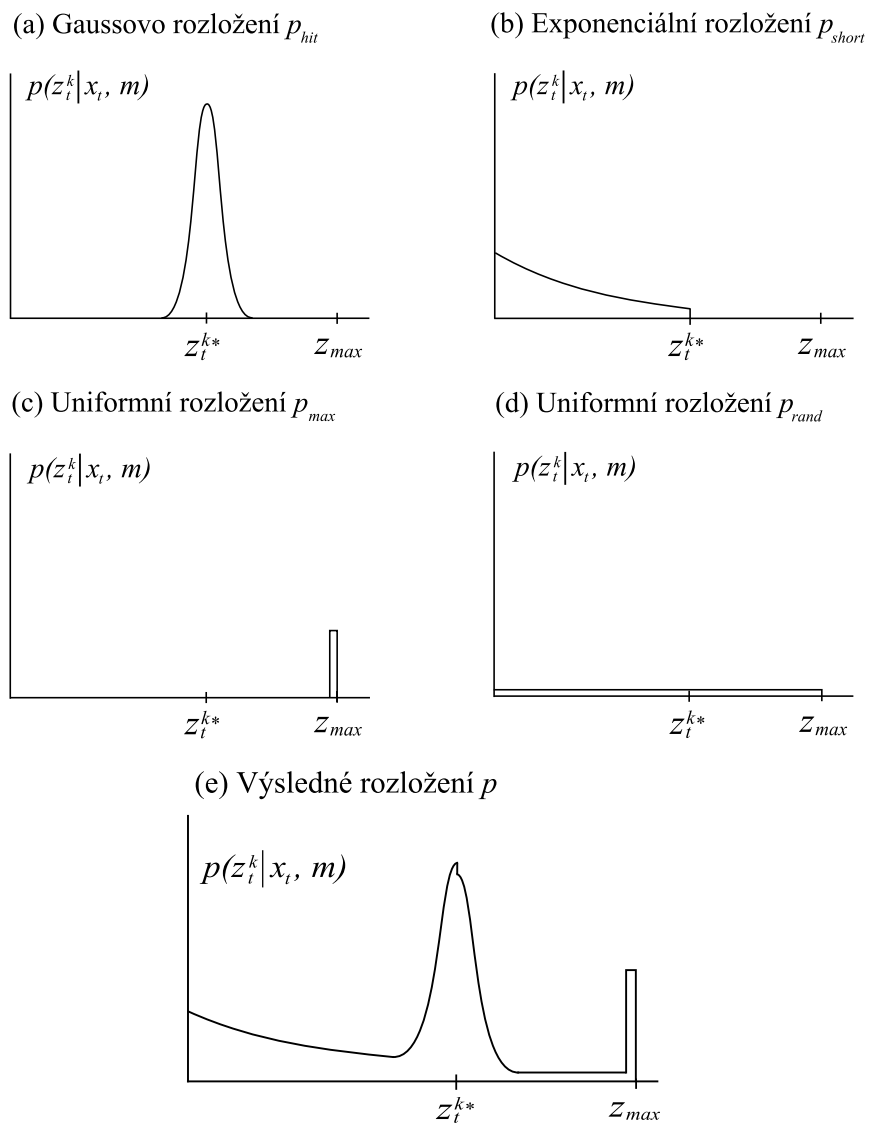
Algoritmus 4 BeamRangeFinderModel

Input: z_t, x_t, m
Output: q

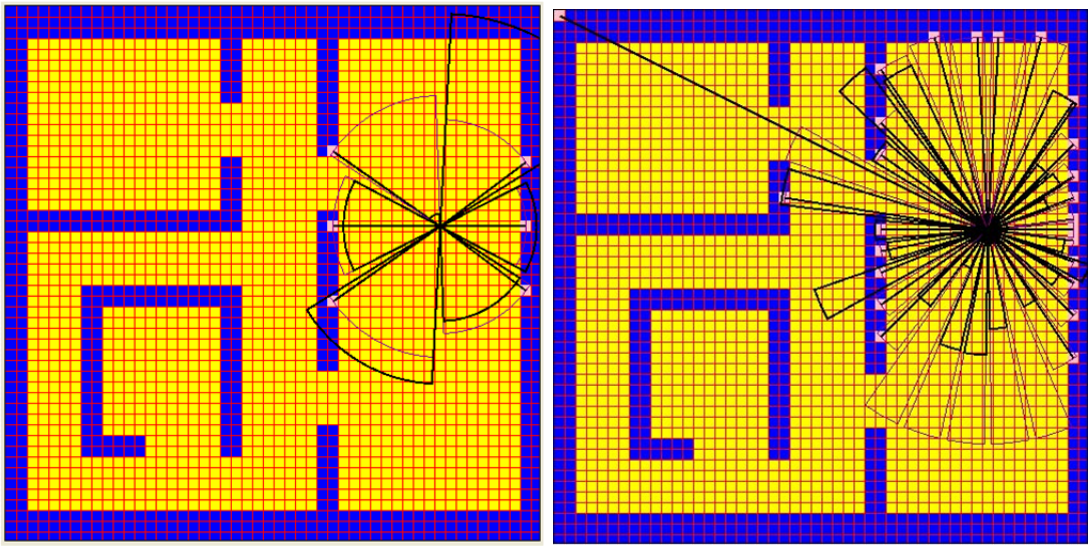
1: $q = 1$
2: **for** $k = 1$ to K **do**
3: compute z_t^{k*} for the measurement z_t^k using ray casting
4: $p = z_{hit}.p_{hit}(z_t^k \mid x_t, m) + z_{short}.p_{short}(z_t^k \mid x_t, m) + z_{max}.p_{max}(z_t^k \mid x_t, m) + z_{rand}.p_{rand}(z_t^k \mid x_t, m)$
5: $q = q.p$
6: **end for**
7: **return** q

2.5.3 Model měření pro kamery

Princip modelu měření pro kamery je ve své podstatě stejný jako pro sonary [51]. Jedna sada měření (nalezení a spárovnání význačných bodů v páru obrázků a výpočet jejich polohy) je tvořena měřeními (detekcí) z_t^i jednotlivých význačných bodů. Pravděpodobnost jedné



Obrázek 2.9: Na obrázcích (a)-(d) jsou čtyři rozložení pravděpodobností, která se uplatňují při měření sonarem nebo laserem. Na obrázku (e) je pak výsledné rozložení, které vznikne sečtením předchozích rozložení pravděpodobnosti. Obrázky převzaty z [87].



Obrázek 2.10: Simulace měření z_t robotem se šesti sonary. Silnou černou čarou jsou vyznačené výseče odpovídající jednotlivým měřením, růžově jsou detekované nejbližší buňky. Tenké výseče pak odpovídají skutečné vzdálenosti těchto buněk od robota. Na obrázku vpravo je příklad robota s více sonary (v tomto případě by to mohl být např. laserový scanner). Tenké výseče na spodní straně pravého obrázku představují maximální dosah měření.

sady měření $p(z_t \mid x_t)$ opět určíme vynásobením pravděpodobností pro jednotlivá měření $p(z_t^i \mid x_t)$.

$$p(z_t \mid x_t) = \prod_{i=1}^{K} p(z_t^i \mid x_t) \tag{2.77}$$

Hodnota K je počet nalezených význačných bodů v páru obrázků. Na rozdíl od analogické hodnoty u sonarů nebo laserů, se tato hodnota u kamer mění, protože počet nalezených význačných bodů je pokaždé jiný.

Pravděpodobnost jednoho měření $p(z_t^i \mid x_t)$ spočítáme jako součet pravděpodobností toho, že daný význačný bod z_t^i odpovídá některému z bodů v mapě $c_i = \psi$. $\psi = \{1, \ldots, M, \phi\}$ (nebo neodpovídá žádnému, a to značí proměnná ϕ), M je počet bodů v mapě. c_i je náhodná proměnná, která udává korespondenci detekovaného bodu a bodu v mapě a nabývá hodnot z ψ.

$$p(z_t^i \mid x_t) = \sum_{\psi=\{1,\ldots,M,\phi\}} p(z_t^i \mid x_t, c_i = \psi) P(c_i = \psi \mid x_t) \tag{2.78}$$

$P(c_i = \psi \mid x_t)$ je *apriori* pravděpodobnost toho, že bod c_i odpovídá některému z bodů v ψ. Protože je to konstanta, můžeme ji označit jako η a vytknout ji před sumu.

$$p(z_t^i \mid x_t) = \eta \sum_{\psi=\{1,\ldots,M,\phi\}} p(z_t^i \mid x_t, c_i = \psi) \tag{2.79}$$

Pravděpodobnost $p(z_t^i \mid x_t, c_i = \psi)$ je v podstatě pravděpodobnost toho, že detekovaný bod z_t^i koresponduje s nějakým bodem v mapě. Tato pravděpodobnost má Gaussovo rozložení, kde μ se spočítá jako rozdíl střední hodnoty detekovaného bodu $\bar{z_t^i}$ a bodu $\bar{m}_\psi$ v mapě a

kovariance Σ jako součet jejich kovariančních matic. Čím si budou oba body v prostoru souřadnic a deskriptoru blíže, tím bude tato pravděpodobnost větší.

$$p(z_t^i \mid x_t, c_i = \psi) = N\left(0; \underbrace{\bar{z}_t^i - \bar{m}_\psi}_{\mu}, \underbrace{\Sigma_{z_t^i} + \Sigma_{m_\psi}}_{\Sigma}\right) =$$

$$\eta' \exp\left\{-\frac{1}{2}(\bar{z}_t^i - \bar{m}_\psi)^T (\Sigma_{z_t^i} + \Sigma_{m_\psi})^{-1} (\bar{z}_t^i - \bar{m}_\psi)\right\} \tag{2.80}$$

kde
$\eta' = \left(2\pi \left|\Sigma_{z_t^i} + \Sigma_{m_\psi}\right|\right)^{-\frac{1}{2}}$.
Hodnoty μ a Σ můžeme rozdělit na části odpovídající souřadnicím bodu a souřadnicím deskriptoru a počítat je zvlášť.

$$N(0; \mu, \Sigma) = \eta' \exp\left\{-\frac{1}{2}\left(\begin{array}{c|c} \mu_{\mathbf{X}}^T & \mu_{\mathbf{F}}^T \end{array}\right) \left(\begin{array}{c|c} \Sigma_{\mathbf{X}} & \mathbf{0} \\ \hline \mathbf{0}^T & \Sigma_{\mathbf{F}} \end{array}\right)^{-1} \left(\begin{array}{c} \mu_{\mathbf{X}} \\ \hline \mu_{\mathbf{F}} \end{array}\right)\right\} =$$

$$= \eta' \exp\left\{-\frac{1}{2}\left(\mu_{\mathbf{X}}^T \Sigma_{\mathbf{X}}^{-1} \mu_{\mathbf{X}} + \mu_{\mathbf{F}}^T \Sigma_{\mathbf{F}}^{-1} \mu_{\mathbf{F}}\right)\right\} =$$

$$= \eta' \underbrace{\exp\left\{-\frac{1}{2}\mu_{\mathbf{X}}^T \Sigma_{\mathbf{X}}^{-1} \mu_{\mathbf{X}}\right\}}_{pozice} \underbrace{\exp\left\{-\frac{1}{2}\mu_{\mathbf{F}}^T \Sigma_{\mathbf{F}}^{-1} \mu_{\mathbf{F}}\right\}}_{deskriptor} \tag{2.81}$$

Vzhledem k tomu, že kovarianční matice deskriptoru je diagonální, vypočítáme její inverzi jako inverzi jednotlivých prvků. Exponent deskriptoru pak spočítáme jako součet součinů odpovídajících si prvků. I když tímto si velice usnadníme výpočet, je i přesto vyčíslení pravděpodobnosti jedné sady měření velice náročné na čas, zvláště uvážíme-li, že je potřeba provést pro každou částici.

2.6 Lokalizace

Lokalizace robota spočívá v určení jeho polohy pomocí mapy a měření ze senzorů. Dá se říct, že existují tři různé druhy lokalizace. V prvním případě se jedná pouze o upřesnění polohy robota poté, co robot ujel nějakou vzdálenost. Tento problém se nazývá *sledování pozice* (ang. position tracking). Druhým případem je *globální lokalizace*, kdy je robot nucen najít svou polohu v rámci celé mapy. V tomto případě je také důležité, aby byl schopen pracovat s více možnými pozicemi (ang. multiple-hypotheses). Poslední je tzv. *problém uneseného robota* (ang. kidnapped robot problem), kdy se změní pozice už lokalizovaného robota, aniž by o tom byl informován. Tato technika se často používá pro testování lokalizačních algoritmů - pokud dojde k fatálnímu selhání algoritmu a my chceme vědět, jak se s tím algoritmus vypořádá. Všechny tyto problémy se stávají obzvláště obtížné, pokud se provádí lokalizace v dynamicky se měnícím prostředí, např. v místech, kde se pohybuje velké množství lidí.

Počátky lokalizačních algoritmů jsou na konci 80. let minulého století. Dobrý přehled těchto algoritmů je v knize [10] J. Borensteina z roku 1996. Částečně z ní vychází i text následujících dvou podkapitol. Jako první se objevily lokalizační algoritmy založené na rozšířeném Kalmanově filtru, např. [39]. Následně se objevil "map matching", lokalizační technika,

kdy se lokální mapa porovnává s globální mapou [76]. Použití mřížky obsazenosti pro lokalizaci robota je pak srovnáno v [69]. Pojem Markovova lokalizace (neboli použití Bayesova filtru na lokalizaci robota) pochází z práce Simmonse a Koeniga [77] a např. z [29]. A na závěr, za state-of-art v lokalizaci je považována Monte Carlo lokalizace, poprvé popsaná v článcích D. Foxe a S. Thruna [28], [89]. S postupným rozšiřováním používání kamer v robotice se objevily i metody Monte Carlo lokalizace pomocí kamery, jedna z těchto metod je popsána například v [23].

2.6.1 Relativní měření pozice

Relativní měření pozice neboli Dead Reckoning [92] je založeno na sledování ujeté vzdálenosti, směru nebo doby pohybu od poslední známé pozice. V robotice se používají hlavně odometrie a inerciální navigace.

Odometrie

Odometrie je nejpoužívanější způsob ke zjištění pozice robota. Používá enkodéry, pomocí kterých počítá otáčky nebo natočení koleček robota. Díky enkodérům a znalosti obvodu koleček můžeme určit ujetou vzdálenost nebo natočení robota. Tato metoda má dobrou krátkodobou přesnost a je velmi jednoduchá a tudíž i levná, ale má jednu velkou nevýhodu a tou je narůstající chyba při měření a nemožnost se této chyby zbavit. Je způsobena prokluzem kol na různých površích.

Enkodéry se obecně dělí na absolutní a relativní. Relativní enkodéry měří pouze rotační rychlost a mohou tak poskytnout pouze relativní informaci o pozici. Naproti tomu výstupem absolutního enkodéru je přímo absolutní natočení měřené osy.

Novým druhem odometrie je vizuální odometrie. Ta je možná za použití kamery. Její princip v jednoduchosti spočívá v tom, že kamera nalezne určité význačné body v obraze a spočítá se jejich vzdálenost od robota. Pokud se pak robot posune a naleznou se ty samé význačné body a opět se spočítá jejich vzdálenost od robota, jsme schopni určit posunutí a změnu natočení robota. Použitím vizuální odometrie také odstraníme největší nevýhodu enkodérů a tou je chybné určení otočení koleček při pohybu na kluzkém povrchu.

Inerciální navigace

Zde se používá tzv. inerciálních senzorů, jako jsou gyroskopy a akcelerometry pro měření rychlosti pohybu vozidla a následné pozice, přičemž primární měřenou veličinou je zpravidla zrychlení. Celý princip spočívá ve faktu, že známe-li startovní pozici objektu a zaznamenáme-li změny zrychlení ve všech osách, které jsou pro měření podstatné, jsme schopni vypočítat současnou rychlost a pozici. Ačkoli princip metody je velmi jednoduchý (ze zrychlení se první integrací získá rychlost a další integrací pozice), praktická realizace je velmi obtížná a zejména nákladná, protože s integrací užitečného signálu je pochopitelně integrována i chyba. S vývojem optických gyroskopů se ovšem dramaticky snížila jejich cena a umožnila tak použití gyroskopů i v mobilních robotech.

2.6.2 Absolutní měření pozice

Je nezávislé na předchozích měřeních, a tudíž není, na rozdíl od relativního měření, zatíženo stále narůstající chybou. Používá se orientačních bodů (přirozených nebo umělých) nebo map.

Orientační body

Tzv. landmarky jsou takové body, které může robot detekovat pomocí svých senzorů. Dělí se na aktivní a pasivní.

Aktivní orientační body
jsou takové body, které aktivně vysílají nějaký signál umožňující zjistit polohu. Mohou to být např. satelity nebo majáky. Používají se většinou k navigaci lodí, letadel nebo ponorek. Pro navigaci pomocí majáků je nutný systém majáků, přičemž aby byla navigace možná, je třeba, aby z každého místa byly viditelné alespoň tři majáky. Navigaci pomocí satelitů používá dobře známý systém GPS, zde je ovšem nutná přímá viditelnost alespoň čtyř satelitů. Spolu s nedostatečnou přesností je toto hlavní nevýhodou navigace pomocí GPS, protože u mnoha robotů je požadavek na pohyb ve vnitřním prostředí (budovy atd.), kde se signál GPS nešíří. Špatný signál je také např. pod stromy nebo ve městech, kde jsou vysoké budovy. GPS se tak dá použít spíše jen jako doplňkový senzor pro určení polohy a jako primární je potřeba použít jiný. V současné době jsou to především enkodéry a inerciální navigace. Další nevýhodou všech aktivních orientačních bodů je nutnost je instalovat a udržovat, což může být finančně nákladné a robot se pak může pohybovat pouze v oblastech pokrytých sítí těchto bodů. Používání GPS je vhodné spíše pro venkovní roboty pohybující se v otevřené krajině, a tak tato metoda najde uplatnění spíše u armádních projektů než u robotů v laboratořích.

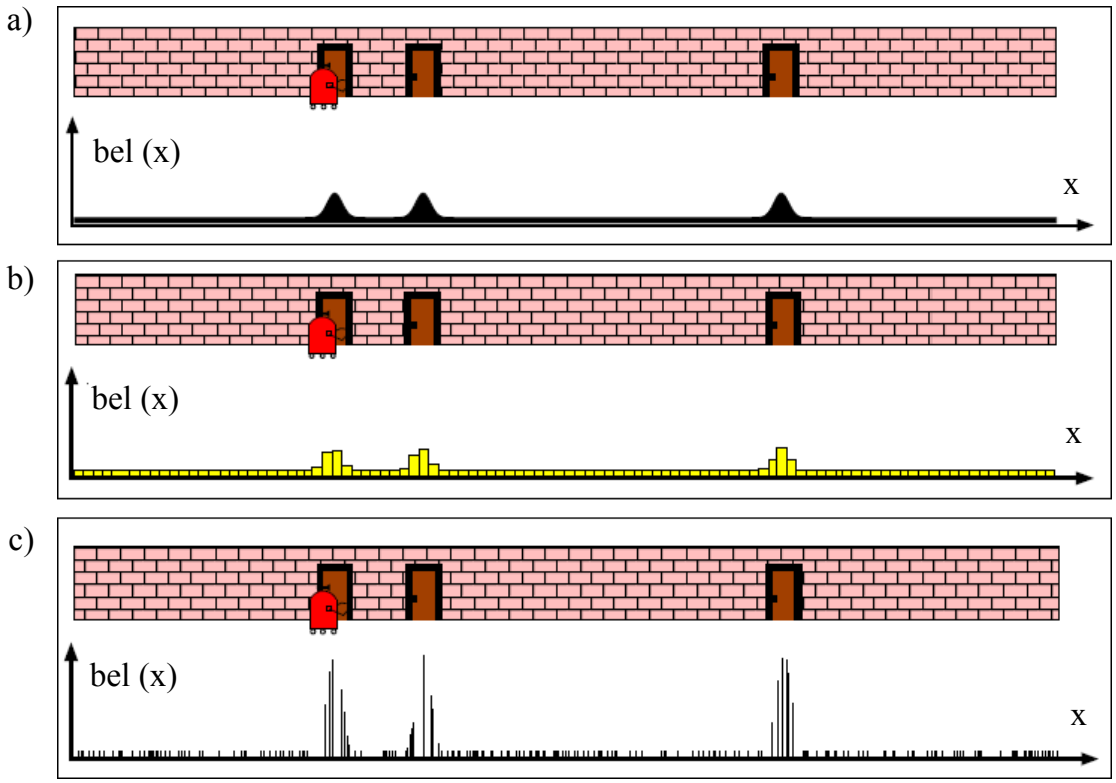
Pasivní orientační body
nevysílají žádný signál a robot je pro určení své pozice musí aktivně vyhledávat. K jejich vyhledávání se nejčastěji používá kamerový systém a stejně jako u předešlé metody i tady je potřeba, aby byly viditelné alespoň tři body, ke správnému určení pozice robota. Pasivní orientační body se dále dělí na umělé a přirozené.

Umělé orientační body
jsou navrhnuty a vhodně rozmístěny v prostředí tak, aby je mohl robot snadno používat pro svou navigaci. Používají se např. různé geometrické tvary jako krychle, koule a další. Jejich použití je vhodné zejména tam, kde nedochází k velkým změnám prostředí. Stejně jako u aktivních orientačních bodů je ale i tady nevýhoda v tom, že robot se může pohybovat pouze v oblastech pokrytých sítí těchto bodů.

Přirozené orientační body
nejsou navrženy přímo pro navigaci robota, ale jsou už součástí jeho prostředí. Mohou to být okna, dveře, světla, stromy, sloupy, semafory a další. Jejich výhodou je to, že nemusí být schválně navrhovány a rozmisťovány v prostředí, kde se bude robot pohybovat. Na druhé straně jejich velkou nevýhodou je velká výpočetní náročnost při jejich rozpoznávání a také to, že ne vždy jsou správně rozpoznány. Mapovací programy, které používají vyhledávání význačných bodů pracují právě s přirozenými orientačními body. Ty nejsou tvořeny celými objekty, ale spíše výraznými body, např. rohy. Každý takový bod je popsán soustavou deskriptorů (viz kapitola 3.4), umožňující jeho identifikaci v databázi nalezených bodů.

Porovnání s modelem prostředí

Je založeno na tom, že robot pomocí svých senzorů vytváří lokální mapu prostředí a tu potom porovnává s globální mapou, uloženou ve své paměti. Pokud je nalezena shoda,



Obrázek 2.11: Tři druhy lokalizací. a) Markovova lokalizace. b) Tzv. grid-based lokalizace. Pravděpodobnost polohy robota je reprezentována pomocí histogramu. c) Monte Carlo lokalizace. Pravděpodobnost je reprezentována hustotou a velikostí jednotlivých částic. Obrázky jsou převzaty z [87].

robot může určit svou pozici v globální mapě. Nevýhodou této metody je, že v závislosti na použitých senzorech robot potřebuje udělat mnoho měření, aby si vytvořil co nejpřesnější lokální mapu prostředí a tu pak mohl co nejlépe srovnat s globální mapou. Velmi často se také systémy s modelem prostředí kombinují s některým typem vyhledávání orientačních bodů.

2.6.3 Markovova lokalizace

Následující druhy lokalizace jsou dobře popsány v [87]. Všechny pravděpodobnostní lokalizační algoritmy vycházejí z Bayesova filtru. Jeho přímá aplikace se nazývá Markovova lokalizace a je schopna vyřešit všechny tři druhy lokalizace popsané výše. Poloha robota je v tomto případě určena pravděpodobnostní funkcí, která má maxima v bodech, kde se robot pravděpodobně vyskytuje. Pokud na počátku lokalizace víme, kde se robot nachází, umístíme toto maximum do tohoto bodu. Pokud počáteční pozici neznáme, rozprostřeme pravděpodobnost rovnoměrně po celém prostoru.

2.6.4 EKF lokalizace

Jiným druhem lokalizace je EKF (Extended Kalman Filter) lokalizace, což je speciální případ Markovovy lokalizace. Použití Kalmanova filtru v robotice je široce používaná a oblíbená metoda. Poloha robota a překážek (často reprezentovaných tzv. landmarky) je určena Gaussovou funkcí pomocí jejich středních hodnot a kovariancí. Nevýhodou tohoto přístupu je možnost pracovat pouze s jednou možnou polohou robota, protože Gaussova křivka má jen jedno maximum. Tato nevýhoda byla překonána použitím více Gaussových křivek (MHT, Multi-hypothesis tracking).

2.6.5 Grid-based lokalizace

Lokalizace na mřížce je opět příkladem lokalizace schopné, na rozdíl třeba od EKF, vyřešit všechny tři druhy lokalizace. Jak už napovídá název, tento způsob dělí prostor na mřížku podle os x a y a také podle úhlu natočení φ. Pravděpodobné polohy robota se reprezentují pomocí histogramu - pravděpodobností polohy robota v jednotlivých buňkách mřížky. Samozřejmě čím jemnější bude dělení prostoru na mřížku, tím přesnější lokalizace bude, ale tím bude také náročnější výpočet.

2.6.6 Monte Carlo lokalizace

V poslední době nejoblíbenější metodou lokalizace je Monte Carlo (MCL). V podstatě je to podobná metoda jako Grid-based lokalizace, jen s tím rozdílem, že pravděpodobnost není reprezentována pomocí histogramu, ale pomocí částic. Jejich počet si můžeme zvolit podle toho, jak velký je prostor, který potřebujeme pokrýt. MCL má spoustu výhod, díky kterým se tato metoda stala v současnosti jednou z nejvíce používaných. Především nevyžaduje, na rozdíl třeba od Kalmanova filtru, aby rozdělení šumu bylo Gaussovo a umožňuje pracovat s více možnými pozicemi robota. Také můžeme v průběhu výpočtu přizpůsobit počet částic tomu, jak dobře je robot lokalizován.

MCL má ovšem také některé nevýhody. Např. pokud bude částic příliš málo, může se stát, že nebude žádná v místě skutečné pozice robota a ten tak ztratí přehled o tom, kde se nachází a místo toho bude za správnou považovat naprosto chybnou polohu. Základní MCL je také nevhodný pro problém uneseného robota, protože po několika měřeních se může stát, že všechny částice budou v místě, kde se robot doopravdy nachází a pokud ho přesuneme jinam, tam už nebude částice žádná a robot nebude mít možnost, jak ji tam vytvořit a bude si myslet, že je na původním, tj. správném místě. V případě příliš přesných senzorů robota určujících jeho polohu (enkodéry, akcelerometry) se také může stát, že algoritmus zcela selže, protože bude generovat vzorky příliš blízko ideálním pozicím robota. Tomuto se lze vyhnout tím, že do pohybu robota budeme uměle vnášet určitý šum. Další možností je přidávat do množiny částic náhodně generované částice [87]. Matematicky je to odůvodnitelné jako velmi malá pravděpodobnost, se kterou může být robot unesen. I když s největší pravděpodobností unesen nebude, zvyšuje náhodné přidávání částic robustnost celého lokalizačního algoritmu. Robustní varianta MCL jménem Mixture Proposal MCL je popsána v [88] a [89]. Ta je pak rozšířena v [23] na lokalizaci i za použití stereokamery.

Během lokalizace je potřeba plánovat pohyb robota. Vyvstává otázka, jak určit pozici, nebo přímo částici, která bude brána za skutečnou polohu robota a tak veškeré plánování bude pro tuto částici $\bar{x}$. Pro uvedený problém existuje několik přístupů.

- Částice s největší vahou - nejintuitivnější přístup. Pro plánování cesty se použije ta

částice, která má aktuálně největší váhu.

$$\bar{x} = x^{max} \tag{2.82}$$

- Vážený průměr

$$\bar{x} = \sum_{j=1}^{M} x_j \omega_j \tag{2.83}$$

- Robustní průměr - vybírají se pouze ty částice, které jsou stejně velké, nebo přinejhorším o ϵ menší než částice s maximální vahou.

$$\bar{x} = \sum_{j=1}^{K} x_j \omega_j :| \; x_j - x^{max} \; | \leq \epsilon \tag{2.84}$$

Algoritmus 5 MonteCarloLocalization

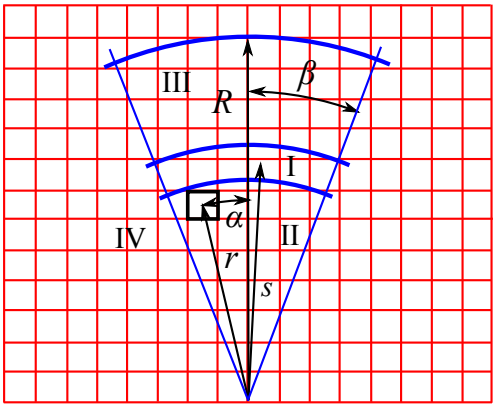
Input: X_{t-1}, u_t, z_t, m
Output: X_t

1: $\overline{X_t} = X_t = 0$
2: **for** $k = 1$ to K **do**
3: $\quad x_t^{[k]} = sample\_motion\_model(u_t, x_{t-1}^{[k]})$
4: $\quad \omega_t^{[k]} = measurement\_model(z_t, x_t^{[k]}, m)$
5: $\quad \overline{X_t} = \overline{X_t} + \langle x_x^{[k]}, \omega_t^{[k]} \rangle$
6: **end for**
7: **for** $k = 1$ to K **do**
8: $\quad$ draw i with probability $\approx \omega_t^{[i]}$
9: $\quad$ add $x_t^{[i]}$ to X_t
10: **end for**
11: **return** X_t

Algoritmus pro MCL (Algoritmus 5) vychází z Bayesova filtru, první cyklus **for** odpovídá části predikce, druhý cyklus **for** odpovídá korekci. V predikci se pro všechny částice K ze znalosti předchozí polohy x_{t-1} a požadovaného řízení u_t vypočítá náhodná nová poloha x_t. Pak se provede měření z_t a spočítá váha částic $\omega_t^{[k]}$. Další cyklus **for** odpovídá části korekce. V něm se podle váhy ω "losují"částice, které se přidají do množiny X_t, a které se pak použijí v příštím kroku $t + 1$. Tím, že se negenerují žádné nové částice, jako např. v genetických algoritmech, zahrnují se vlastně do výpočtu i minulá měření. Kdyby tomu tak totiž nebylo, mohla by díky jednomu špatnému měření snadno zvítězit nově vygenerovaná částice, která je ovšem na úplně jiném místě než robot. Funkce *sample_motion_model()* a *measurement_model()* jsou funkce popsané v kapitole 2.5.

2.7 Mapování

Mapováním se v robotice rozumí tvorba mapy na základě měření ze senzorů. Podstatný rozdíl oproti SLAMu, kde je nepřesné měření i pohyb, je ovšem v tom, že v mapování se měření bere jako nepřesné, ale pozice robota/senzoru je zde brána jako přesná. Za mapování se tedy dá považovat i jen převádění hodnot naměřených senzory do reprezentace mapy.



Obrázek 2.12: Schéma čtyř oblastí sonaru. Podle úhlu α a vzdálenosti r a podle rovnic (2.87) až (2.89) se určí hodnota obsazenosti dané buňky.

Způsob tvorby mapy do značné míry závisí na tom, jaké senzory máme k dispozici a jaký druh mapy chceme vytvořit. Se sonary a lasery se nejčastěji tvoří 2D mřížky obsazenosti. V některých případech, např. mapování zatopených jeskyní [26] se dělají 3D mřížky obsazenosti i z měření sonarů. V posledních letech, s tím, jak se do popředí zájmu dostávají kamery, se vytváří mapy i z těchto dat.

2.7.1 2D mapa

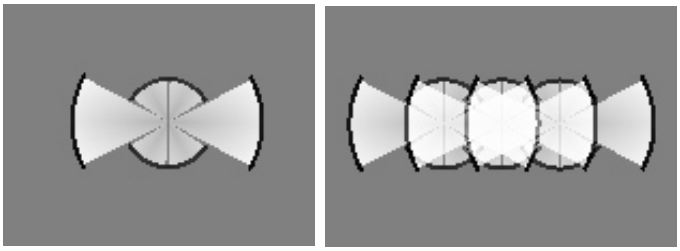
Ve 2D prostoru bývá mapa nejčastěji reprezentována mřížkou obsazenosti. V případě sonarů, ale dá se použít i pro laser, se oblast měření rozdělí do čtyř částí. Část I odpovídá vzdálenosti naměřené sonarem a část II je volná plocha před touto naměřenou vzdáleností. Část III je oblast mezi naměřenou vzdáleností a maximálním dosahem sonaru. Část IV je pak oblast mimo měření sonaru. Nás budou zajímat jen oblasti I a II, protože pouze v nich se mění hodnoty v mřížce. Způsobů, jak vytvořit model sonaru a pomocí něj převést měření na hodnoty jednotlivých buněk je několik; některé jsou popsány např. v [93]. Nejstarším modelem je Elfesův model sonaru [22].

$$P(s \mid Empty) = E(r)A(\alpha) \tag{2.85}$$
$$P(s \mid Occupied) = O(r)A(\alpha) \tag{2.86}$$

$$E(r) = \begin{cases} 1 - \left(\frac{r}{z}\right)^2 & r \in \langle 0, z - \epsilon \rangle \\ 0 & jinak \end{cases} \tag{2.87}$$

$$O(r) = \begin{cases} 1 - \left(\frac{r - z}{\epsilon}\right)^2 & r \in \langle z - \epsilon, z + \epsilon \rangle \\ 0 & jinak \end{cases} \tag{2.88}$$

$$A(\alpha) = \begin{cases} 1 - \left(\frac{2\alpha}{\beta}\right)^2 & \alpha \in \langle -\beta/2, \beta/2 \rangle \\ 0 & jinak \end{cases} \tag{2.89}$$



Obrázek 2.13: Obrázek vlevo ukazuje, jak vypadá mřížka obsazenosti po jednom měření šesti sonary. Vpravo je stejná mřížka po třech měřeních. Dominantní šedá barva představuje pravděpodobnost obsazení buněk rovnu 0,5. To znamená, že o obsazenosti nic nevíme. Čím je šedá světlejší, tím roste pravděpodobnost toho, že je buňka volná. Naopak, čím je šedá tmavší, tím je větší pravděpodobnost, že je buňka obsazená. V tomto obrázku odpovídá jeden pixel jedné buňce.

kde

- z - hodnota vrácená sonarem
- R - max. dosah sonaru
- β - maximální úhel sonaru
- α - úhel buňky od osy sonaru
- r - vzdálenost buňky od sonaru
- ϵ - polovina šířky oblasti I

Kromě modelu sonaru potřebujeme ještě způsob, jak současná měření zakomponovat do měření minulých. K tomu můžeme použít několik různých způsobů. Mezi nejpoužívanější patří pravděpodobnostní způsob, kde se nové hodnoty počítají pomocí Bayesova pravidla. Jiným způsobem může být např. věrohodnostní přístup, kde se využívá Dempster-Shaferovo pravidlo, nebo použití fuzzy množin [93].

V případě použití Bayesova pravidla vypadá algoritmus následovně:

$$P(Occupied \mid s) = \frac{P(s \mid Occupied)P(Occupied)}{P(s \mid Occupied)P(Occupied) + P(s \mid Empty)P(Empty)} \tag{2.90}$$

$P(Occupied \mid s)$ je nová hodnota v buňce obsazenosti, $P(Occupied)$ a $P(Empty)$ jsou předchozí hodnoty v buňkách obsazenosti a $P(s \mid Occupied)$ je pravděpodobnost získaná z modelu sonaru. $P(s \mid Empty)$ je spočítáno jako $1 - P(s \mid Occupied)$.

2.7.2 3D mapa

V případě 3D map bude mapa definována stejně jako měření a to jako sada 3D význačných bodů (landmarků) [51].

$$\begin{aligned} z_t &= \{z_t^i\}_{i=\{1,\dots,N\}} \quad kde \quad z_t^i = \langle \mathbf{X}_t^i, \mathbf{F}_t^i \rangle \\ m_t &= \{m_t^j\}_{j=\{1,\dots,M\}} \quad kde \quad m_t^j = \langle \mathbf{X}_t^i, \mathbf{F}_t^i \rangle \end{aligned} \tag{2.91}$$

N je počet nalezených význačných bodů v jednom měření, M je počet význačných bodů v mapě a t je čas. Každý význačný bod, ať už v mapě nebo z měření, se skládá ze souřadnic bodu a z deskriptoru tohoto bodu. Deskriptor je vektor 128 hodnot popisující bod (viz kapitola 3.4). Protože měření jsou nepřesná, jsou bod $\mathbf{X}$ i deskriptor $\mathbf{F}$ určeny jako náhodná veličina s normálním rozložením a se střední hodnotou (průměrem) μ a kovarianční maticí Σ.

$$\mathbf{X} \sim N(\mu_{\mathbf{X}}, \Sigma_{\mathbf{X}}) \tag{2.92}$$

$$\mathbf{F} \sim N(\mu_{\mathbf{F}}, \Sigma_{\mathbf{F}}) \tag{2.93}$$

Každý význačný bod l tedy bude mít následující strukturu.

$$l \sim N\left(\langle \mu_{\mathbf{X}}, \mu_{\mathbf{F}} \rangle^T, \left(\begin{array}{c|c} \Sigma_{\mathbf{X}} & \mathbf{0} \\ \hline \mathbf{0}^T & \Sigma_{\mathbf{F}} \end{array}\right)\right) =$$

$$N\left(\left(\begin{array}{c} x \\ y \\ z \\ d_1 \\ \vdots \\ d_{128} \end{array}\right), \left(\begin{array}{ccc|ccc} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} & & & \\ \sigma_{yx} & \sigma_y^2 & \sigma_{yz} & & \mathbf{0} & \\ \sigma_{zx} & \sigma_{zy} & \sigma_z^2 & & & \\ \hline & & & \sigma_{d1}^2 & \dots & 0 \\ & \mathbf{0}^T & & \vdots & \ddots & \vdots \\ & & & 0 & \dots & \sigma_{d128}^2 \end{array}\right)\right) \tag{2.94}$$

$\mathbf{0}$ je matice nul o velikosti 3×128, $\mu_{\mathbf{X}}$ a $\mu_{\mathbf{F}}$ jsou střední hodnoty polohy bodu a deskriptoru, $\Sigma_{\mathbf{X}}$ a $\Sigma_{\mathbf{F}}$ jsou pak příslušné kovarianční matice. $\Sigma_{\mathbf{X}}$ se zjistí z rovnice (3.28), hodnotu $\Sigma_{\mathbf{F}}$ je potřeba určit empiricky ze série měření stejných bodů.

2.8 SLAM - simultánní lokalizace a mapování

Spojením lokalizace a mapování vznikne *současná lokalizace a mapování* (Simultaneous Localization and Mapping, SLAM) [21]. Někdy bývá tento problém označován jako podobný problému slepice a vejce, robot se totiž snaží lokalizovat v mapě, kterou si teprve sám vytváří.

Nejstarším a patrně nejvlivnějším algoritmem je zde EKF SLAM (Extended Kalman Filter SLAM). První popisy tohoto algoritmu se objevily v pracech Smithe a Cheesemana [80], [81] a pak Moutaliera a Chatily [52], [53] a Leonarda a Durranta-Whytea [40], [41].

Tento algoritmus má však některá vážná omezení. Jedno z nich je, že vytvořené mapy jsou tzv. feature-based, tzn. jsou založeny na bodových značkách - *landmarcích*, což jsou význačné objekty v prostředí robota, např. uměle vytvořené značky, které robot snadno rozpozná. V opačném případě musí být mapovací algoritmus vybaven nějakým druhem rozpoznávání např. rohů chodeb, spojování chodeb, dveří, oken, atd., což pak bude tvořit přirozené landmarky. Toto omezení ovšem neplatí v případě použití kamery a detekování význačných bodů. V tomto případě totiž robot nedetekuje žádné uměle vytvořené značky, ale přirozeně význačné body. Tyto body mají navíc nějaký deskriptor, a tak je robot schopen je rozpoznat znovu a přiřadit je k bodům už jednou detekovaným.

Dalším omezením tohoto algoritmu je nutnost splnění předpokladu Gaussova šumu a také množství šumu nesmí být moc velké, jinak by do výpočtu zanášelo příliš velkou chybu.

EKF SLAM existuje ve dvou druzích, v prvním druhu máme landmarky nějak identifikované a v případě, že robot narazí na další landmark, ví, zda se jedná o nový, nebo už jednou rozpoznaný landmark (ang. known data association). V druhém případě landmarky nijak identifikovány nejsou a robot se sám musí rozhodnout (např. pomocí metody maximum likelihood), zda tento landmark už ve své mapě má a který to je, případně zda se jedná o landmark nový (ang. unknown data association). Toto je právě případ, který nastane při detekci význačných bodů kamerou.

Dalším důležitým algoritmem je FastSLAM, ve kterém se používá částicový filtr. Tento algoritmus byl poprvé popsán v článcích M. Montemerla a kol. [49], [48]. Důležitou práci v oblasti použití částicových filtrů udělal také S. Thrun a jeho spolupracovníci. Ve shrnutích [85] a [86] popsal pravděpodobnostní algoritmy a jejich použití v robotice. Především ale se svými kolegy napsal řadu článků zabývajících se tvorbou mapy pomocí laserového scanneru, které používal robot autonomně mapující opuštěné doly [90], [91]. Poznatky z této práce pak shrnuli spolu s W. Burgardem a D. Foxem ve vynikající knize Probabilistic Robotics [87].

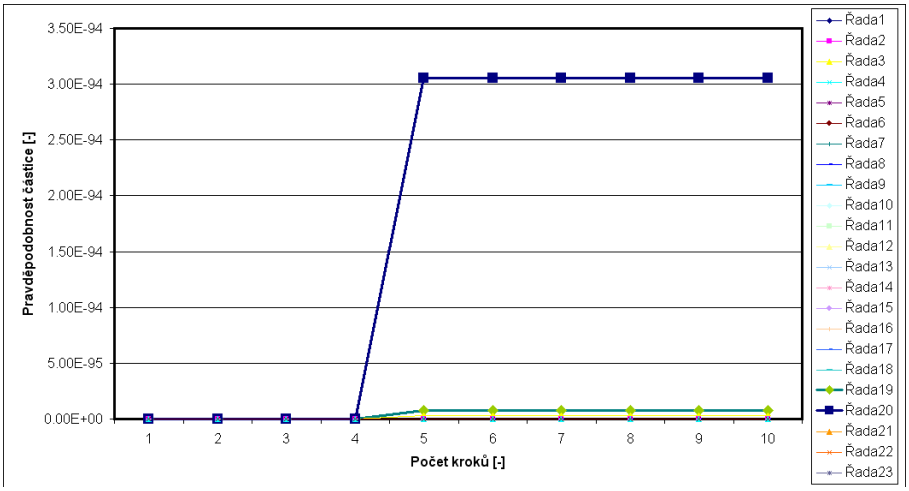
Během posledního desetiletí se pro tvorbu mapy začínají používat i kamery. Důvody jsou celkem prosté. Je to především nízká cena a váha, což jsou vlastnosti pro malé roboty velice důležité. Na druhou stranu nevýhodou použití kamer je nutnost obraz nejprve zpracovat a vyextrahovat z něj důležité informace, což je v tomto případě především poloha význačných bodů vzhledem ke kameře. Tyto algoritmy se pak vzhledem k použití kamery nazývají VisualSLAM a dají se rozdělit na dva druhy. Prvním druhem je použití pouze jedné kamery, druhým pak použití dvou, tzv. stereokamery. Použití jedné kamery rozpracoval v sérii publikací především A. J. Davidson [17], [18] a [19]. Použití stereokamery a Rao-Blackwellovy varianty SLAMu se hodně věnují R. Sim a J. J. Little v rozsáhlé sérii článků [72], [73], [74], [75] a [24]. Kromě nejčastějšího případu jedné nebo dvou kamer, existují i systémy, které používají trojici kamer, příkladem může být práce [70], kde je celkem detailní popis lokalizace a tvorby mapy pomocí SIFT deskriptorů [43]. Tyto deskriptory se používají relativně často, například i v práci T. Barfoota [5] nebo v [51]. Dobrý přehled algoritmů pro oblast VisualSLAMu je v [60] a především pak v [8].

Algoritmus 6 FastSLAM

Input: (X_{t-1}, u_t, z_t)
Output: X_t

1: $\overline{X_t} = X_t = 0$
2: **for** $k = 1$ to M **do**
3: $\quad x_t^{[k]} = sample\_motion\_model(u_t, x_{t-1}^{[k]})$
4: $\quad \omega_t^{[k]} = measurement\_model(z_t, x_t^{[k]}, m_{t-1})$
5: $\quad m_t^{[k]} = updated\_occupancy\_grid(z_t, x_t^{[k]}, m_{t-1}^{[k]})$
6: $\quad \overline{X_t} = \overline{X_t} + \langle x_x^{[m]}, \omega_t^{[m]} \rangle$
7: **end for**
8: **for** $k = 1$ to M **do**
9: $\quad$ draw i with probability $\approx \omega_t^{[i]}$
10: $\quad$ add $\langle x_x^{[k]}, m_t^{[k]} \rangle$ to X_t
11: **end for**
12: **return** X_t

Algoritmus pro FastSLAM je v podstatě stejný, jako algoritmus pro Monte Carlo lokalizaci. Jediným rozdílem je to, že FastSLAM obsahuje navíc funkci *updated_occupancy_grid()*,



Obrázek 2.14: Graf vývoje pravděpodobnosti jednotlivých částic. Na grafu je vidět nárůst pravděpodobnosti částice v kroku 5, který odpovídá návratu částice do už zmapovaného prostoru.

která aktualizuje mapu novým měřením. Pro případ mřížky obsazenosti je tato funkce v podstatě popsána v sekci Mapování. Obecně se dá říci, že přidá nová měření do stávající mapy. To jakým způsobem a na základě jakých pravidel se to provede, už závisí na použitém druhu mapy.

Problém SLAMu a mapování obecně je v tom, že pokud je pohyb robota nepřesný, robot svoje měření (i když ta mohou být přesná) vkládá do mapy na špatná místa, čímž je pak mapa také nepřesná. Jediným způsobem, jak takovouto mapu opravit, je návrat robota (ovšem jinou cestou) na místo, kde už jednou byl. V ideálním případě je to to místo, kde mapování začal. Tam je totiž jeho poloha nejpřesnější, protože jsme si ji sami zvolili, nejčastěji jako bod [0, 0]. Takovéto akci se říká *uzavření smyčky* (ang. loop-closing). Aby k ní mohlo dojít, je třeba splnit dvě podmínky. Za prvé robot musí rozpoznat, že dané místo už jednou zmapoval. A za druhé, prostředí musí mít takový tvar, aby se dalo do výchozího bodu vrátit jinou cestou.

Splnění první podmínky záleží především na reprezentaci mapy. Pokud bude mapa reprezentována pomocí mřížky obsazenosti, potom to, že jsme se navrátili do už jednou zmapovaného místa poznáme tak, že náhle vzroste pravděpodobnost měření $p(z_t \mid x_t, m)$. Nové měření totiž nejenže bude "pasovat"na předchozí pozici robota, ale částečně na některém místě překryje už vytvořenou mapu, což způsobí náhlý vzrůst pravděpodobnosti měření. Příklad takového vzrůstu pravděpodobnosti částice je na obrázku 2.14. Důležité ovšem je, aby alespoň jedna z částic filtru byla na správném místě. V případě, že je mapa reprezentována pomocí landmarků, je potřeba, aby se dal každý jednoznačně identifikovat. Robot pozná, že je na už jednou prozkoumaném místě tím, že znovu detekuje landmarky, které už má ve své mapě. Pokud je to možné, je vhodné, aby k převzorkovávání částic došlo právě po detekci uzavření smyčky. Když k nim dochází i v jiných okamžicích, znamená to pro robota ztrátu diverzity částic, což může být nežádoucí.

Splnění druhé podmínky je poněkud složitější. Je jasné, že robot nemůže nijak zajistit, aby měl prostor takový tvar, aby umožňoval navrácení se do předem prozkoumaného místa jinou cestou. Robot ani nemá o tvaru prostoru žádnou informaci, a tak neví, zda tam takové smyčky existují, případně, jak jsou dlouhé. Ve výsledku tak neví, jak daleko se má

po nějaké cestě pohybovat a zda se už nemá raději stejnou cestou vrátit a zkusit jinou, protože daná cesta netvoří smyčku a pravděpodobně vede z mapované oblasti ven. Z hlediska tvorby mapy pomocí SLAMu je totiž nejvhodnější, aby robot nejprve zmapoval ty největší smyčky a až teprve potom se věnoval detailnějšímu průzkumu prostředí. V případě použití částicového filtru totiž dochází k tomu, že vzhledem k nepřesnému pohybu robota se od sebe částice navzájem vzdalují. Pokud bychom příliš často převzorkovávali, mohlo by dojít k situaci, kdy se robot sice vrátí do už jednou prozkoumaného místa, ale protože této poloze neodpovídá žádná částice, nebude to rozpoznáno a mapa se tak nepovede. Proto je dobré převzorkovávat co nejméně. Pokud to uděláme a robot ujede velkou[4] vzdálenost, může opět dojít k situaci, kdy se vrátí na už jednou zmapované místo, ale nerozpozná to, protože na tomto místě nebude žádná částice. V ideálním případě by tak robot měl zmapovat nejprve tu smyčku, která tvoří hranici zkoumaného prostoru a teprve potom ostatní smyčky, zase ideálně od největších po nejmenší. Zde ovšem předpokládáme, že žádná cesta nevede z tohoto prostoru ven. Protože to však nelze splnit, je toto pouze ideální případ.

2.9 Plánování pohybu

Poslední částí, která je potřeba pro tvorbu mapy, je plánování pohybu robota. Mapu je sice možno vytvořit i s dálkově ovládaným robotem, nebo v případě testování příslušných algoritmů můžeme kamery/robota držet v rukou a pohybovat s nimi sami [60]. První přístup zvolili v už zmiňovaném článku o mapování podzemních dolů [90].

Pohybování kamerou, drženou v rukách, se často používá mezi komunitou zaměřenou čistě na počítačové vidění. Tyto algoritmy jsou určené spíše pro vytvoření modelu malého objektu, který se kamerou snímá kolem dokola, a kde případná nepřesnost nebude mít na dokonalost modelu moc velký vliv. Uvedené algoritmy lze využít také u větších objektů, jako jsou průčelí budov, různé plastiky atd. [67], [68]. Poslední a nejambicióznější je tvorba modelů částí ulic, ovšem v tomto případě netvoří model uzavřenou oblast, tzn. kamera se nevrací do místa, kde už jednou byla; případně se kromě kamery používá například i GPS [62].

Pro vytvoření opravdu autonomního robota je potřeba, aby si veškerou cestu plánoval robot sám. Plánování pohybu rozdělíme na dvě části. První částí je samotné plánování cesty k cílovému bodu. Zde se robot rozhoduje, kudy pojede. Druhou částí je vybírání cílových bodů, kde se robot rozhoduje, kam pojede. Tato druhá část tvoří jakousi vyšší úroveň řízení robota. V anglicky psané literatuře se jí říká "exploration", tzn. něco jako průzkum nebo objevování.

2.9.1 Plánování cesty

Algoritmů pro plánování cesty je velké množství. Nejznámějšími jsou algoritmy pro hledání cesty v grafu, jako např. A*, BFS, DFS, atd. Ty se ale samostatně v robotice příliš nepoužívají, spíše se používají v kombinaci s jinými metodami. Velké množství algoritmů pro plánování cesty pro roboty je popsáno v knize Robot Motion Planning od H. Choseta a dalších [14] nebo v knize Planning Algorithms od S. M. LaValleho [38].

Jednou z metod pro plánování cesty jsou *potenciálová pole* [14], kde se oblast možného pohybu robota rozdělí na jednotlivé buňky a pro všechny se spočítá hodnota, která klesá (nebo roste) od polohy robota směrem k cíli. Nevýhodou těchto metod je možné uvíznutí

[4] Velkou myslíme vzhledem k nepřesnosti jeho pohybu.

v lokálním minimu. Pro vyřešení tohoto problému se používají např. harmonická potenciálová pole (viz např. [65]). Problémem, který se však dále objevuje, je velká výpočetní náročnost. Abychom nalezli cestu pro robota, musíme vypočítat soustavu rovnic, kde pro každou buňku potřebujeme jednu rovnici. Výhodou je naopak to, že pokud se cíl nezmění, můžeme vypočítat pole jen jednou a používat je opakovaně. Navíc, cesta se jednoduše v poli nalezne tak, že prostě najdeme buňku, která má v okolí (například osmiokolí zkoumané buňky) nejmenší hodnotu.

Další metodou pro plánování cesty je *přesný rozklad na buňky* (ang. Exact Cell Decomposition). V tomto případě se volný prostor rozdělí na jednotlivé buňky. Ty jsou spojeny do grafu, kde jednotlivé vrcholy grafu tvoří buňky a společná hrana mezi dvěma buňkami pak tvoří spojnici mezi dvěma vrcholy v grafu. Plánování cesty se provede tak, že se nejprve identifikují ty buňky, ve kterých je obsažen počáteční a koncový bod pohybu robota. Potom se pomocí hledání cesty v grafu nalezne cesta mezi počátečním a koncovým bodem. Základním algoritmem zde je lichoběžníková dekompozice, která, jak už název napovídá, dělí volný prostor na lichoběžníky. Tento základní algoritmus není příliš vhodný v případě, že chceme pokrýt pohybem robota celý volný prostor (např. u robotických sekaček či vysavačů, nebo v případě odminovávacích robotů). V tomto případě se používá rozšíření tohoto algoritmu, nazývané *boustrophedon*, které vytvoří sice složitější buňky, ale protože je jich méně, umožní tak pokrýt celý prostor kratší cestou.

Jinou metodou pro plánování cesty jsou roadmapy. Zde je volný prostor pokryt soustavou cest, mezi kterými se pak hledá ta nejvhodnější pro robota. Mezi nejznámější roadmapy patří mapy viditelnosti a Voronoiovy mapy. V případě map viditelností se pospojují vrcholy polygonů, které vymezují obsazený prostor a cesta pro robota se hledá mezi těmito spojnicemi. U Voronoiových map, neboli diagramů, se cesty vytvoří jako spojnice míst, které mají stejnou vzdálenost mezi dvěma překážkami. Cesta se v tomto případě hledá obdobně, jako u map viditelnosti.

Poslední zde popsanou metodou je pravděpodobnostní plánování. Vzhledem k jeho většímu významu mu bude věnována samostatná kapitola.

2.9.2 Pravděpodobnostní plánování cesty

Protože stavový prostor všech možných poloh robota je v podstatě nekonečný, nemůžeme ho dost dobře prohledat celý. Proto vzniklo pravděpodobnostní plánování cesty, jehož podstatou je navzorkování možných poloh robota a pospojování těchto cest do grafu [14], [59]. Vyhledávání cesty opět probíhá v tomto grafu.

Pravděpodobnostní algoritmy se dělí podle toho, zda graf, který vytvoří, může být použit opakovaně ke hledání cesty mezi různými body, nebo jen jednou mezi dvěma body a pro další vyhledávání je potřeba zkonstruovat nový graf.

- Vícedotazové algoritmy - tyto algoritmy nejprve vytvoří graf, zachycující celý prostor. K tomu se pak postupně připojují startovací a cílové body. Jejich výhoda spočívá v tom, že nemusíme znovu vytvářet nový graf, pokud se změní polohy startu či cíle. Do této kategorie patří základní pravděpodobnostní algoritmus PRM (Probabilistic Roadmap).

- Jednodotazové algoritmy - vytvoří graf mezi dvěma konkrétními body v prostoru. Pokud víme, že cestu v daném prostoru nebudeme hledat opakovaně, je výhodnější použít jednodotazové algoritmy, protože ty nevzorkují celý prostor, ale soustředí se

na oblast mezi startem a cílem. Sem patří algoritmy RRT (Rapidly-Exploring Random Trees) a EST (Expansive-Spaces Trees).

- Kombinované algoritmy - kombinují výhody obou předchozích způsobů a měly by tak patřit k nejvýkonnějším algoritmům. Vytvářejí graf, pokrývající celý stavový prostor, ovšem používají k tomu jednodotazové algoritmy. Příkladem tohoto algoritmu je SRT (Sampling-Based Roadmap of Trees).

Vzorkování ve stavovém prostoru

Stavový prostor je množina všech možných poloh robota. *Volný stavový prostor* pak obsahuje takové polohy robota, které nekolidují s žádnou překážkou a ani s robotem samotným. Kolika rozměrný stavový prostor bude, záleží na tom, kolik potřebujeme proměnných na jednoznačné popsání polohy robota v prostoru. Pokud by robot byl koule nebo bod v prostoru, stačí nám tři stavové proměnné a prostor tedy bude třírozměrný. Pokud má robot nějaké pohyblivé části, pro každou pohyblivou část musíme přidat jeden rozměr. Pokud by robotem bylo např. autíčko s natáčecími předními koly a pohybující se v rovině, byl by prostor čtyřrozměrný - tři rozměry na určení polohy v prostoru a čtvrtý rozměr na určení natočení koleček. Rozměry robota se někdy přidávají k rozměrům překážek (překážky se zvětší o jeho rozměr) a my tak můžeme za robota ve stavovém prostoru považovat pouze bod.

Nejčastějším způsobem vzorkování je vzorkování s normálním rozložením pravděpodobnosti. Tímto způsobem pokryjeme prostor pravidelně jednotlivými vzorky. Tento způsob je vhodný jen pro jednoduchá prostředí. Pokud se totiž robot bude pohybovat v blízkosti překážek nebo v místech úzkých průchodů jako jsou např. dveře, bude tento způsob nedostatečný. V takovýchto místech je potřeba, aby byl pohyb robota jemnější, tzn. aby tam vzorkování bylo co nejhustší. Tato nevýhoda může být odstraněna vhodnou změnou způsobu vzorkování. Například budeme generovat dvojice bodů a pokud oba body budou ležet v obsazeném prostoru a střed jejich spojnice ve volném prostoru, přidáme tento střed do množiny vzorků.

Lokální plánovač

Lokální plánovač spojuje jednotlivé vzorky do grafu. Podle typu algoritmu se toto provádí buďto až po dostatečném navzorkování celého prostoru (např. v případě algoritmu PRM) nebo vzorky připojujeme ke grafu ihned po tom, co je vygenerujeme (algoritmy EST a RRT). Úkolem lokálního plánovače je zajistit, aby cesta mezi dvěma spojovanými vzorky byla volná. Tzn. plánovač postupně testuje body na spojnici mezi těmito dvěma zkoumanými vzorky až do určitého rozlišení. Toto rozlišení by mělo být menší, než je nejmenší překážka v prostoru. Testování se pak může provést dvěma způsoby. Prvním je testovat prostor postupně bod po bodu mezi vzorky A a B s krokem d. Druhým způsobem je rozdělit hranu mezi vzorky A a B na polovinu a v tomto bodě C provést nové testování, pak opět rozdělit hranu A a C na poloviny a zase ve středu provést testování a to samé s hranou C a B a takhle rekurzivně pokračovat až do dosažení určité přesnosti nebo do nalezení kolidujícího vzorku. V praxi se ukazuje, že i když je druhý způsob složitější, vede k lepším výsledkům.

Vícedotazové algoritmy

Algoritmus PRM, coby příklad vícedotazovacích algoritmů, pracuje ve dvou fázích. V první fázi je navzorkován prostor a vytvořen graf a ve druhé fázi je k tomuto grafu připojen počáteční a cílový bod a vyhledána cesta. Vzorkování probíhá tak, jak je popsáno

v předešlé části. Až je konfigurační prostor dostatečně navzorkován (např. až je dosaženo požadovaného počtu vzorků n), lokální plánovač začne vzorky spojovat do grafu. Postupně se vybírají vzorky a najde se k nim k nejbližších dalších vzorků. Pokud mezi těmito dvěma vzorky q a q' existuje volná cesta, tzn. $\Delta(q, q') \neq NIL$ a zároveň už hrana není součástí grafu, jsou spojeny hranou. Po ukončení připojování vzorků následuje druhá fáze, kdy je ke grafu připojen i start a cíl a je vyhledána cesta mezi těmito dvěma body. Pro její vyhledání je možno použít některého z algoritmů pro hledání cesty v grafu, nejčastější A* nebo Dijkstrova algoritmu.

V případě nevhodně zvolených hodnot n a k se může stát, že do grafu nebudou připojeny všechny vzorky nebo se bude graf skládat z více oddělených částí. V obou případech může nastat situace, kdy požadovaná cesta mezi startem a cílem nebude nalezena. Pak je potřeba navzorkovat prostor větším množstvím vzorků. Takováto situace může nastat například v prostoru, ve kterém existují úzké průchody.

Jednodotazové algoritmy

Jednodotazové algoritmy (EST a RRT) negenerují vrcholy grafu náhodně po celém prostoru, ale začínají z počátečního uzlu a případně zároveň i z cílového. V blízkosti grafu je vygenerován další vrchol q_{new} a ten je ihned připojen ke grafu (samozřejmě pouze pokud mezi vrcholem q_{new} a vybraným vrcholem z grafu existuje nekolizní cesta). Vzhledem k tomu, že se nové vrcholy generují v blízkosti existujícího grafu a snaží se připojit k cíli, nebo ke grafu generovaného z cíle, není nutné vygenerovanými vrcholy zaplnit celý stavový prostor jako je tomu u vícedotazových algoritmů. Jednodotazové algoritmy se často používají pro řešení kinodynamických úloh, což je takový typ úloh, v nichž se pro hledání cesty uvažuje i s rychlostí a hmotností robota.

Kombinované algoritmy

Kombinací obou předchozích algoritmů je algoritmus SRT (Sampling-Based Roadmap of Trees). Tento algoritmus totiž vytváří graf v celém stavovém prostoru, pro spojování vrcholů však používá jednodotazové algoritmy. Může tak být použit pro opakované hledání cesty s měnícími se polohami startu i cíle, ale i pro jeden dotaz, kde může být efektivnější, než jednodotazové algoritmy. Zajímavou vlastností tohoto algoritmu je to, že vhodným nastavením konstant můžeme dostat všechny tři předchozí algoritmy.

2.9.3 Výběr cílů

Ještě předtím než naplánujeme cestu ke konkrétnímu bodu, musíme tento bod nějakým způsobem vybrat. Na pořadí výběru cílových bodů záleží kvalita vytvářené mapy. Pokud se mapa vytváří pomocí algoritmu SLAM, je lepší, když se robot nepohybuje náhodně, ale pokud se systematicky snaží objevit co nejrychleji co největší území. Pokud by se snažil zmapovat území co nejpřesněji hned od začátku, musel by velice často měnit směr svého pohybu, což by mělo negativní vliv na přesnost určení jeho pozice a tím i na přesnost mapy.

Výběr cílů (v ang. exploration) se nejčastěji dělí do tří oblastí:

- Objevování při tvorbě mapy - robot autonomně vytváří mapu okolí a vybírá taková místa, kde o svém okolí získá co nejvíce nových informací, tzn. snaží se dostat do neprobádaných území.
- Hledání osob ve známé mapě (ang. pursuit-evasion problem) - v tomto případě je úkolem robota ve známém prostředí nalézt pohybující se osobu. Vyžaduje často opětovné

navštívení už jednou navšíveného místa.

- Aktivní lokalizace ve známé mapě - robot se potřebuje lokalizovat v mapě, a tak sám vybírá taková místa, kde se může lokalizovat co nejrychleji.

Problém v plánování pohybu je ten, že robot má na výběr většinou z velkého množství cílů, ke kterým se může vydat a s objevováním dalších cílů množství možných pohybových sekvencí roste do neúnosných mezí. To má za následek nemožnost používání složitějších plánovacích algoritmů, které by naplánovaly pohyb robota více do budoucna. Objevování musí být velice reaktivní a robot musí často po pár pohybových sekvencích změnit svůj cíl. Většina plánovacích algoritmů tak patří mezi tzv. *greedy* algoritmy, kdy si robot vybere tu akci, která je pro něj nejvýhodnější v krátkodobém horizontu.

Příklad plánování pohybu v mapě reprezentované pomocí landmarků je například v práci [84]. Jiný způsob je uvedený v článku [34], kde jsou vybírána místa, odkud bude mít robot, vybavený laserovým scannerem, nejlepší výhled a bude tak z nich schopen vytvořit co nejlepší mapu okolí.

V případě mřížek obsazenosti je nejčastěji používanou technikou tzv. *frontier-based exploration*, která je posána např. v pracech B. Yamauchiho [96], [97]. Podobná technika je použita i v článku [75], kde používají stereokamery pro tvorbu mapy, ta má ovšem podobu mřížek obsazenosti.

Sofistikovanějším přístupem je použití entropie [87], zvláště v už zmíněném případě tvorby mapy pomocí SLAM algoritmu. Entropie je největší v okamžiku, kdy o mapě okolí nevíme vůbec nic a v případě mřížek obsazenosti by tedy všechny buňky měly hodnotu 0,5. Cílem robota tedy je vytvořit mapu co nejpřesnější a tím entropii minimalizovat. V případě SLAM algoritmu je entropie rozdělena do dvou částí. První odpovídá očekávané entropii mapy a druhá entropii polohy robota. V případě tvorby mapy totiž dochází ke dvěma protichůdným věcem. Na jedné straně chceme mít co nejpřesnější mapu co největšího území, na druhé straně chceme, aby informace o pozici robota byla co nejpřesnější. Ta je ovšem nejpřesnější v okamžiku začátku tvorby mapy, protože v tom okamžiku známe polohu robota naprosto přesně (volíme si ji totiž sami, nejčastěji na souřadnicích [0, 0]) a s každým další robotovým pohybem je méně a méně přesná. Pokud bychom tedy chtěli minimalizovat entropii robotovy pozice, nesměl by se robot vůbec pohnout a jediná mapa, kterou bychom vytvořili, by byla ta z jeho senzorů v této pozici. Proti tomu jde naštěstí druhá část entropie a tou je snaha robota zmapovat co největší prostor. Ve výsledku tak pohyb robota vypadá tak, že robot objevuje neznámé území, ale jednou za čas získá navrch ta část entropie, která minimalizuje nejistotu polohy a robot se tak vrátí do místa, které už navštívil, aby si upřesnil svoji polohu.

Kapitola 3

Zpracování obrazu

V následující kapitole bude ukázán matematický model kamery a popsán způsob kalibrace kamery a stereokamery. Dále budou uvedeny metody používané pro nalezení a spárování význačných bodů a výpočet jejich 3D polohy v prostoru. Nakonec bude popsána Delaunayho triangulace a způsob vytváření 3D modelů.

3.1 Model stereokamery

Kamera je obvykle tvořena optickou soustavou (sada čoček) a něčím, na co se promítá obraz, dříve to byl film, v současnosti je to většinou CCD nebo CMOS čip. Aby se dal tento proces popsat matematicky, používá se zjednodušený model, tzv. *dírková kamera* [11], [82].

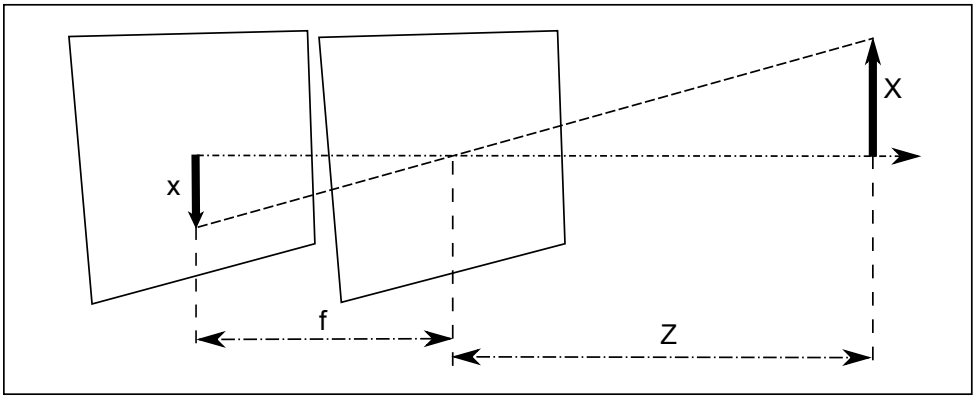
3.1.1 Model dírkové kamery

V kamerách a ve fotoaparátech se většinou používá objektiv s několika sadami čoček, ať už spojek nebo rozptylek. My budeme uvažovat pouze jednu čočku, spojku, a to takovou, která se nazývá *tenká čočka* (ang. thin lens) [44]. Je to čočka, jejíž tloušťka je zanedbatelná oproti její ohniskové vzdálenosti. V tomto případě je čočka definovaná svojí optickou osou a rovinou k ní kolmou, nazývanou ohnisková rovina. Tenká čočka má dva parametry, ohniskovou vzdálenost f a průměr d. Dále má dvě charakteristické vlastnosti. První vlastností je, že všechny paprsky, které procházejí čočkou rovnoběžně s optickou osou se protínají v ohnisku. Druhá vlastnost je, že paprsky, které procházejí optickým středem čočky o, se nelámou. Pokud zmenšujeme poloměr čočky až k nule, tzn. jako bychom zmenšovali otvor do kamery (zvětšovali clonu u fotoaparátu), procházejí všechny paprsky těsně okolo optického středu čočky (teoreticky procházejí optickým středem) a nelámou se. Tímto zjednodušením se dostáváme k *ideálnímu modelu dírkové kamery* (ang. ideal pinhole camera model).

Předpokládejme, že globální souřadnice a souřadnice kamery jsou totožné, potom má bod $\mathbf{X}$ střed v optickém středu o a osa z je v optické ose kamery. Bod v prostoru $\mathbf{X} = [X, Y, Z]^T$ se tak zobrazí do bodu $\mathbf{x} = [x, y]^T$ na obrazové rovině. Z podobných trojúhelníků (obr. 3.1) můžeme spočítat souřadnice x, y bodu $\mathbf{x}$.

$$-x = f\frac{X}{Z} \tag{3.1}$$

$$-y = f\frac{Y}{Z} \tag{3.2}$$



Obrázek 3.1: Zjednodušené schéma dírkové kamery. Bod **X** se promítne do bodu **x** podle rovnice 3.1.

Souřadnice *x, y* jsou záporné, což nám vyjadřuje to, že obraz bude převrácený. Abychom z převráceného obrazu udělali normální, jednoduše umístíme obrazovou rovinu před optický střed kamery, tzn. jako bychom změnili znaménko u ohniska z *+f* na *-f*. Tím se nám také změní znaménka u souřadnic x a y a navíc se výpočty stanou jednodušší. Bod, který je v průsečíku optické osy a obrazové roviny teď bude představovat tzv. *hlavní bod* (ang. principal point). Všechny paprsky se stejně jako předtím budou sbíhat v *ohnisku kamery* (ang. focal point).

Rovnice (3.1) a (3.2) můžeme přepsat na

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix} \tag{3.3}$$

$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.4}$$

Souřadnice Z (hloubka bodu) je obvykle neznámá, a tak ji můžeme vyjádřit skalárem λ. Matici obsahující ohniskové vzdálenosti můžeme ještě přepsat na

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.5}$$

kde jednotková matice je nazývána *standardní projekční matice*. Výsledná rovnice modelu dírkové kamery bude mít tvar

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.6}$$

3.1.2 Intrinsické parametry

Pokud zmenšíme otvor do kamery, začne být dominantní ohyb světla místo lomu světla, se kterým jsme dosud počítali a náš model už nebude úplně správný. Navíc pokud zmenšíme otvor do kamery až k nule, nebude na obrazovou plochu dopadat žádné světlo a tudíž se nevytvoří žádný obraz. Proto se ve skutečném světě používají čočky, které umožňují směrovat na jeden bod více světla. To nás ovšem nutí k použití složitějšího geometrického popisu, než je ten u dírkové kamery a také způsobuje problémy jako je deformace obrazu z důvodu nedokonalosti čoček. Aby náš popis kamery byl co nejblíže realitě, zavádí se pojem *vnitřní parametry* (intrinsické) kamery [44]. Ty také umožňují přepočítat polohu bodu z metrického systému na polohu bodu v pixelech, kde navíc počátek soustavy souřadnic začíná v levém horním rohu obrazu.

Nejprve tedy přepočítáme souřadnice z metrického systému na pixely, k tomu nám poslouží následující matice

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} s_x & \alpha \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.7}$$

Hodnoty s_x a s_y závisí na velikosti pixelů v metrických jednotkách. V případě, že jsou pixely čtvercové, jsou si tyto dvě hodnoty rovny. Hodnota α vyjadřuje zkosení pixelů na čipu, tato hodnota se ovšem u moderních čipů blíží nule, a proto ji nebudeme uvažovat.

Dalším krokem je posunutí soustavy souřadnic ze středu obrázku do levého horního okraje. To provedeme tak, že k hodnotám x_s a y_s přičteme hodnoty o_x a o_y, což je vzdálenost v pixelech hlavního bodu od počátku soustavy souřadnic obrázku (v podstatě by to měly být souřadnice středu obrázku, v praxi tomu tak ale obvykle není).

$$x' = x_s + o_x \tag{3.8}$$
$$y' = y_s + o_y \tag{3.9}$$

V maticovém tvaru pak

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.10}$$

Pokud zkombinujeme předchozí projekční model s tímto modelem, dostaneme transformační model mezi homogenními souřadnicemi 3D bodu v prostoru a homogenními souřadnicemi 2D bodu obrazu vyjádřeného v pixelech.

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.11}$$

V praxi si většinou volíme globální soustavu souřadnic shodnou se soustavou souřadnic kamery, takže odpovídající maticovou transformaci můžeme vypustit. Zbylé tři matice se

pak vynásobí a dostaneme tak výslednou rovnici

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & 0 & o_x & 0 \\ 0 & fs_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.12}$$

Podle [11] se nedají parametry f, s_x a s_y zjistit přímo a vyjadřují se proto dohromady.

3.2 Parametry zkreslení

V praxi je výroba kamery s minimálním zkreslením velice nákladná (jsou to především dražší fotoaparáty a kamery a ani tam se vždy zkreslení nepodaří vyhnout), a tak je potřeba obraz před vlastním zobrazením na monitoru opravit. Nejčastějšími druhy zkreslení jsou radiální a tangenciální [11].

Radiální zkreslení se projevuje ohýbáním rovných čar, především na krajích obrazu. Často se mu také říká soudkové zkreslení a typickým příkladem může být obraz z objektivů typu rybí oko. Toto zkreslení je nulové ve středu obrazu a směrem ke krajům se zvětšuje. Charakterizuje se prvními dvěma, nebo třemi (v případě velkého zkreslení) parametry Taylorovy řady. Nové polohy bodů se tak určí z následující dvojice rovnic.

$$x_{corr} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.13}$$

$$y_{corr} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6), \tag{3.14}$$

kde $r^2 = x^2 + y^2$

Tangenciální zkreslení je důsledkem ne zcela rovnoběžné polohy čipu a čočky. Charakterizuje se dvěma parametry p_1 a p_2.

$$x_{corr} = x + [2p_1 y + p_2(r^2 + 2x^2)] \tag{3.15}$$

$$y_{corr} = y + [p_1(r^2 + 2y^2) + 2p_2 x]. \tag{3.16}$$

Zkreslení kamery se tak udává pomocí vektoru obsahujícího parametry k_1, k_2, k_3, p_1 a p_2.

3.3 Kalibrace stereokamery

Kalibrace kamery se provádí proto, abychom zjistili parametry popisované v předchozích dvou kapitolách a mohli tak opravit zkreslení kamery a dále zjistili ohniskovou vzdálenost a střed obrazu. V případě stereokamery musíme ještě navíc spočítat vzájemnou polohu obou kamer tvořících stereokameru. V praxi se kalibrace provádí většinou pomocí kalibrační šachovnice. Stereokameře (nebo kameře) se předloží série fotografií šachovnice s předem známým počtem políček a jejich známou velikostí. Z těchto hodnot se spočítají přesné souřadnice bodů šachovnice v prostoru.

Kalibrace kamery je dobře popsána např. v [11] a vlastně znamená výpočet čtyř vnitřních parametrů $f_x s_x$, $f_y s_y$, c_x a c_y z rovnice (3.11) a pěti parametrů zkreslení k_1, k_2, k_3, p_1 a p_2. Výpočet se provede tím, že se body v prostoru přepočítají tak, aby odpovídaly těm samým bodům na snímači. Body v prostoru se většinou umístí do jedné roviny (tedy na šachovnici). Tento přepočet z jedné roviny do druhé je pak v počítačové grafice nazýván

rovinná homografie. V podstatě se přepočítá poloha jedné souřadné soustavy do druhé. Pokud si bod v prostoru označíme Q a odpovídající bod na snímači q, jsou jejich homogenní souřadnice

$$Q = [\ x_Q \quad y_Q \quad z_Q \quad 1\ ]^T \tag{3.17}$$

$$q = [\ x_q \quad y_q \quad 1\ ]^T \tag{3.18}$$

přepočet pomocí homografie tedy bude

$$q = sHQ \tag{3.19}$$

kde s je měřítko zvětšení a H je matice homografie. Vzájemnou polohu spočítáme tak, že souřadnou soustavu s bodem Q orotujeme a posuneme do počátku soustavy souřadnic bodu q. Tato rotace a posunutí se vypočítají stejně, jako v sekci o pohybu robota (kapitola 2.5.1). Jsou určeny třemi neznámými parametry rotace a třemi neznámými parametry posunutí. Označíme-li je

$$W = [\ R \quad t\ ] \tag{3.20}$$

můžeme přepsat rovnici (3.19) na

$$q = sMWQ \tag{3.21}$$

kde

$$M = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.22}$$

Pokud předpokládáme, že body v prostoru leží všechny v jedné rovině, můžeme souřadnici z_Q položit rovnu 0. Když si navíc rotační matici R představíme jako tři sloupcové vektory $[r_1\ r_2\ r_3]$, zjistíme, že třetí vektor můžeme vynechat, protože se bude vždy násobit se souřadnicí z_Q, která je nulová. Rozepsaná rovnice přepočtu bodu Q na bod q tedy vypadá následovně.

$$\begin{bmatrix} x_q \\ y_q \\ 1 \end{bmatrix} = sM[\ r_1 \quad r_2 \quad r_3 \quad t\ ] \begin{bmatrix} x_Q \\ y_Q \\ 0 \\ 1 \end{bmatrix} = sM \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} x_Q \\ y_Q \\ 1 \end{bmatrix} \tag{3.23}$$

Výsledkem úpravy je, že matice H má nyní velikost jen 3×3. Z této dvojice rovnic (každý bod má dvě souřadnice, což jsou dvě rovnice), tedy chceme spočítat deset neznámých parametrů. Z toho se šest neznámých, popisujících polohu, mění s každým novým snímkem, protože každý snímek má jinou polohu a natočení. Vnitřní parametry kamery jsou konstantní. Pro výpočet těchto deseti parametrů potřebujeme tedy teoreticky pouze pět bodů, což je deset rovnic. Protože používáme rovinnou homografii, musíme pro výpočet použít dva snímky s minimálně čtyřmi body [11]. Pro výpočet parametrů zkreslení nám stačí jen tři body, což je šest rovnic. V praxi je vzhledem k šumu potřeba bodů mnohem více, aby výpočet byl co nejrobustnější.

Po provedení kalibrací jednotlivých kamer můžeme provést i kalibraci stereokamery. Kalibrací rozumíme výpočet vzájemné polohy obou kamer. Výsledkem tohoto procesu, tzv.

rektifikace, jsou takové obrazy kamer, které mají vzájemně zarovnané řádky, tzn. bod v obrazu jedné kamery má stejnou y-souřadnici jako ten samý bod v obrazu druhé kamery. Vzájemnou polohu kamer zjistíme výpočtem pomocí *epipolární geometrie*, detailněji viz např. [82]. Bod P v prostoru se promítne do bodů p_l a p_r v obrazech levé a pravé kamery. Tyto body leží na tzv. *epiliniích*. Bod P může ležet kdekoliv na přímce spojující tento bod se středem kamery, v druhém obrazu se tato přímka promítne do úsečky, nazývané epilinie. Pokud najdeme transformační matici (rotaci a translaci), kterou bod p_l v rovině levé kamery přepočítáme do bodu p_r v rovině pravé kamery, zjistíme vzájemnou polohu těchto dvou kamer. Tato vzájemná poloha je udána tzv. *esenciální* a *fundamentální* maticí. Esenciální matice E popisuje rotaci a translaci mezi dvěma kamerami, fundamentální matice F k tomuto popisu ještě přidává informace o intrinsických parametrech.

Subpixelová přesnost

Zpracování obrazu patří mezi výpočetně velice náročné operace. Proto se většinou nepoužívá příliš velké rozlišení kamer, i když kamery samotné to umožňují a navíc by i polohy bodů ve 3D byly vypočítány přesněji. Místo toho se více používá výpočtu bodů se subpixelovou přesností. Tímto způsobem je možno vypočítat polohu význačného bodu s přesností např. až na setiny pixelu. Při kalibraci stereokamery a i dále při výpočtech polohy detekovaných bodů je tento způsob také použit.

3.4 Detekce význačných bodů a hledání korespondencí

Na vyhledávání význačných bodů se používají dva typy algoritmů, *rohové detektory* a *příznakové deskriptory*.

Rohové detektory už podle názvu detekují v obraze především rohy. Mezi nejznámější a nejpoužívanější patří například Harrisův detektor [82]. Používají se často v případě, kdy chceme sledovat body v sérii navazujících snímků.

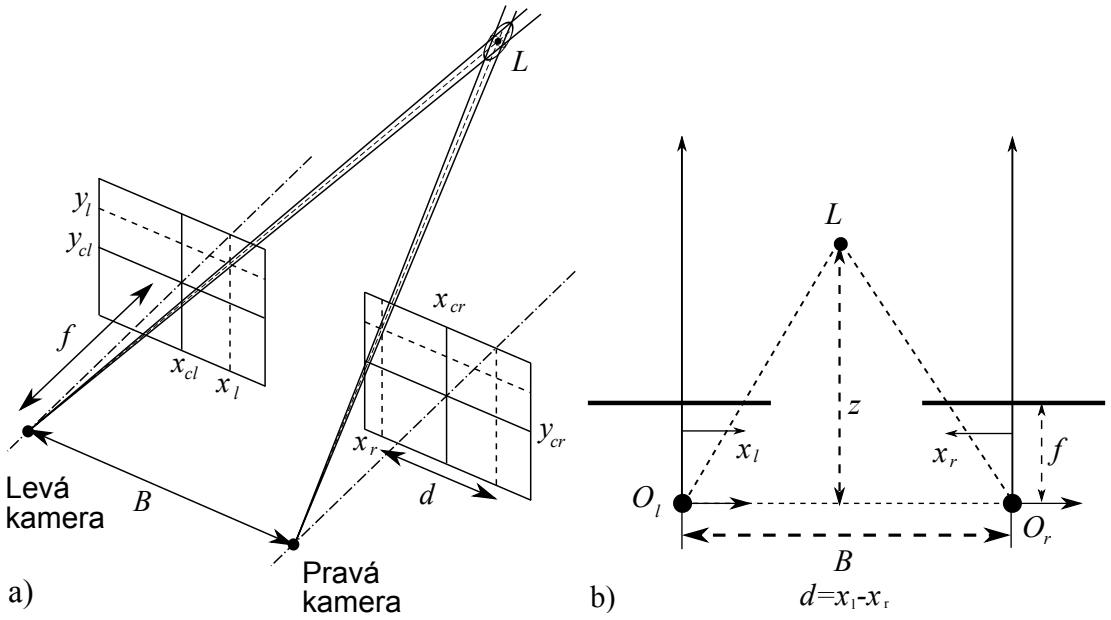
Deskriptory na druhou stranu detekují body, které jsou něčím významné. Mohou to být rohy, středy nějakých oblastí, atd. Rozdíl oproti detektorům je především v tom, že popíší nalezený bod *vektorem příznaků*. Mezi nejznámější patří SIFT [43] a SURF [6] detektory. Oba popíší nalezené body vektorem 64 nebo 128 hodnot. Výhoda těchto deskriptorů je v tom, že jsou odolné proti natočení nebo změně měřítka obrazu. Proto se používají v algoritmech SLAM kde mohou detekovat situaci, kdy se robot vrátí zpět na místo, které už jednou prozkoumal. Fungují tak v tomto případě jako detektory uzavření smyčky.

Vzájemné korespondence se určí pomocí vzdáleností bodů v 64 nebo 128 rozměrném prostoru. Navíc, protože detekované body mají nějaké 3D souřadnice, mohou se k hodnotám deskriptoru přidat i tyto hodnoty 3D pozic a detekce korespondencí tak bude robustnější. V případě stereokamer se mohou vzájemné korespondence v levém a pravém snímku také rozpoznávat pomocí těchto deskriptorů. Ukázka nalezených korespondencí, včetně těch chybných, je na obr. 4.16.

Zřejmou nevýhodou tohoto přístupu je však delší doba nutná pro vyhledání nejbližšího bodu v mapě, ve které mohou být řádově stovky nebo spíše i tisíce bodů.

3.5 Výpočet 3D polohy

3D poloha bodu v prostoru se určí pomocí triangulace z podobných trojúhelníků,



Obrázek 3.2: Model stereokamery a) Souřadnice bodu L vypočítáme z polohy bodu v obraze levé kamery a z disparity d. Elipsa kolem bodu L vyjadřuje nejistotu polohy tohoto bodu. b) Schématické znázornění podobných trojúhelníků ze kterých se vypočítá poloha bodu L.

$$z = \frac{fB}{d} \tag{3.24}$$

$$x = \frac{x_l z}{f} \tag{3.25}$$

$$y = \frac{y_l z}{f} \tag{3.26}$$

kde z je vzdálenost roviny bodu od roviny kamery, f je ohnisková vzdálenost získaná při kalibraci kamery, B je vzájemná vzdálenost kamer a d je disparita.

$$d = (x_l - x_{cl}) - (x_r - x_{cr}) \tag{3.27}$$

x_l je x-ová souřadnice bodu v levém obrázku, x_r je x-ová souřadnice bodu v pravém obrázku a x_{cl}, x_{cr} jsou x-ové středy obrázků. Tzn. v ideálním případě u obrázku o velikosti 640×480 je $x_{cl} = x_{cr} = 320$. y_l je y-ová souřadnice bodu v levém obrázku. Při výpočtu hodnoty Z se může stát, že tato hodnota vyjde záporná. To je samozřejmě nesmysl, protože to by znamenalo, že bod je za kamerou. Je to způsobeno špatným výsledkem hledání korespondence mezi odpovídajícími si body a takové body je třeba z dalšího zpracování vyloučit.

U detekovaného bodu můžeme z jeho polohy určit podle [51] i jeho kovarianční matici,

která odpovídá šumu měření:

$$R = \left(\frac{B}{d}\right)^2 \begin{pmatrix} \sigma_x^2 + \frac{\sigma_d^2(x_l - x_{cl})^2}{d^2} & \frac{\sigma_d^2(x_l - x_{cl})(y_l - y_{cl})}{d^2} & \frac{\sigma_d^2(y_l - y_{cl})f}{d^2} \\ \frac{\sigma_d^2(x_l - x_{cl})(y_l - y_{cl})}{d^2} & \sigma_y^2 + \frac{\sigma_d(y_l - y_{cl})^2}{d^2} & \frac{\sigma_d^2(y_l - y_{cl})f}{d^2} \\ \frac{\sigma_d^2(x_l - x_{cl})f}{d^2} & \frac{\sigma_d^2(x_l - x_{cl})f}{d^2} & \frac{\sigma_d^2 f^2}{d^2} \end{pmatrix} \tag{3.28}$$

Hodnoty σ_x, σ_y a σ_d se zjistí empiricky.

Jedním ze stěžejních úkolů algoritmu SLAM je vypočítat co nejpřesněji polohu bodů ve 3D prostoru. To můžeme udělat buď jednoduše pomocí váhovaného průměru anebo složitěji, ale lépe pomocí Kalmanova filtru [51], [94]. V tom případě bude stavem systému poloha bodu ve 3D prostoru a výsledkem měření budou opět souřadnice ve 3D prostoru. Protože stav systému je v podstatě souřadnice bodu ve 3D prostoru (její střední hodnota), budeme v Kalmanově filtru psát rovnou μ_t^- místo x_t^- a Σ_t^- místo P_t^-. Predikce v Kalmanově filtru tedy bude mít tvar

$$\mu_t^- = F_t\mu_{t-1} + B_t u_t \tag{3.29}$$
$$\Sigma_t^- = F_t\Sigma_{t-1}F_t^T + Q_t \tag{3.30}$$

a korekce bude mít tvar

$$K_t = \Sigma_t^- H_t^T (H_t\Sigma_t^- H_t^T + R_t)^{-1} \tag{3.31}$$
$$\mu_t = \mu_t^- + K_t(z_t - H_t\mu_t^-) \tag{3.32}$$
$$\Sigma_t = (\mathbf{I} - K_t H_t)\Sigma_t^- \tag{3.33}$$

Protože 3D poloha bodů je statická, tzn. v čase se nemění a také na ně nepůsobí žádné řízení, můžeme položit $Q_t = 0$, $F_t = \mathbf{I}$ a $u_t = 0$. Navíc, protože rozměr měření je stejný jako rozměr stavu systému, bude i matice $H = \mathbf{I}$. Rovnice predikce se tak zjednodušší na

$$\mu_t^- = \mu_{t-1} \tag{3.34}$$
$$\Sigma_t^- = \Sigma_{t-1} \tag{3.35}$$

a korekci přepíšeme do tvaru

$$\mu_t = \mu_{t-1} + \Sigma_{t-1}(\Sigma_{t-1} + R_t)^{-1}(z_t - \mu_{t-1}) \tag{3.36}$$
$$\Sigma_t = (\mathbf{I} - \Sigma_{t-1}(\Sigma_{t-1} + R_t)^{-1})\Sigma_{t-1} \tag{3.37}$$

Rovnici (3.36) roznásobíme a abychom se u (3.37) zbavili jednotkové matice, položíme $\mathbf{I} = (\Sigma_{t-1} + R_t)/(\Sigma_{t-1} + R_t)$ a odečteme.

$$\mu_t = \mu_{t-1} + \Sigma_{t-1}(\Sigma_{t-1} + R_t)^{-1}z_t - \Sigma_{t-1}(\Sigma_{t-1} + R_t)^{-1}\mu_{t-1} \tag{3.38}$$
$$\Sigma_t = R_t(\Sigma_{t-1} + R_t)^{-1}\Sigma_{t-1} \tag{3.39}$$

Když v rovnici (3.38) vytkneme μ_{t-1}, dostaneme opět jednotkovou matici $\mathbf{I}$, kterou nahradíme podobně jako v předchozím případě $\mathbf{I} = (\Sigma_{t-1} + R_t)/(\Sigma_{t-1} + R_t)$ a opět odečteme. A rovnici (3.39) můžeme ještě přepsat. Takže obě rovnice teď budou mít tvar:

$$\mu_t = \Sigma_{t-1}(\Sigma_{t-1} + R_t)^{-1}z_t + R_t(\Sigma_{t-1} + R_t)^{-1}\mu_{t-1} \tag{3.40}$$
$$\Sigma_t = (\Sigma_{t-1}^{-1} + R_t^{-1})^{-1} \tag{3.41}$$

Když do rovnice (3.40) vhodně dosadíme rovnici (3.39) dostaneme

$$\mu_t = \Sigma_t R_t^{-1} z_t + \Sigma_t \Sigma_{t-1}^{-1} \mu_{t-1} \tag{3.42}$$

Pokud ještě z (3.42) vytkneme Σ_t dostaneme konečnou podobu obou rovnic.

$$\mu_t = \Sigma_t(\Sigma_{t-1}^{-1}\mu_{t-1} + R_t^{-1} z_t) \tag{3.43}$$

$$\Sigma_t = (\Sigma_{t-1}^{-1} + R_t^{-1})^{-1} \tag{3.44}$$

Těmito dvěma rovnicemi, tedy novým měřením z_t, aktualizujeme polohu a kovarianci bodu v čase t. R_t je šum měření v čase t.

3.6 Vizuální odometrie

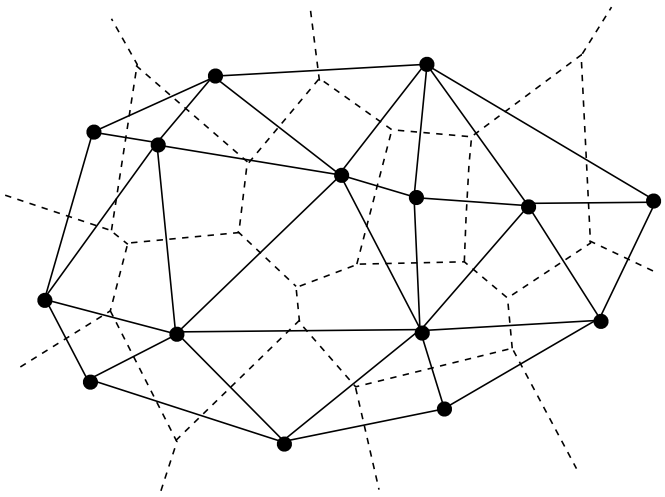
Vizuální odometrie je název pro model pohybu robota, který nepoužívá žádné senzory pro zjištění posunutí, ale spoléhá se pouze na kameru. Pomocí jí detekovaných bodů spočítá posunutí kamery. Tato technika je použita v řadě článků, např. v [56] a dalších.

Vizuální odometrie v podstatě spočívá v tom, že robot nalezne ty samé význačné body ve dvou následujících snímcích a z jejich známé polohy ve 3D a z posunutí bodů v obraze spočítá posunutí kamer. Snímky po sobě nemusí bezprostředně následovat, je výhodnější, pokud robot mezi nimi vykoná nějaký větší pohyb. Přesnost této metody závisí na tom, jak přesně jsme schopni určit polohu 3D bodů v prostoru, resp. polohu těchto bodů v obraze. Důležité je, aby nalezených dvojic bodů bylo co nejvíce, a eliminovaly se tak nepřesnosti způsobené špatným určením polohy bodu v obraze.

Jedním ze způsobů, jak vypočítat posunutí kamer je pomocí fundamentální matice, stejně jako při kalibraci stereokamery. Jiným způsobem je využít stereokamery, vypočítat 3D polohy bodů v jednotlivých obrazech a potom tyto *mračna bodů* (ang. point clouds) co nejpřesněji spárovat pomocí metody ICP (Iterative Closest Point). Tím dostaneme matice rotace a translace a určíme tak vzájemnou polohu kamer. Tato metoda je použita např. v knihovně MRPT [2]. Vychází z článku od B. Horna [33], který ve své práci použil i Moreno a kol. [51]. Spočívá v tom, že se vypočítají středy obou mračen bodů, to nám určí požadovanou translaci a pomocí vlastních vektorů se vypočítá i rotace. Problémem je, že v případě chybných korespondencí nebo jen nepřesných poloh některých bodů dochází k postupné ztrátě přesnosti polohy. Ta se dá částečně eliminovat tím, že jednotlivým bodům přiřadíme váhy, které nám budou říkat, jak moc má daný bod přispívat k vypočítané poloze. Tyto váhy budou vypočítány z kovarianční matice bodu.

3.7 Delaunayho triangulace

Zde budeme triangulací rozumět pospojování bodů ve 2D prostoru do trojúhelníkové sítě. V případě 3D prostoru se buďto z trojúhelníků stanou čtyřstěny a vyplní tak celý prostor mezi body nebo se vytvoří opět trojúhelníková síť. V tomto případě se takováto sít nazývá *povrch*. Dobře jsou základní principy triangulace vysvětleny v knize [7] na příkladě tvorby povrchu země. Jednotlivé body zde tvoří místa, kde jsme si jisti nadmořskou výškou, protože jsme ji v tomto místě změřili. Pospojováním jednotlivých změřených bodů do trojúhelníkové sítě se snažíme dosáhnout přibližné aproximace povrchu. Přestože o skutečném tvaru povrchu nemáme žádné informace, snažíme se, aby mu naše aproximace odpovídala co nejlépe.



Obrázek 3.3: Delaunayho triangulace (plnou čarou). Čárkovaně je zobrazena tzv. Voronoiova tesselace [58].

Představme si, že máme čtyřúhelník a chceme spojit body úhlopříčkou. Máme dvě možnosti jak to udělat a přitom chceme, aby tato úhlopříčka co nejlépe odpovídala skutečnému tvaru prostoru. Intuitivně nejsprávnější řešení je spojit spolu ty dva body, které jsou k sobě blíže. Tím v podstatě aproximujeme neznámý prostor na základě známých, co nejbližších, údajů[1]. Nejznámější metodou, která vytváří triangulaci s námi požadovanou vlastností je *Delaunayho triangulace* (DT).

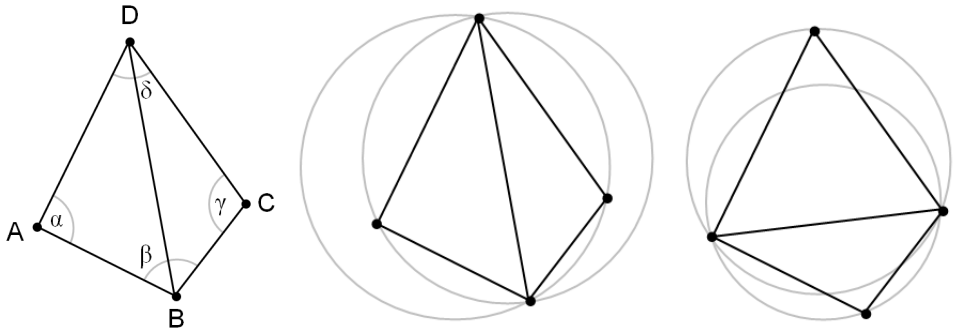
3.7.1 Obyčejná Delaunayho triangulace

Delaunayho triangulace [30] je taková triangulace, kde pro vsechny tři vrcholy tvořící trojuhelník platí, že v kružnici jimi opsané neleží žádný další vrchol. Tato vlastnost vede k některým dalším vlastnostem. Nejdůležitější z nich je ta, že Delaunayho triangulace maximalizuje minimální úhel v trojúhelníku. Tyto vlastnosti znamenají, že se DT snaží vytvářet trojúhelníky svým tvarem co nejvíce podobné rovnostrannému trojúhelníku a minimalizovat tak počet trojúhelníků s příliš ostrými úhly.

Způsobů, jak vytvořit DT, je celá řada. Jedním z nejjednodušších způsobů je prohazování hran (viz obr. 3.4). Jestliže součet dvou úhlů ležících naproti hraně společné pro dva trojúhelníky je větší než 180°, tyto dva trojúhelníky nesplňují Delaunayho podmínku. Pokud tuto hranu prohodíme s hranou spojující druhé dva body, budou nově vytvořené trojúhelníky Delaunayho podmínku splňovat.

Pravděpodobně nejčastějším způsobem, jak vytvořit DT, je však inkrementální metoda. Vrcholy do už vytvořené triangulace přidáváme postupně, spojíme je s těmi vrcholy v trojúhelníku, do kterého nový vrchol padl a pak, např. metodou prohazování hran, obnovíme DT. Abychom zajistili, že nový vrchol padne vždy do nějakého trojúhelníku, vytvoříme si na začátku trojúhelník, jenž bude obsahovat celou oblast s vrcholy, které budeme chtít triangulovat. Uvedený přístup se používá např. v knihovně OpenCV [95].

[1] To ovšem neznamená, že tato vybraná hrana odpovídá správnému tvaru povrchu. Pouze to znamená, že naše volba byla správná s ohledem na to, že o tvaru povrchu nic bližšího nevíme.



Obrázek 3.4: Ukázka prohazování hran. Součet úhlů α a γ je větší než 180°, proto v kružnicích opsaných oběma trojúhelníkům leží vždy ještě další bod. Pokud ale hranu $B - D$ prohodíme za hranu $A - C$ bude už Delaunayho podmínka splněna. (Obrázky převzaty z http://en.wikipedia.org/wiki/Delaunay_triangulation)

Operací, kterou je třeba v Delaunayho triangulaci často provádět, je výpočet, zda daný vrchol P leží v kružnici opsané bodům A, B, C. Toto se dá snadno určit výpočtem determinantu (3.45). Pokud jsou body A, B, C uspořádány proti směru hodinových ručiček, je determinant kladný v případě, že bod P leží uvnitř opsané kružnice.

$$\begin{vmatrix} A_x & A_y & A_x^2 + A_y^2 & 1 \\ B_x & B_y & B_x^2 + B_y^2 & 1 \\ C_x & C_y & C_x^2 + C_y^2 & 1 \\ P_x & P_y & P_x^2 + P_y^2 & 1 \end{vmatrix} > 0 \tag{3.45}$$

3.7.2 Váhovaná triangulace

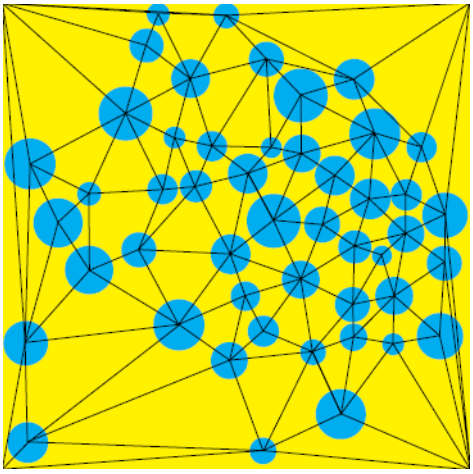
Dalším typem triangulace je váhovaná triangulace [1]. V tomto případě každému bodu P přísluší i váha w, která představuje kružnici K kolem tohoto bodu s poloměrem $r = \sqrt{w}$. Příklad takovéto triangulace můžeme vidět na obrázku 3.5. Kružnice opsaná trojúhelníku ve validní váhované triangulaci pak nesmí protnout žádnou kružnici K. Na obyčejnou DT se tak můžeme dívat jako na váhovanou triangulaci, kde všechny váhy jsou nulové. Analogicky tak do determinantu v rovnici (3.45) přidáme váhy w a spočítáme, zda bod P leží mimo opsanou kružnici [27].

$$\begin{vmatrix} A_x & A_y & w_A - (A_x^2 + A_y^2) & 1 \\ B_x & B_y & w_B - (B_x^2 + B_y^2) & 1 \\ C_x & C_y & w_c - (C_x^2 + C_y^2) & 1 \\ P_x & P_y & w_P - (P_x^2 + P_y^2) & 1 \end{vmatrix} > 0 \tag{3.46}$$

3.7.3 Constrained triangulace

Posledním, zde popsaným typem triangulace, je Constrained Triangulation[2] [30]. V tomto typu triangulace jsou určité hrany předem dány a nemohou být triangulací změněny. Potom

[2]Český překlad by asi zněl vynucená triangulace



Obrázek 3.5: Příklad váhované triangulace. Modrá kolečka představují váhy jednotlivých vrcholů. Obrázek převzat z [27].

však není zajištěna Delaunayho podmínka bodu uvnitř opsané kružnice. Uvedený typ triangulace se používá například při použití metody konečných prvků. V tomto případě se také často používá úpravy pro tzv. *high-quality mesh*, kdy mohou být do triangulace přidávány další body tak, aby se dosáhlo splnění Delaunayho podmínky a vytvořené trojúhleníky byly co nejblíže rovnostranným trojúhleníkům [13].

3.8 Tvorba 3D modelu

Tvorba 3D modelů objektů je popsána ve velkém množství článků. Ve velké míře jsou jejich autory však spíše členové komunity počítačového vidění a zpracování obrazu než lidé, věnující se spíše robotice. Obě dvě skupiny mají také k problému trochu jiný přístup. Cílem tvorby modelu v počítačovém vidění je spíše vytvořit vizualizaci objektu, než snaha o co nejpřesnější polohu jednotlivých bodů na modelu, jako je tomu v případě robotiky. To však neznamená, že by modely vytvořené v rámci počítačového vidění byly nepřesné. Buď k žádným nepřesnostem nedochází, neboť modelovaný objekt je malý a je snímán kamerami umístěnými okolo tohoto objektu, např. v [36], nebo jde více o realističnost modelu, než o jeho přesný tvar, jako např. v [61]. Další práce z oblasti počítačové grafiky, které popisují tvorbu rozsáhlých modelů, jsou např. [67], [68], [15]. V článku [62] je popsáno zmapování většího území pomocí kamery, ovšem ta je pro přesnější lokalizaci doplněna systémem GPS a inerciální jednotkou.

Všechny uvedené práce používaly pro tvorbu modelu obrazy z kamer. Jiný přístup pro tvorbu modelu je použít laserový scanner a množinu bodů tak místo ze stereokamery získat pomocí laseru. Známé práce v této oblasti jsou ty, týkající se Projektu Michalangelo, kdy byla pomocí velice přesného laserového scanneru nasnímána řada Michelangelových soch. Tvorba modelu z takových dat je popsána například v článku [16] Podobné přístupy mapování pomocí laseru se objevují i v robotice, např. [47].

Problém, který však mají oba tyto způsoby společný je, jak z množiny získaných bodů vytvořit povrch objektu. Jedním z nejznámějších algoritmů na tvorbu povrchu je PowerCrust Niny Amenty [3], [4] a další je popsán např. v [32]. Nevýhodou těchto algoritmů je výpočetní náročnost. Pro robotiku by bylo vhodné, aby tyto algoritmy pracovaly ideálně

v reálném čase. Dalším problémem, proč nemůžeme tyto algoritmy použít je to, že body, se kterými tyto metody pracují, jsou blízko sebe a tak se dá odhadnout tvar povrchu. To ovšem v oblasti robotiky nebývá splněno. Nemůžeme předem nijak zaručit, že se robot bude pohybovat v oblasti, kde budou objekty s výraznou texturou, na kterých bude schopen detekovat velké množství bodů. Proto tyto přístupy nejsou pro nás vhodné. Navíc je nežádoucí, aby mapu tvořilo tak velké množství bodů, protože by to znamenalo obtíže při kopírování, ukládání a vůbec při veškeré manipulaci s mapou.

Otázkou je, zda je vůbec možné vytvořit model z řídce rozmístěných bodů. Pokud bychom měli pouze samotné body, tak to možné není. Nevěděli bychom, kde má být povrch a tak bychom triangulací mohli udělat plochy i tam, kde být nemají. Naší výhodou však je, a to je další rozdíl oproti předešlým algoritmům, že body získáváme a zpracováváme postupně tak, jak je detekuje robot. Víme tedy, odkud byl který bod viditelný.

Jednou z mála prací, zabývajících se tvorbou mapy z řídce rozmístěných bodů jsou články A. Hiltona [45], [31].

Postup tvorby modelu podle A. Hiltona je tedy následující:

1. Globální model M, omezující podmínky viditelnosti C a význačné body F, se inicializují jako prázdné množiny.

2. Vytvoří se konzistentní model M_i pro i-tý pohled jako triangulace význačných bodů F_i v rovině kolmé na směr pohledu.

3. Zruší se trojúhelníky v globálním modelu M, které odporují podmínkám viditelnosti C_i i-tého pohledu. Výsledný globální model M' je nyní konzistentní s omezující podmínkou $C' = \{C \bigcup C_i\}$.

4. Zruší se trojúhelníky v M_i, které odporují podmínkám viditelnosti C, výsledkem je M_i' které je konzistentní s C'.

5. Trojúhelníky M_i' a neredundantní trojúhelníky z M' se sloučí do $M'' = \{M' \bigcup M_i'\}$.

6. Aktualizuje se globální pohled $M = M''$ a omezující podmínky $C = C'$.

7. Kroky 2-6 se opakují pro všechny snímky N.

Model M se tak bude skládat ze dvou druhů trojúhelníků, reálných a virtuálních. Virtuální trojúhelníky se budou postupně rušit a pro počet snímků N, jdoucích k nekonečnu, se bude model blížit skutečnému povrchu. V praxi je možné, z důvodů náročnosti uvedeného algoritmu, testovat pouze podmínky viditelnosti z nového snímku C_i na globálním modelu M. Aby se vyloučilo opakované přidávání už odstraněných trojúhelníků do nového modelu, přidávají se pouze ty trojúhelníky, jejichž alespoň jeden vrchol je tvořen nově objevenými body ve snímku i a alespoň jeden vrchol tvoří body nalezené v dřívějších snímcích. Trojúhelníky, jejichž všechny vrcholy tvoří už dříve nalezené body, se přidávat nebudou.

Kapitola 4

Mapování 3D prostředí

V předchozích kapitolách byla popsána teorie nutná pro vytváření map pomocí autonomních mobilních robotů, včetně částí nezbytných pro práci a zpracování obrazu ze stereokamery. V následující kapitole popíšeme rozšíření potřebné pro autonomní tvorbu 3D mapy robotem pomocí sterokamery. Jednat se bude především o navržený způsob plánování objevování dosud neprozkoumaného prostoru a o úkoly s tím spojené.

4.1 Reprezentace 3D prostředí

Jedním z úkolů práce je navrhnout reprezentaci mapy pro 3D modely. Vzhledem k tomu, že se předpokládá použití stereokamer a i algoritmy (např. SURF), popsané v této práci, jsou založeny na detekci význačných bodů, je přirozené, že i navržená mapa se bude skládat z význačných bodů. To znamená, že se v obraze nebudou vyhledávat plochy ani hrany. Protože chceme, aby mapu, nebo přesněji řečeno model, mohl používat i člověk, musí výsledná mapa vypadat jako vizualizace z různých CAD programů a počítačových her. Potom bude moci být kromě použití robotem využita i na vizualizaci například památek, obtížně přístupných míst atd. Rozdílem oproti těmto běžným 3D modelům bude to, že mapa vytvářená robotem musí počítat i s určitou nepřesností, způsobenou nedokonalostí použitých senzorů. To znamená, že mapa, alespoň ve fázi vytváření, bude v sobě obsahovat kromě souřadnic bodů i další nezbytné položky.

Každý význačný bod má o sobě uloženy následující informace:

- index bodu
- 3D souřadnice polohy
- kovarianční matice polohy
- 128-rozměrný vektor deskriptoru
- kovarianční matice (nebo vektor) deskriptoru
- souřadnice bodu v obrazu levé kamery
- počet zhlédnutí bodu
- typ bodu

- odkaz na předcházející a následující bod na volné ploše
- souřadnice místa, odkud byl bod detekován

Účel prvních položek je zřejmý. Počet zhlédnutí bodu je vhodné uchovávat z toho důvodu, aby body s malým počtem zhlédnutí mohly být z modelu vymazány, protože se s největší pravděpodobností bude jednat o chybně určené korespondence a tudíž o chybné body. Položka typ bodu bude udávat, o jaký bod se jedná. Např. u bodů, tvořících průjezdnou oblast bude udávat, zda se jedná o hraniční body, o vnitřní body atd. Použití položky typ bodu a význam předcházejících a následujících bodů bude více vysvětlen v kapitole 4.2.

Kromě uvedených informací je potřeba použít strukturu, v níž bude uložena triangulace a následně i mesh. Pro tyto účely se často používá tzv. doubly-connected edge list. Detailní popis této struktury lze nalézt například v [7]. Nalezené body budou pomocí triangulace spojeny do struktury označované jako *mesh* a doplněny o texturu získanou z jednotlivých snímků z kamer.

V případě, že si budeme chtít prohléhnout kompletní nebo prozatím vytvořenou mapu, můžeme ji převést například do formátu .obj. Jedná se o textový soubor vytvořený firmou Wavefront. Jeho výhodou je to, že jde o široce rozšířený a podporovaný formát, ve kterém jsou modely uloženy jako posloupnost vrcholů a trojúhelníků. Dá se v něm také pomocí materiálového souboru .mtl uložit textura pro daný model. Struktura souboru .obj je ukázána v příloze.

4.2 Průzkum prostředí pro 3D mapování

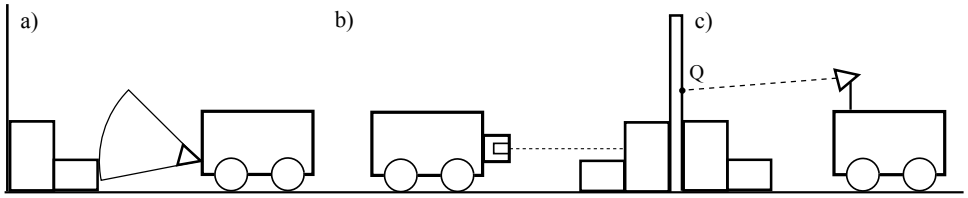
Jestliže chceme, aby robot vytvářel mapu autonomně, bez zásahu uživatele, je nutné jej vybavit plánovacím algoritmem. Ten mu bude říkat, do které části prostoru se vydat tak, aby mapa byla co nejpřesnější a co nejrychleji vytvořená. Námi navržený plánovací algoritmus se bude skládat ze dvou částí.

- Výběr cílů - výběr míst, ke kterým robot pojede.
- Plánování cesty - samotné plánování cesty robota v mapě.

Plánování cesty do velké míry závisí na způsobu reprezentace mapy. V případě, že je mapa tvořena body v prostoru a hranami, které tyto body spojují, musíme zvolit jiné plánování cesty než v případě, kdy je mapa reprezentována např. mřížkou obsazenosti.

Vzhledem ke zvolené reprezentaci mapy bude nejvhodnější reprezentovat cestu stejně, tzn. jako posloupnost bodů spojených hranami. Nebude tak potřeba vytvářet další struktury, ve kterých by se teprve plánování cesty provádělo. Například mít zvlášť vytvořenou mřížku obsazenosti, ve které by se hledala cesta. Plánování cesty tak bude mít podobu hledání cesty v grafu, na což můžeme použít známé algoritmy, jako je např. Dijkstrův nebo A*.

Body určené pro plánování cesty robota můžeme vytvořit několika způsoby. Buď si je do mapy vygenerujeme náhodně, tak jako v případě pravděpodobnostního plánování cesty, nebo jimi můžeme pokrýt zvolenou oblast pravidelně. Možností, kterou jsme navrhli v této práci je však vytvořit cestu z bodů, které se budou generovat na místech, kde se už robot pohyboval. V případě použití těchto bodů máme totiž jistotu, že cesta po nich nalezená je průjezdná, protože víme, že robot po ní už jel a odpadá tak plánování a testování průjezdnosti.



Obrázek 4.1: Ukázka detekování překážky sonarem, laserem a kamerou. V případě sonaru můžeme předpokládat, že mezi robotem a detekovanou překážkou není jiná překážka. Stejně tak se to předpokládá i v případě měření laserem, i když zde tento předpoklad nemusí být zcela správný. Naproti tomu u kamery detekování překážky v žádném případě neznamená, že před touto překážkou nemůže být ještě jiná překážka.

4.2.1 Vyznačování průjezdné oblasti

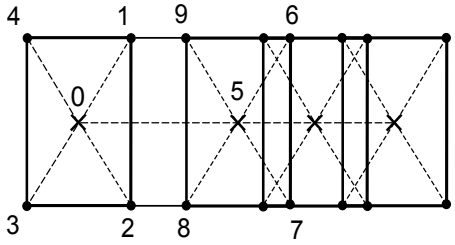
Pokud robot vytváří mapu ve 2D, používá k tomu většinou laserový scanner nebo sonary. Překážky, které těmito senzory detekuje, jsou pak zaneseny do mapy a volná plocha představuje pro robota oblast, která je průjezdná (Obr. 4.1). I když v případě použití laseru, který je umístěn na robotovi v určité výšce a vlastně detekuje překážky v této výšce, se může stát, že si robot označí jako volné místo i takové, které bude obsahovat překážku nižší než je výška, ve které je laser umístěn.

V případě použití kamer je situace odlišná, protože nemůžeme oblast mezi robotem a detekovanými body automaticky považovat za průjezdnou. Robot totiž může nalézt nějaký bod, např. na zdi, před níž je ještě skříň, kterou už robot nevidí, a tak si celou oblast chybně označí jako průjezdnou. Stejně tak se může stát, že robot detekuje bod nad průjezdným místem a pak si tuto část označí jako neprůjezdnou. Navíc by bylo z hlediska plánování vhodné, aby byla průjezdná oblast od neprůjezdné oblasti nějakým způsobem oddělena. Vzhledem k reprezentaci mapy by toto oddělení mohlo být provedeno speciálně označenými hranami, které budou vymezovat volný prostor. Není ani příliš vhodné použít k vyznačení překážek průmět nalezených bodů do podlahy a ve volném prostoru mezi nimi pak hledat cestu pro robota. Znamenalo by to vytvořit si ještě navíc k mapě další plochu, do které se bude zaznamenávat volná oblast a v ní pak hledat cesta.

Způsobem, jak si vyznačit oblast, která je zcela jistě průjezdná, a to navíc přímo do mapy, je vytvořit ji z jednotlivých poloh robota. Robot si během cesty bude do mapy dělat jakési "značky"(obr. 4.2) a jejich spojením pak dostaneme území, označované v této práci jako *průjezdná oblast* nebo také volný prostor. Protože se celá mapa skládá z bodů a z hran (trojúhelníků), bude se z nich skládat i tato část mapy. Mapa tedy bude vlastně ze dvou oddělených částí. Jedna část budou body na překážkách, detekované kamerami a druhá část budou body vyznačující podlahu, tedy už zmíněnou průjezdnou oblast. Body na podlaze a hlavně jejich spojnice pak mohou navíc sloužit i k plánování cesty robota.

Vzhledem k tomu, že se body v mapě budou spojovat navzájem pomocí triangulace, budou se tak spojovat i body vymezující volný prostor. Zvoleným způsobem triangulace bude tzv. Constrained Delaunay Triangulation (CDT), popsaná v kapitole 3.7.3. Hranici robotova pohybu a tudíž i hranici mezi oběma částmi mapy (volný, průjezdný prostor a překážky detekované kamerami) tak budou vymezovat "neodstranitelné"(ang. constrained) hrany. Body, tvořící tyto hrany se budou nazývat *hraniční vrcholy.*

Tyto *neodstranitelné hrany* se budou vytvářet postupně jako spojnice jednotlivých robotových poloh, tzv. *značek.* Značky se budou skládat z pěti bodů, čtyři budou v rozích robota a jeden ve středu robota (předpokládáme, že robot bude mít tvar obdélníku). Tyto



Obrázek 4.2: Na obrázku jsou znázorněny tři po sobě jdoucí pohyby robota. Při prvním pohybu se robot posunul o více než je jeho délka, při druhých dvou o méně než jeho délka. Zároveň je také ukázáno číslování vrcholů. Tyto body pak budou použity pro triangulaci průjezdné oblasti.

body a jejich spojnice se budou následně dělit do několika druhů. Informace o druhu pak budou uloženy u každého bodu v reprezentaci mapy, viz kapitola 4.1.

1. Body ležící uvnitř volné plochy - tyto body budou moci být použity pro plánování pohybu robota.
2. Body ležící na hranicích volné plochy - jejich spojnice budou tzv. neodstranitelné hrany. Tyto body mohou být také použity pro plánování cesty robota. V tomto případě ale bude muset být nalezená cesta otestována na případné kolize. Body se budou dělit na další tři typy.
   (a) Body, které budou mít spojnice jen s body na hranici nebo uvnitř volné plochy. Ty primárně nebudou pro robota nijak zajímavé.
   (b) Body, které budou mít spojnici s bodem nalezeným kamerou. Tyto body budou tvořit hranici prozkoumávaného území.
   (c) Body na hranici, jichž nepůjde použít pro plánování cesty, protože robot by se na nich dostal do kolize s překážkou. Cílem robotova mapování bude vytvořit ze všech hraničních bodů předchozího typu body tohoto typu. Při splnění této podmínky je zřejmé, že žádnou další oblast nelze prozkoumat a celá oblast je tedy zmapována.

Pravidla pro označování hran

Smyslem dále uvedených pravidel je zajistit, aby se neodstranitelné hrany tvořily jen na hranicích volného prostoru a nikoliv i uvnitř.

Pravidla pro označování hran musejí vzít v úvahu následující možné pohyby robota (obr. 4.3).

- Pohyb dopředu s otáčením.
- Otáčení se na místě.
- Pohyb po hranici volného prostoru.
- Vstup do volného prostoru.

- Opuštění volného prostoru.
- Pohyb uvnitř volného prostoru.

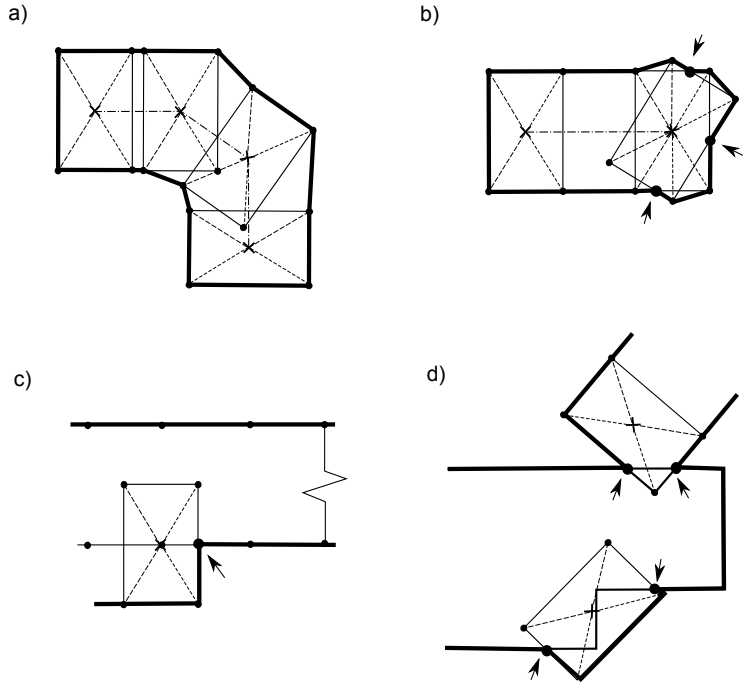
Hrany mezi rohovými body robotovy pozice se budou označovat jako *neodstranitelné*, hrany spojující středový bod jako *odstranitelné*. Přitom je důležité, že hrany, označené jako neodstranitelné, nemohou být při následující triangulaci odstraněny, zatímco hrany odstranitelné mohou. Pokud se robot pohne takovým směrem, že se dostane nad neodstranitelnou hranu, bude tato hrana přeznačena jako odstranitelná a nová poloha robota bude spojena se starou polohou neodstranitelnými hranami.

Navržená pravidla pro vytváření značek jsou následující:

- Vzdálenost mezi jednotlivými značkami nesmí být větší, než je šířka robota. Z důvodu plánování cesty po středových bodech je totiž žádoucí, aby se středy poloh robota spojovaly navzájem, a to i v případě následné CDT.
- Značka musí být vytvořena v takovém okamžiku, aby rotace robota byla menší, než úhel mezi jeho středem a dvěma rohy na straně robota.
- Do struktury s cestou se vždy musí přidat všech pět bodů, které budou tvořit značku a to v pořadí střed, vlevo vepředu, vpravo vepředu, vpravo vzadu, vlevo vzadu. To proto, abychom podle jejich ID poznali, zda je bod středový, případně o který z krajních bodů se jedná. Tato informace bude použita pouze ke správnému připojení k předchozí značce.
- Nově přidaný bod se bude připojovat k bodům přidaným předtím, nejprve k bodům předchozí pozice, pak k bodům současné pozice.
- Jestliže se neodstranitelné hrany nové pozice protínají se současnou hranicí volného prostoru, jsou vypočítány souřadnice jejich průsečíků a vnitřek staré hranice je označen jako odstranitelný.
- Jestliže hrana, která by měla být označena jako neodstranitelná, je celá uvnitř volného prostoru, je označena jako odstranitelná.

Podle uvedených požadavků je tedy posloupnost a připojování bodů následující:

1. Středový bod se přidává jako první, proto nemůže být připojen k žádným bodům současné pozice.
2. Levý přední bod se nepřipojuje k žádnému dalšímu bodu.
3. Pravý přední bod se připojí k levému přednímu bodu.
4. Pravý zadní bod se připojí k pravému přednímu, a v případě, že robot zatáčí doleva nebo jede přímo i k pravému přednímu bodu předchozí pozice. Jestliže robot zatáčí doprava, připojí se tento bod k pravému zadnímu bodu předchozí pozice.
5. Levý zadní bod se připojí k levému přednímu a v případě, že robot zatáčí doprava nebo jede přímo i k levému přednímu bodu předchozí pozice. Jestliže robot zatáčí doleva, připojí se tento bod k levému zadnímu bodu předchozí pozice.
6. Přední hrana předchozí pozice je označena jako smazatelná.



Obrázek 4.3: Příklad volné oblasti vytvořené a) robotem pohybujícím se vpřed a zatáčejícím, b) robotem otáčejícím se na místě, c) robotem jedoucím po hranici, d) robotem vjíždějícím nebo vyjíždějícím do volného prostoru. Šipky na obrázcích zároveň ukazují na jeden, dva nebo tři průsečíky s hranicí volného prosotru.

7. Jestliže se robot otáčí vpravo, pravá hrana je označena jako smazatelná.
8. Jestliže se robot otáčí vlevo, levá hrana je označena jako smazatelná.
9. Pokud se robot pohybuje uvnitř volného prostoru, přidávají se do mapy pouze body středové pozice.

Předcházející pravidla platí pro robota jedoucího vpřed, pokud by robot couval, jsou pravidla náležitě přizpůsobena. Po přidání nové neodstranitelné hrany ji musíme otestovat, zda se neprotíná s jinou neodstranitelnou hranou (hranicí volného prostoru). V tomto případě může nastat několik různých situací (obr. 4.3):

- Žádný průsečík - robot se celý nachází v nové oblasti nebo ve volné oblasti. Jestliže body na rozích předchozí pozice byly hraniční body, robot je v nové oblasti. Jestliže body na rozích nebyly hraniční, robot je ve volné oblasti.
- Jeden průsečík - robot se pohybuje po hranici. Nová hranice bude vytvořena mezi průsečíkem robotovy přední hrany a těmi rohovými body, jež byly označeny v předchozí poloze jako hraniční.
- Dva průsečíky (obecně sudý počet průsečíků) - robot vjel do volné oblasti nebo z ní vyjel. Robot vjel do volné oblasti, jestliže body na jeho předních rozích v předchozí poloze byly označeny jako hraniční. V tomto případě je smazána stará hranice mezi těmito dvěma průsečíky. Pokud robot vyjel z volné oblasti, je opět smazána stará hranice mezi dvěma průsečíky a novou hranici teď tvoří druhá spojnice dvou průsečíků.

- Tři průsečíky - robot se otočil na místě. V tomto případě připojíme levý přední bod původní pozice s novým předním levým a analogicky pravý přední bod původní pozice s novým pravým předním. Ten zadní bod, který není ve volné oblasti připojíme k příslušnému bodu předchozí pozice. Spočítáme společné průsečíky levých, předních a pravých hran a připojíme je k rohovým bodům.

Po každé aktualizaci hranice přidáme nové body a hrany do CDT. Abychom nemuseli dělat novou strukturu pro ukládání těchto hran, bude mít každý bod tvořící volnou plochu indexy na svoje předchůdce a následníky. Pokud bod vznikne jako průsečík hran a nikoliv z polohy robota, indexovat se nebude.

Triangulaci uvnitř volné plochy budeme tvořit jen ze středových bodů robota. To proto, aby uvnitř volného prostoru nebylo velké množství bodů a nalezené cesty tak byly hladší.

4.2.2 Plánování cesty

Plánování cesty se vzhledem ke zvolené reprezentaci mapy redukuje na vyhledávání cesty v grafu. Pokud budeme předpokládat, že cílový bod leží mimo volnou plochu, nalezneme k tomuto bodu nejbližší hraniční bod a k tomu naplánujeme cestu z robotovy současné pozice.

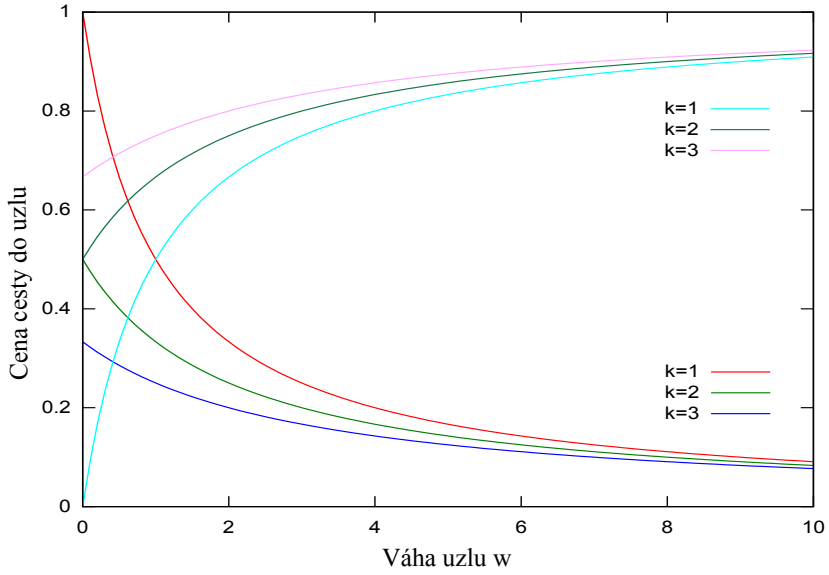
Otázkou zůstává, jaké chceme, aby měla takováto cesta vlastnosti. Může být např. nejkratší, nejrychlejší atd. Obecně není příliš dobré, aby se robot pohyboval nejkratší cestou. Znamenalo by to totiž těsné objíždění překážek a tím pádem vzhledem k nepřesnosti senzorů i zvýšenou pravděpodobnost kolize. Navíc ani člověk, pokud jde například po chodbě, nevolí nejkratší cestu, ale jakýsi kompromis mezi délkou cesty a její "bezpečností". Pokud třeba potřebujeme zahnout za roh, většinou jej neobcházíme příliš blízko, ale necháváme určitou mezeru ode zdi, abychom viděli dále před sebe a mohli tak dříve reagovat na případného protijdoucího člověka. Analogické chování tak požadujeme i od robota. To mu mohou zajistit např. potenciálová pole, která se snaží vybírat takovou cestu, která je co nejdále od překážek. My použijeme jejich následující navrženou analogii.

Každému bodu ve volném prostoru přiřadíme hodnotu w, která bude odpovídat ceně, nebo také riskantnosti cesty přes daný vrchol. Chceme, aby hodnota w byla přímo úměrná vzdálenosti vrcholu ve volném prostoru od vrcholu na hranici. Hodnotu tedy přirozeně zvolíme jako nejmenší počet vrcholů, přes které se dostaneme z ohodnocovaného vrcholu k nejbližšímu[1] hraničnímu vrcholu. V případě už vytvořené volné plochy je ohodnocování jednoduché. Hraničním vrcholům přiřadíme hodnotu 0, bodům, které mají s nimi přímou spojnici hodnotu 1 atd. Tímto způsobem můžeme ohodnocovat body jen v případě už vytvořené mapy, jinak bychom museli přepočítat hodnoty všech vrcholů po každém přidání nového vrcholu. Abychom se tomu vyhnuli, můžeme hodnotu vrcholů určit následujícími dvěma způsoby.

1. Hodnoty vrcholů přepočítat až poté, co robot dosáhne své cílové pozice a bude nutné nalézt cestu k novému cíli.

2. Vypočítat nové hodnoty po každém přidání nového vrcholu, ale jen v té části triangulace, která byla tímto přidáním ovlivněna, tzn. u těch bodů, kterým byla nějaká hrana přidána nebo odebrána. Způsob ohodnocení v tomto případě bude stejný, jako kdybychom ohodnocovali celou volnou plochu, nově ohodnocených vrcholů ale bude podstatně méně.

[1] Nejbližším není myšleno podle vzdálenosti, ale podle počtu vrcholů, které je třeba překonat.



Obrázek 4.4: Závislost ceny cesty do uzlu na váze uzlu w. Průběhy v horní části grafu jsou podle vzorce (4.1), v dolní části grafu podle vzorce $1/(w+k)$.

Pokud má robot zmapovat danou oblast co nejrychleji, je nutné zajistit, aby jezdil vždy trochu jinou trasou. V ideálním případě by tato nová trasa měla částečně překrývat starou tak, aby dohromady vytvořily jednotný celek. Jako vhodná se jeví trasa, naplánovaná po hraničních vrcholech. Takovou trasu by mělo zajistit vhodné naplánování cesty, při kterém musí robot sice minimalizovat délku své cesty, ovšem při maximalizaci počtu vrcholů ležících na hranici.

Pro ohodnocení uzlů bychom mohli použít pouze jejich váhu w, tím by ale vytvořená funkce byla lineární. Pro naše potřeby je vhodnější, aby ohodnocovací funkce se vzrůstající vzdáleností od hraničního vrcholu nejprve prudce a potom stále pozvolněji rostla. To odpovídá našemu požadavku, aby se robot Navržený výpočet vah má tedy tvar

$$1 - \frac{1}{w+k} \tag{4.1}$$

Hodnotu $k > 1$ připočítáme proto, aby nám ohodnocení hraničních bodů nevyšlo nekonečně velké a také tím udáváme, jak moc chceme případně penalizovat vrcholy. Aby nám váha podle rovnice (4.1) nevyšla rovna nule (viz graf 4.4), což je nevhodné, musí být k větší než jedna. Váha vrcholu ω může nabývat hodnoty celých čísel větších než nula, a protože zlomek potřebujeme pro nejlepší, tzn. hraniční, vrcholy minimalizovat, odečteme jej od jedničky. Hodnotu cesty z uzlu v_S do v_N vypočítáme jako

$$G_E(v_S, v_N) = \sum_{n=1:N} \left(1 - \frac{1}{w_n + k}\right) d(v_{n-1}, v_n), \tag{4.2}$$

Subskript E značí, že se jedná o cestu tzv. "objevitelskou". $d(v_{n-1}, v_n)$ je vzdálenost mezi vrcholem v_{n-1} a vrcholem v_n. Zlomek $1 - 1/(w_n + k)$ udává cenu přechodu do vrcholu v_n a N je celkový počet vrcholů, které musíme překonat, v_S je počáteční vrchol.

Opačným případem je situace, kdy už máme mapu částečně nebo zcela hotovou a v tom případě nechceme, aby robot používal cesty vedoucí po hranici. Místo toho chceme, aby

naplánovaná cesta vedla dále od překážek a byla tak pro robota bezpečnější. Cenu cesty do cíle tak vypočítáme obdobně, jako v předchozím případě.

$$G_w(v_S, v_N) = \sum_{n=1:N} \frac{1}{w_n + k} d(v_{n-1}, v_n). \tag{4.3}$$

Funkcí, kterou tedy budeme minimalizovat, je

$$F(v_S, v_G) = G_i(v_S, v_N) + H(v_N, v_G), \tag{4.4}$$

kde $F(v_S, v_G)$ je odhadovaná cena váhovaného přechodu z počátečního uzlu v_S k cílovému uzlu v_G. Skládá se z hodnoty $G_i(v_S, v_N)$, kde $i \in \{E, w\}$ a heuristiky $H(v_N, v_G)$, což je odhad ceny cesty z uzlu v_N do cílového uzlu v_G.

Poslední funkcí, kterou použijeme, je funkce používaná u algoritmu A*. V tomto případě už se nepočítá s vahou uzlů a funkce nalezne nejkratší cestu mezi dvěma vrcholy.

$$F(v_S, v_G) = G(v_S, v_N) + H(v_N, v_G), \tag{4.5}$$

Tuto funkci použijeme pro otestování vzdálenosti objevitelské cesty. Pokud bude poměr této nalezené cesty a nejkratší cesty větší než námi zvolená konstanta, bude místo objevitelské cesty použita tato nejkratší cesta.

Testování průjezdnosti

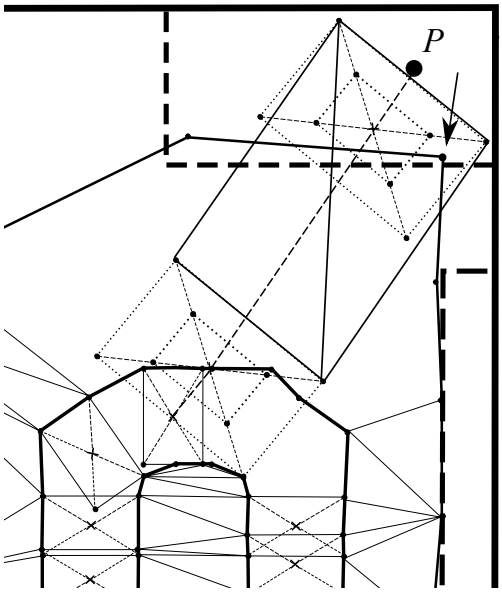
Poté, co robot naplánoval svou cestu, je potřeba části, které vedou po hranicích, otestovat, zda jsou průjezdné. Stejným způsobem se pak bude testovat i průjezdnost cesty naplánované od hranice k cílovému bodu. Protože robot nezná přesně svoji polohu a ani polohu bodů, je dobré kolem něj vytvořit jistou *bezpečnou oblast* a při testování průjezdnosti testovat na kolizi celou tuto oblast.

Aby se omezilo množství potřebných testů, budou se na možnou kolizí testovat jen body se z-ovou souřadnicí menší než je výška bezpečné oblasti kolem robota a dále ty body, které budou spojeny přímou hranou z bodů, po kterých bude naplánována cesta. Ke kolizi dojde v případě, když testovaný bod, nebo jeho průnik s rovinou cesty v případě bodů spojených s hranicí, bude ležet uvnitř kvádru, který svým pohybem vytvoří robot. Velikost tohoto kvádru se bude brát včetně bezpečné oblasti okolo robota. Testovat se bude tak, že se body promítnou do podlahy mapy a pokud budou ležet uvnitř trojúhelníků, ze kterých dočasně vymezíme cestu robota, bude to znamenat potenciální kolizi.

Vyhlazování cesty

V případě použití pravděpodobnostního plánování, což byla inspirace pro zde navržené plánování cesty po bodech, po kterých se robot pohyboval, není cesta příliš hladká a je dobré ji v postprocessingu vyhladit. Tím získáme cestu, která bude obsahovat méně zatáček a celkově bude kratší. Nevýhodou ovšem je to, že nově nalezené spojnice mezi body se musejí znovu otestovat kvůli případné kolizi.

Pokud ke hledání cesty používáme algoritmy vycházející z A*, nedá se předpokládat, že bychom nalezli cestu ještě kratší. Je ovšem třeba si uvědomit, že pokud máme nalezeny nejkratší cesty mezi body A, B a B, C, není nejkratší cesta mezi A a C cesta procházející bodem B. Navíc, protože cestu plánujeme v neznámém, dosud neobjeveném prostředí a



Obrázek 4.5: Schéma znázorňující naplánovanou trasu z pozice na hranici k bodu P. Potenciální polohy robotů jsou zde vyznačeny tečkovanou čarou a to včetně bezpečné oblasti kolem robota. Dva trojúhelníky spojující přední hrany bezpečných oblastí vymezují oblast, do které nesmí padnout žádný detekovaný bod. To by znamenalo překážku v pohybu robota směrem k naplánovanému cíli, jak je ukázáno šipkou.

algoritmy pro hledání cesty ji hledají v prostoru známém, může mezi body A a C existovat i cesta kratší, než ta nalezená pomocí A*.

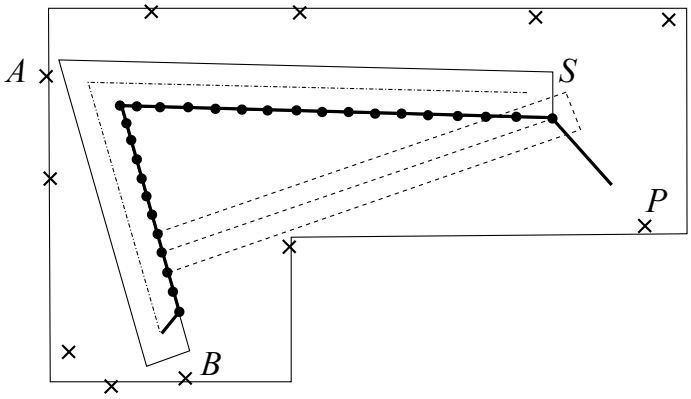
Kratší cestu nalezneme v případě použití algoritmů pro vyhlazování cesty. Nejznámější takovýto algoritmus je tzv. hladový (ang. greedy) a je popsán např. v [14]. V jednoduchosti funguje tak, že se snaží spojit přímo cílový bod v_G s bodem počátečním v_S, pokud se to nepovede, snaží se cílový bod připojit k bodu v_{S+1}, atd. Až uspěje, snaží se bod v_0, nově připojený přímo k bodu v_G opět spojit s počátečním bodem v_S atd. Jak už bylo řečeno, nevýhoda tohoto přístupu je v tom, že musíme každou novou spojnici testovat, zda je cesta po ní průjezdná. Na druhou stranu je to vyváženo možným nalezením mnohem kratší cesty a objevením většího prostoru. Ukázka nalezení kratší cesty je na Obr. 4.6.

Vyhýbání se překážkám

Plánování cesty, tak jak je navrženo v této práci, je nedostatečné pro vyhýbání se neočekávaným překážkám, na které robot může narazit při cestě neprozkoumaným prostorem. Plánování cesty jen po bodech, které leží na jeho rozích, případně ve středu, nebude dost jemné na to, aby se robot překážce vyhnul a mohl pokračovat směrem k naplánovanému cíli. Překážku by musel buď složitě objíždět a nebo naplánovat cestu ke zcela jinému cílovému bodu.

Vhodnou metodou pro plánování cesty při vyhýbání se překážkám je pravděpodobnostní plánování, zejména pak jednodotazové algoritmy. Ty mají navíc tu výhodu, že hledají cestu jen mezi startovním a cílovým bodem a není tak třeba navzorkovávat celý prostor.

Po vygenerování náhodného bodu cesty bude tento bod spojen s předchozím bodem cesty. Testování volného přechodu mezi těmito dvěma body se provede stejně, jako je



Obrázek 4.6: Ukázka naplánované cesty (tučně) a nalezené kratší (čárkovaně) pomocí vyhlazování. Robot jel z místa S prozkoumat bod A, pak bod B a nakonec se rozhodl prozkoumat bod P. Body detekované kamerami jsou zobrazeny křížkem, kolečka představují body na jedné straně hranice průjezdné oblasti, po kterých se naplánovala nejkratší cesta.

popsáno v části Testování průjezdnosti. Toto je také rozdíl oproti testování průjezdnosti u pravděpodobnostích algoritmů.

Přibližování se k překážkám

Robot by se neměl přiblížit k žádné překážce na menší vzdálenost, než je jeho *slepý bod* (obr. 4.7). Ten je určen vzájemnou vzdáleností kamer a jejich zorným úhlem. Aby byl robot schopen detekovat překážku, musí ji vidět zároveň oběma svými kamerami. To znamená, že pro robota nemá žádný význam přiblížit se k překážce co nejblíže, protože by ji stejně nebyl schopen detekovat a navíc by mu hrozila možná kolize. Vzdálenost slepého bodu určíme výpočtem z podobných trojúhelníků.

$$\tan\frac{\alpha}{2} = \frac{f}{s/2} \tag{4.6}$$

$$\tan\frac{\alpha}{2} = \frac{p}{B/2} \tag{4.7}$$

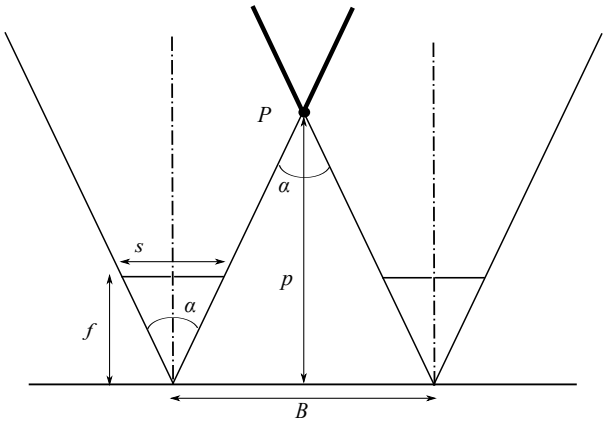
kde α je zorný úhel, s je velikost snímače, f je ohnisková vzdálenost a B je vzájemná vzdálenost kamer. Vzdálenost p slepého bodu bodu dostaneme jako

$$p = B\frac{f}{s} \tag{4.8}$$

Robot se může na nulovou vzdálenost přiblížit pouze k průmětu prozkoumávaného bodu P do podlahy a to pouze pokud budeme uvažovat i určitou bezpečnou oblast kolem něj. Přímá vzdálenost robota od bodu P navíc při tom nemůže být menší než je vzdálenost slepého bodu.

4.2.3 Výběr cílů

Zbývá určit, jakým způsobem bude robot vybírat pořadí prozkoumávaných cílů a tím pádem vlastně vytvořit strategii pro tvorbu mapy. Máme dvě možnosti, jak mapu vytvořit:



Obrázek 4.7: Schéma nejbližšího bodu viditelného oběma stereokamerami. V tučně vyznačené oblasti leží body, které je robot schopen detekovat.

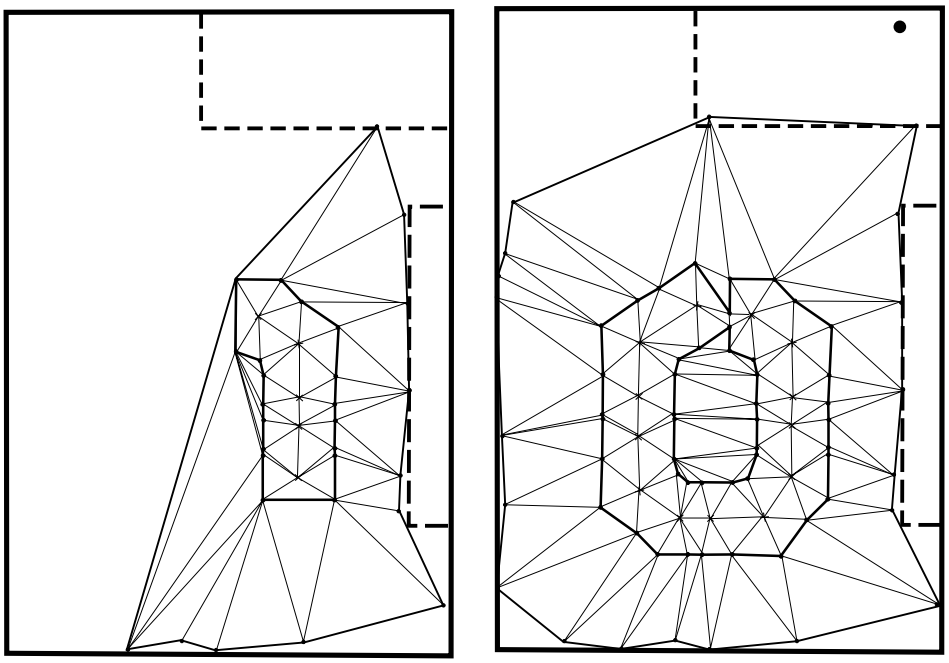
- Robot se snaží prozkoumat prostředí místo vedle místa tak, aby už během prvního průchodu vytvořil souvislou hranici mezi průchozím a neprůchozím územím, tzn. aby danou oblast zmapoval co nejpřesněji už od samého začátku mapování.
- Robot chce objevit co největší území za co nejkratší čas a nebere v úvahu malá neprozkoumaná místa na okrajích mapy. Tato neprozkoumaná místa robot zmapuje ve druhé fázi mapování, kdy bude upřesňovat už vytvořenou mapu.

Protože se robot při tvorbě mapy v ní zároveň lokalizuje, je nutné, aby se po určité době vrátil zpět na místo, kde už jednou byl, ovšem jinou cestou a uzavřel tak smyčku [83]. Uzavření smyčky je základní předpoklad pro tvorbu přesné mapy algoritmem SLAM. Je také dobré takovouto situaci rozpoznat a pokud k ní dojde, provést převzorkování. Robot ovšem tím, jak se pohybuje, ví stále méně a méně o tom, kde se nachází a než dojde k uzavření smyčky, neměl by zbytečně zvětšovat nejistotu své polohy. To znamená, že by měl omezit na minimum především rotaci, která, následována pohybem vpřed, nejvíce zvětšuje nejistotu polohy. Tento požadavek ve výsledku znamená, že je nežádoucí, aby robot před tím, než dojde k uzavření smyčky, vytvářel podrobnou mapu už od začátku mapování.

Z předchozí úvahy plyne, že je výhodnější, aby byl plánovací algoritmus rozdělen na dvě části. V první části robot zmapuje co největší území, i když nedokonale[2], s tím, že se bude snažit se vždycky po určité době vrátit na místo, kde už jednou byl. V ideálním případě by tímto místem měl být začátek mapování, tzn. bod [0, 0], protože tam zná robot svou pozici zcela přesně. Teprve ve druhé části mapování bude upřesňovat mapu tak, aby vytvořil co nejpřesnější hranici mezi volnou a obsazenou oblastí.

Robot, který má nulovou znalost o svém okolí, nemůže dopředu vědět, kudy se vydat, aby se mu podařilo znovu se vrátit do stejného místa. Kromě toho není ani zaručeno to, že prostor bude mít požadovaný tvar.

[2]Nedokonalostí je myšleno to, že nebudou přesně určené hranice mezi obsazeným a volným prostorem. Důležitější v této fázi mapování je to, aby mapa byla správná globálně, protože drobné nepřesnosti robot domapuje ve druhé fázi.



Obrázek 4.8: Schéma dvou okamžiků tvorby mapy. Na levém obrázku je robot po čtyřech posunutích s částí už triangulovaného prostoru. Čárkovaně jsou vyznačeny překážky. Na pravém obrázku robot ukončil pohyb ve tvaru elipsy, bod v horním pravém rohu představuje příští cíl pohybu.

Hrubá tvorba mapy

Cílem hrubé tvorby mapy je zmapovat co největší část prostoru za co nejkratší dobu (obr. 4.8). Pro kvalitu mapy by navíc bylo vhodné, aby robot nejprve zmapoval vnější smyčku a pak teprve pokračoval v mapování dalších částí. Robot ovšem neví, kde v mapě se nachází, a ani není schopen rozpoznat, zda je na vnější smyčce. Navržený způsob, jak vnější smyčku detekovat, nebo spíše zmapovat, je následující. V okamžiku, kdy robot začne mapovat prostor, začne se pohybovat vpřed a tento směr si označí úhlem 0°. Pokud předpokládáme, že bude schopen detekovat směry koridorů, ve kterých se pohybuje (mohou to být např. ulice nebo chodby), bude zatáčet jen na křižovatkách a v případě, že budou zatáčet samy koridory. Pokaždé, když robot zatočí, přičte nebo odečte si úhel o který zatočil od svého stávajícího úhlu (ten bude na začátku již uvedených 0°). V okamžiku, kdy se robot dostane na křižovatku a bude si moci vybrat z více možných cest, vybere si tu, která je ve směru, který mu nejvíce minimalizuje absolutní hodnotu jeho úhlu natočení. Tímto způsobem se robot bude snažit pohybovat v přímém směru, který ho navede na vnější smyčku. Řekněme, že po ní se bude pohybovat proti směru hodinových ručiček. Uvedený způsob se ho bude snažit na křižovatce zatočit doprava, což bude mít za následek to, že se robot bude pohybovat po největší možné smyčce. V okamžiku, kdy detekuje, že je na již jednou prozkoumaném území, bude smyčka uzavřena.

Nyní robot pokračuje v mapování vnitřku smyčky. Zde navrženou metodou pro tuto část mapování je vybrat si vždy ten detekovaný bod, který má největší vzdálenost od hraničních bodů. *Přitažlivost* detekovaných bodů bude ovšem zmenšena o vzdálenost, kterou bude robot potřebovat, aby se dostal k jejich nejbližšímu hraničnímu bodu. Výsledná přitažlivost,

se tedy vypočítá jako

$$A_P = d(P_p, P_h) - \eta G_i(v_S, P_h) \tag{4.9}$$

kde A_P je přitažlivost bodu P, $d(P_p, P_h)$ je vzdálenost od hraničního bodu P_h k bodu P_p což je průmět bodu P do podlahy a $G_i(v_S, P_h)$ je cena cesty z robotovy současné pozice v_S do hraničního bodu. Index $i \in \{E, A^*\}$ udává, zda se vybere cesta "objevitelská"nebo nejkratší. Kromě těchto dvou cest se může také použít vyhlazená cesta. Parametr η určuje, jak moc chceme penalizovat cestu, po které robot musí dorazit do hraničního bodu. Protože bodů v mapě bude velké množství, budou se cílové body vybírat jen z těch, které budou mít přímou spojnici s hraničními body typu 2b.

První fáze mapování bude ukončena v okamžiku, kdy vzdálenost $d_o(P, P_h)$ žádného bodu nebude větší než zvolená konstanta $\mathcal{K}$. Její velikost bude záležet na velikosti mapovaného území.

Upřesnění mapy

Po ukončení první fáze, tj. hrubé tvorby mapy, nastává fáze druhá - zpřesňování mapy. V ní by měl robot co nejpřesněji určit hranice mezi volným a obsazeným prostorem. K tomuto zpřesňování budou sloužit body 2c, popsané v předchozí části. Jsou to body, ležící na hranici volného území, které nelze použít k plánování cesty, protože by to vedlo ke kolizi s překážkou. Cílem robotova upřesňování mapy tedy je, aby se ze všech hraničních bodů staly body typu 2c. V okamžiku, kdy toto nastane, můžeme považovat tvorbu mapy za ukončenou[3]. U zpřesňování mapy už tak nezáleží na pořadí výběru cílů, a tak můžeme jednoduše prozkoumávat hraniční body seřazené podle vzdálenosti od robota.

V průběhu této fáze se může stát, a je to pro komplexnější prostředí pravděpodobné, že robot narazí na větší neprozkoumané území. Zjistí to tak, že detekuje bod, který bude ve vzdálenosti větší než zvolená konstanta $\mathcal{K}$. Potom se přepne zpět do první fáze mapování a vydá se na průzkum nově objeveného území.

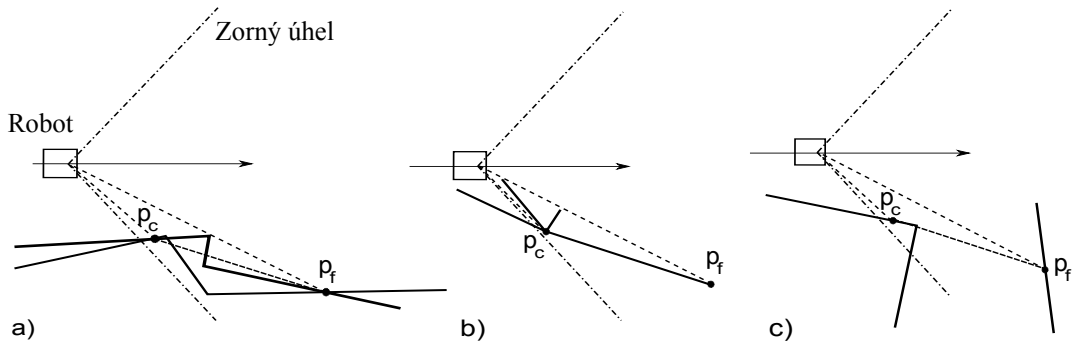
4.2.4 Vodorovný pohyb kamerou

Shopnost pohybovat kamerami v průběhu mapování zlepšuje možnost prozkoumat okolní prostředí. Robot se může pohybovat jedním směrem, zatímco kamery se dívají jiným. Toto robotovi umožní prozkoumat během svého pohybu mnohem větší území, než kdyby měl kamery připevněny napevno. Samozřejmě robot nesmí věnovat veškerý svůj čas během pohybu sledování okolí napravo a nalevo od něj, protože musí také kontrolovat cestu před sebou kvůli výskytu neočekávaných překážek.

Předokládáme, že kamery (stereokamera) jsou umístěny na pohyblivé platformě, která umožňuje ve vodorovném směru rozsah pohybu 180° a ve svislém natočit kamery kolmo vzhůru (a samozřejmě také směrem k zemi, ale tady jsme omezeni konstrukcí robota). Pokud počítáme s tím, že úhel pohledu robota (FOV) je 90°, jsme tak schopni vodorovně pokrýt oblast 270° okolo robota.

Při detekování význačných bodů a při následné triangulaci je přirozené, že budou navzájem spojeny body, které neleží na jedné rovině, ale mezi nimiž je další volný prostor, viz příklad na obr. 4.9.

[3]Toto samozřejmě platí jen částečně. Mapa se bude vytvářet stále během robotova pohybu v ní. Vzhledem k tomu, že používáme kamery, může i malá změna osvětlení znamenat, že robot bude detekovat stále nové a nové body. Ukončení mapování znamená pouze to, že je u konce hlavní část tvorby mapy.



Obrázek 4.9: Příklady různých druhů prostředí zobrazené ve 2D při pohledu směrem shora. Čerchovanou čarou je vyznačen zorný úhel robota, šipkou pak jeho směr pohybu. Body p_c a p_f jsou bližší a vzdálenější *body zájmu* spojené hranou. Silná plná čára vyznačuje možný tvar prostředí. Na obrázku a) je mezi oběma body roh. Na obrázku b) je příklad čtyř možností prostředí. Silná čára v tomto případě může představovat např. dveře. Na obrázku c) je příklad dvou bodů spojených chybně hranou. Tato chybná hrana může být odstraněna nalezením dalších bodů mezi těmito dvěma body.

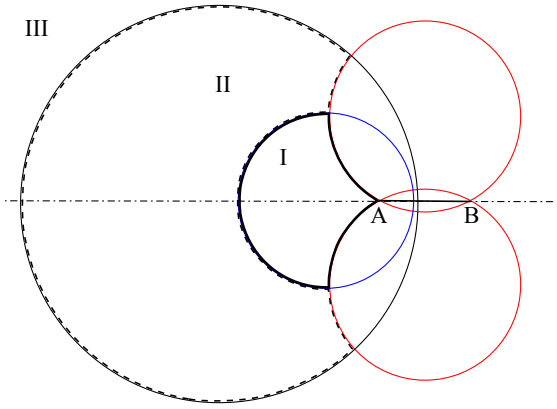
Robot by měl tuto situaci rozpoznat a při pohybu kolem uvedených bodů (hran), se na ně zaměřit svými kamerami.

Detekce hran zájmu

Hrana zájmu, nebo také zajímavá hrana, bude taková, jejíž dva krajní body, budou v obraze kamery ležet blízko u sebe, avšak jejich vzdálenosti od kamery budou velmi rozdílné. Příklad viz na obrázku 4.9, kde body p_c a p_f budou v obraze kamery ležet blízko sebe, ale vzdálenost bodu p_f od kamery je téměř třikrát větší, než u bodu p_c. Tak si můžeme definovat, na čem bude míra zajímavosti záviset:

- Úhel, který svírá hrana k poloze kamery - čím menší tento úhel bude, tím bude míra zajímavosti větší.
- Zorný úhel, pod kterým vidíme oba body - opět, čím je tento úhel menší, tím větší bude míra zajímavosti.
- Vzdálenost hrany ke kameře - čím je hrana blíže, tím je její zajímavost větší.
- Délka hrany - čím je hrana delší, tím roste její míra zajímavosti.

Délku hrany zde uvažovat nebudeme, protože do značné míry závisí na velikosti robota. Úhel, který svírá hrana k poloze kamery můžeme nahradit výpočtem r, což bude poměr vzdáleností obou bodů. Tím také do jisté míry zavedeme závislost na vzdálenosti hrany, protože čím bude hrana dále, tím bude tento poměr bližší jedničce. Hodnoty prahů pro r musíme stanovit empiricky. Například pokud r bude v intervalu $\langle 0; 0,5\rangle$ bude hrana označena jako potenciálně zajímavá. Pokud bude r v intervalu $\langle 0,5; 0,95\rangle$, bude sice hrana také označena jako potenciálně zajímavá, ale nebude sledována z místa polohy robota, protože její vzdálenost od robota bude buďto příliš velká, nebo bude hrana příliš krátká. Pokud bude r ležet v intervalu $\langle 0,95; 1\rangle$, hrana jako zajímavá označena nebude. Poměr r



Obrázek 4.10: Na obrázku jsou vyznačeny Apolloniovy kružnice bodů A a B. Pokud bude robot hranu A-B detekovat z oblasti I, bude hrana označena jako zajímavá a robot ji bude sledovat. Pokud bude hranu detekovat z čárkované oblasti II, označí ji jako potenciálně zajímavou, ale protože je daleko, sledovat ji bude až v okamžiku, kdy se dostane do oblasti I. Pokud bude mimo tyto dvě oblasti v oblasti III, hranu jako zajímavou neoznačí.

musí být zkombinován se zorným úhlem φ, pod kterým vidíme oba body. To zabrání tomu, aby byla jako zajímavá označena hrana, u které by se robot nacházel mezi jejími krajními body.

Hrana tedy bude označena jako zajímavá v případě, že splní obě podmínky

$$r \in \langle 0; 0,5\rangle \wedge \varphi\langle 0; \pi/6\rangle, \tag{4.10}$$

kde $r = d(v_S, p_c)/d(v_S, p_f)$ a hodnota $\pi/6$ je opět stanovena empiricky.

Pokud hrana splní pouze slabší podmínku, kde $r \in \langle 0,5; 0,95\rangle$ hrana bude označena jako potenciálně zajímavá, ale robot ji bude sledovat až v případě, pokud se dostane do místa blíže k ní.

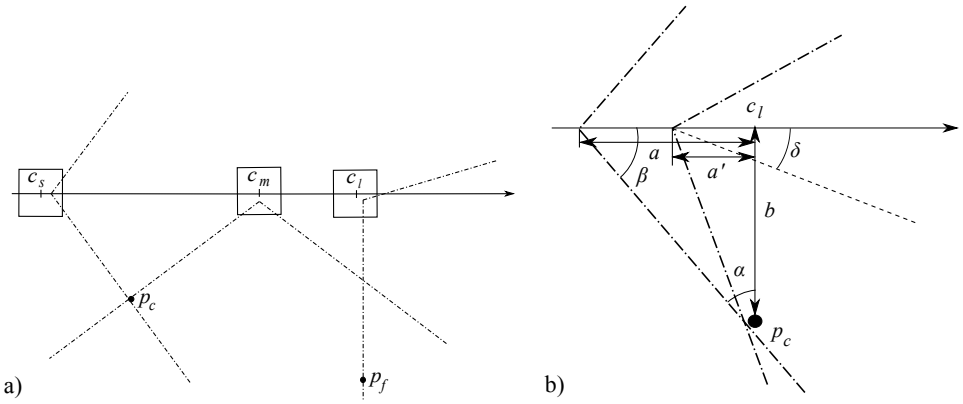
Konstantní poměr vzdáleností od dvou bodů splňuje jeden druh kružnice, nazývaný Apolloniova kružnice. Tato kružnice v sobě obsahuje bližší bod a její střed leží na prodloužení hrany. Druhou podmínku, týkající se zorného úhlu, splňuje jiný druh Apolloniovy kružnice, která prochází oběma krajními body hrany. Oba druhy kružnice i s vyznačenou oblastí zájmu jsou ukázány na obrázku 4.10.

S tím, jak se bude robot prostorem pohybovat, budou se hodnoty r a φ u zajímavých hran přepočítávat a robot je bude sledovat svými kamerami. Pokud se dostane z oblasti na obrázku 4.10 označené jako I do oblasti III, bude hrana přeznačena jako nezajímavá a dále se už sledovat nebude. Pokud robot v oblasti mezi body p_c a p_f nalezne jiný bod, bude původní hrana nahrazena jinou triangulací a pokud hrany této triangulace budou splňovat podmínku zajímavosti, bude se ve sledování kamerami pokačovat. Pokud nebudou, kamery se vrátí zpět do výchozí pozice.

Příklady popsané v následujících částech počítají s tím, že mezi body p_c a p_f původní hrany budou nalezeny další body a kamera tak bude pokračovat ve své rotaci.

Sledování zájmových bodů

Robotovo chování při zkoumání neznámého prostoru by mělo být co nejvíce podobné chování člověka při stejné činnosti. Předpokládáme, že lidské chování v tomto případě je



Obrázek 4.11: Na obr. a) je příklad robota pohybujícího se zleva doprava ve směru šipky. V bodě c_s robot začne otáčet kamerami doprava tak, aby měl bod p_c stále ve svém zorném poli. V bodě c_m robot nenávratně ztratí bod p_c z dohledu a tak začne otáčet kamerami nazpět. V bodě c_l robot vidí bod p_f okrajem svého zorného pole, tento okraj je zároveň kolmo na směr robotovy trasy. Na obr. b) je naznačeno postupné otáčení kamer tak, aby byl bod p_c stále v zorném poli robota.

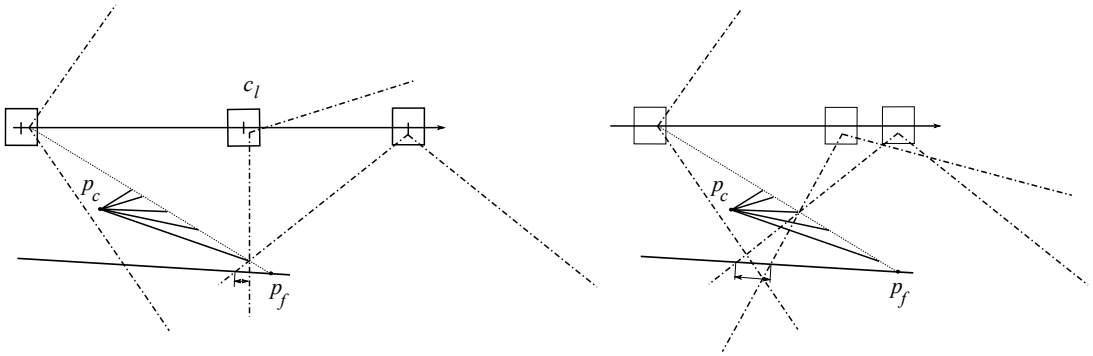
v jistém smyslu slova ideální. Člověk se při cestě neznámým územím dívá střídavě doleva a doprava, podle toho, kde očekává objevení nějakých nových prostor, případně chce-li si něco lépe prohlédnout. V určitých okamžicích může i zpomalit nebo dokonce až zastavit, aby si prohlédl místa, která jsou na stejné úrovni, ale po jeho levé i pravé straně. Určování, kdy a kam se otočit, je však u obou poněkud odlišné. Robot podle navrženého algoritmu spoléhá pouze na detekování význačných bodů, zatímco člověk vnímá okolní prostor mnohem komplexněji.

V okamžiku, kdy robot detekuje pár zajímavých bodů, se je bude snažit udržet ve svém zorném poli (obr. 4.11). To znamená, že v bodu c_s začne otáčet svými kamerami. Rotace stereokamery bude pokračovat až do bodu c_m, kde dosáhne svého maxima, tzn. předpokládáme 90°. Po dosažení tohoto bodu se začne kamera otáčet zpět a to takovou rychlostí, aby robot v bodu c_l okrajem svého zorného pole viděl vzdálenější bod p_f.

Nejlepší místo pro průzkum dvou zajímavých bodů a hrany mezi nimi je z kolmice na tuto hranu a procházející jejím středem. Bohužel tato kolmice nemusí protínat naplánovanou cestu (a ani rovina jí proložená), nebo tento průsečík bude daleko od robotovy současné pozice. Proto tento přístup není prakticky proveditelný a místo něj byl tedy navržen popsaný způsob.

Dalším problémem, který bylo potřeba vyřešit je, kdy začít s otáčením kamerami zpět do výchozí pozice. V ideálním případě by zůstaly kamery natočené na svoje maximum až do doby, kdy se ze zorného pole ztratí vzdálenější bod p_f. To by ovšem neúnosně prodlužovalo dobu, kdy robot nevidí před sebe. Pokud by se začalo otáčet kamerami maximální rychlostí nazpět, neviděl by robot dobře za případné objekty mezi body p_c a p_f. Proto bylo zvoleno popsané řešení. Na obr. 4.12 vidíme, že pokud začneme otáčet kamerami zpět v okamžiku, kdy bližší bod p_c zmizí ze zorného pole, je rozdíl prohlédnutého prostoru minimální vzhledem k uražené vzdálenosti a tím i k době, kdy robot nevidí před sebe.

Vzdálenost a (na obr. 4.11b) od bodu c_l, což je průsečík kolmice na cestu robota,



Obrázek 4.12: Na obou obrázcích vidíme různé způsoby otáčení kamer zpět a rozdíl prozkoumaného prostoru (znázorněn úsečkou). Robot v prostřední pozici se otáčí tak, aby byl kolmo na vzdálenější bod krajem svého zorného úhlu. Robot v pravé pozici pokračuje v cestě s maximálně natočenými kamerami.

procházející bodem p_c, ve které bližší bod zmizí ze zorného pole, je dána vztahem

$$a = b\frac{\cos\varphi/2}{\cos\alpha} \tag{4.11}$$

kde b je vzdálenost bodu p_c od trasy robota a $\varphi/2$ je polovina zorného pole robota. Úhel α pak spočítáme jako $\alpha = 180 - 90 - \beta$. Ve vzdálenosti a od bodu c_l je vztah 4.15 roven nule a robot zde musí začít otáčet svými kamerami.

Úhel δ, o který se musí robot otočit, odvodíme podle obrázku 4.11b

$$\tan\beta = \frac{b}{a} \tag{4.12}$$

$$\delta = \frac{\varphi}{2} - \beta \tag{4.13}$$

$$\beta = \arctan\frac{b}{a} \tag{4.14}$$

Pokud do rovnice (4.13) dosadíme rovnici (4.14), dostaneme

$$\delta = \frac{\varphi}{2} - \arctan\frac{b}{a} \tag{4.15}$$

Při otáčení může dojít k situaci, kdy bude bod c_l ležet před bodem c_m. To by znamenalo, že robot má začít otáčet kamerami zpět dříve, než bližší bod zmizí ze zorného pole. To by ovšem způsobilo zmenšení prozkoumaného prostoru, a proto bude mít požadavek na počátek otáčení zpět vyšší prioritu než vlastnost, že má robot vidět vzdálenější bod na okraji svého zorného pole pod kolmým úhlem.

V průběhu průzkumu se může stát, že se robot bude muset podívat na opačnou stranu ještě dříve, než dokončí průzkum zajímavých bodů na první straně (obr. 4.13). Lidské chování nám v tomto případě může opět posloužit jako ideální vzor. Pokud člověk během chůze potřebuje prozkoumat místa po obou stranách, začne tím bližším a v okamžiku, kdy myslí, že by mu oblast na druhé straně zmizela z dohledu, zapamatuje si prozkoumanou oblast

na první straně a rychle otočí hlavu na druhou stranu, aby ji prozkoumal. A opět, když myslí, že by mu zmizela hranice prozkoumané oblasti na první straně, opět si zapamatuje prozkoumanou oblast na této straně a otočí hlavu zpět na první stranu. Jestliže jsou obě hranice zkoumaných oblastí téměř na stejné úrovni, zpomalí a může i zastavit, aby mohl prozkoumat obě dvě.

Podobně jako člověk, tak i robot nejdříve prozkoumá tu hranu, která je na bližší úrovni. Pokud zjistí, že není schopen prozkoumat bod na protější straně, může také zpomalit, nebo i zastavit v případě, že obě hrany leží na stejné úrovni. Na rozdíl od člověka není robot schopen si zapamatovat hranici, nebo celou oblast, kde dokončil prozkoumávání. Může si ale na zkoumané hraně vytvořit bod p_A, kterým si bude označovat dosud prozkoumanou oblast. Tento bod mu v podstatě nahradí bod p_c. Tento bod bude ležet na kolmici k trase robota, která trasu protíná v bodě, ve kterém je kraj robotova zorného úhlu kolmý na tuto trasu (viz obr. 4.13). Poté, co robot dokončí zkoumání protější oblasti, musí se otočit zpět tak, aby opět viděl tento bod. Nejzazší místo, odkud ho může vidět, je bod c_{aR}.

Až dosud jsme se zabývali situací, kdy body byly víceméně rovnoběžně se zemí. I v případě, že by dva zajímavé body byly nad sebou, probíhalo by jejich zkoumání popsaným způsobem. Průzkum více bodů a hran, umístěných nad sebou by také probíhal obdobně, jen s tím rozdílem, že robot by začal kamerami otáčet zpět až poté, co by z jeho zorného pole zmizel poslední bližší bod.

4.2.5 Svislý pohyb kamerou

Během průzkumu se může stát, že se některý z dvojice zajímavých bodů dostane mimo zorné pole robota. V tom případě je nutné, aby se robot podíval svými kamerami vzhůru. Stejně tak, když se dostane k cílovým bodům, dá se předpokládat, že budou mimo zorné pole a bude nutné podívat se vzhůru. Protože v tomto případě je cílem robota prozkoumat i okolí tohoto bodu, měl by mít cílový, tzn. zkoumaný bod ve středu zorného pole. Vztah mezi úhlem osy pohledu kamer α a vzdáleností od bodu P, odvozený z pravoúhlého trojúhelníka (obr. 4.14), je následující:

$$\alpha = \arctan\left(\frac{h-l}{d}\right) - \frac{\beta}{2} \tag{4.16}$$

kde h je výška bodu od podlahy, l je výška kamer od podlahy, d je vzdálenost kamer od svislice z bodu do podlahy a β je svislý zorný úhel kamery.

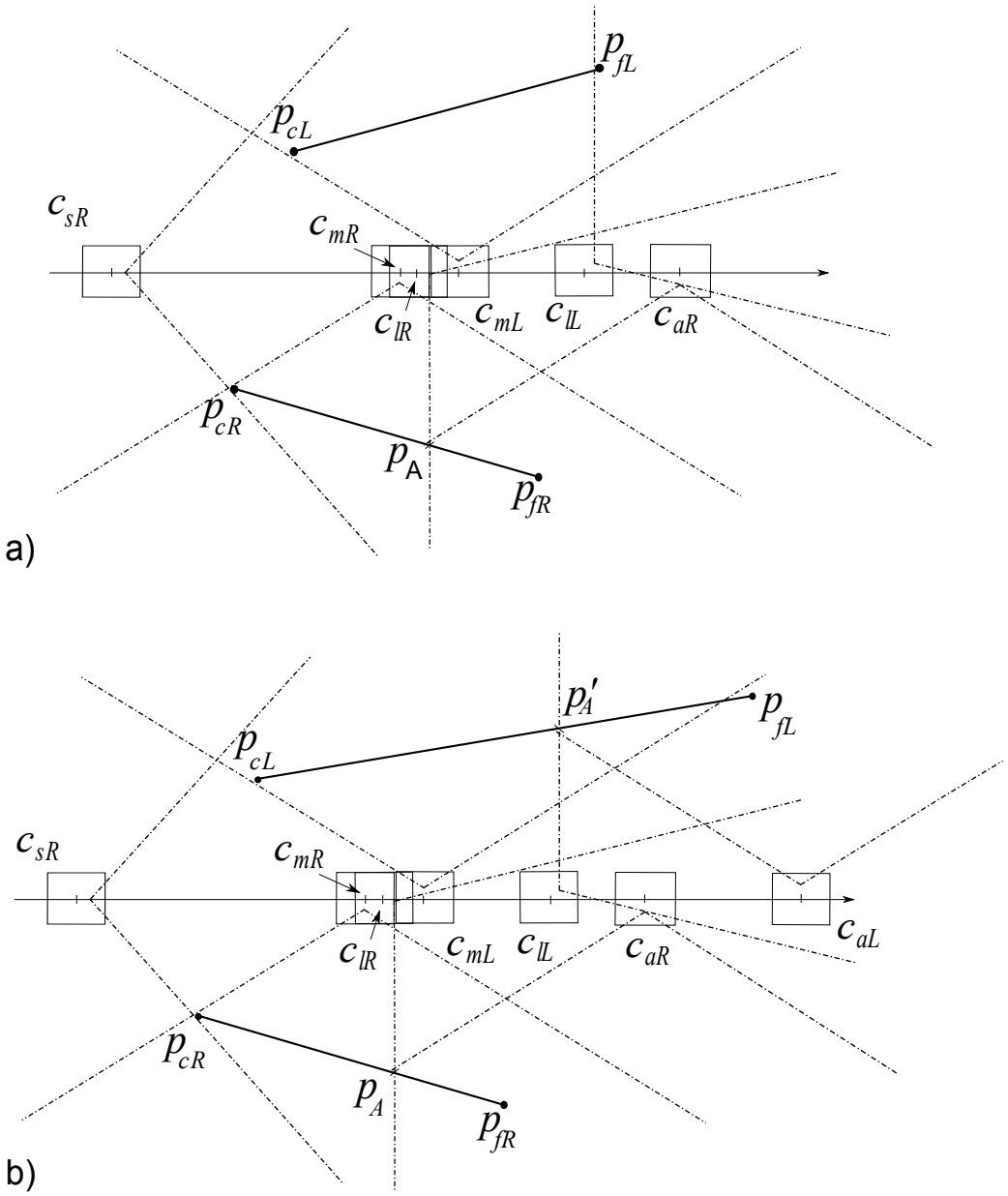
4.3 Implementace a testování

V této sekci popíšeme jednotlivé části od kalibrace stereokamery až po plánování pohybu robota.

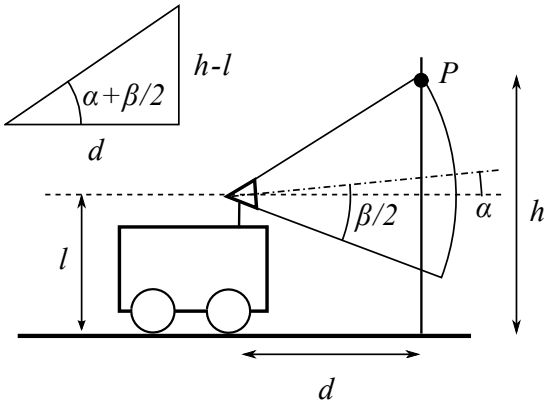
4.3.1 Kalibrace stereokamery

Pokud chceme vytvářet mapy pomocí stereokamer, prvním krokem musí být kalibrace stereokamery. Detailní popis kalibrace je v [11] a také v [37], odkud pocházejí i naše stereokamery VidereDesign.

Kalibrací dostaneme nezdeformovaný obraz, ve kterém budou mít odpovídající si body stejnou y souřadnici, ale hlavně získáme řadu parametrů nutných pro přesný výpočet polohy bodu.



Obrázek 4.13: Příklad robota, snažícího se prozkoumat prostor na obou svých stranách. Robot se opět pohybuje vpravo ve směru šipky. Nejdříve se otáčí smerem doprava tak, aby měl v zorném poli bod p_{cR}, v bodě c_{mR} se začne otáčet vlevo. V bodě c_{IR} si na hraně spojující pravé body udělá značku p_A udávající hranici prozkoumaného prostoru. Pak pokračuje v otáčení vlevo tak, aby v bodě c_{mL} byl na maximálním natočení kamer a přitom viděl bod p_{cL} krajem svého zorného pole. Ihned se začne otáčet zpět vpravo tak, že v bodě c_{IL} vidí periferně bod p_{fL} a v bodě c_{aR} opět při maximálním natočení vidí periferně bod p_A. Příklad na obr. b) je obdobný, s tím, že robot ještě musí prohlédnout zbytek hrany na levé straně.



Obrázek 4.14: Schéma výpočtu pohybu kamery ve svislém směru. α je zde úhel, o který se kamera otočí nahoru oproti vodorovné ose.

Důležité matice, které získáme, jsou matice vzájemné rotace a translace kamer. Matice rotace pro stereokameru VidereDesign je

$$R = \begin{bmatrix} 0.9999 & 1.5803e-003 & 5.9135e-004 \\ -1.5777e-003 & 0.9999 & -4.3428e-003 \\ -5.9820e-004 & 4.3418e-003 & 0.9999 \end{bmatrix} \tag{4.17}$$

Dá se říci, že matice je jednotková, to znamená, že kamery vůči sobě nejsou otočeny. Matice translace je

$$T = \begin{bmatrix} -52.1764 & 0.0515 & -0.3253 \end{bmatrix} \tag{4.18}$$

První parametr zde představuje vzájemnou vzdálenost kamer, druhý je posunutí v y ose a třetí v z ose. Vidíme, že vzájemná vzdálenost kamer je 52cm, posunutí v osách y a z jsou zanedbatelná.

Další matice, které kalibrací dostaneme jsou vnitřní parametry kamer, tyto parametry odpovídají matici z rovnice (3.12). Hodnoty fs_x a fs_y jsou u všech hodnot stejné, to proto, že pixely v kamerách mají čtvercový tvar a ohnisková vzdálenost je stejná ve směru x i y. Dalšími parametry jsou hodnoty o_x a o_y, které udávají geometrický střed obrazu, tzn. v případě rozlišení 640×480 by to měly být hodnoty 320 a 240. Vzhledem k nepřesnostem při výrobě kamery jsou však tyto hodnoty odlišné.

$$M_l = \begin{bmatrix} 349.3057 & 0.0 & 311.5939 \\ 0.0 & 349.3057 & 226.7340 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} = \begin{bmatrix} fs_x & 0 & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.19}$$

$$M_r = \begin{bmatrix} 349.3057 & 0.0 & 329.2601 \\ 0.0 & 349.3057 & 245.7464 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{4.20}$$

Obrázek 4.15: Na levém obrázku je nezkalibrovaný obraz z kamery. Na pravém obrázku je obraz už zkalibrovaný. Vidíme, že soudkovité zkreslení je odstraněno, ale přišli jsme tím o část obrazu. Pokud tak zkalibrujeme sterokamery, přijdeme o část obrazu, kterou vidí levá i pravá kamera zároveň. Slepý bod se nám tak posune dále od kamer.

Dalšími kalibračními maticemi jsou matice zkreslení, popsané kapitole 3.2. V těch je pět parametrů. V tomto případě jsou první dva parametry radiální zkreslení, další dva jsou tangenciální zkreslení a poslední je opět parametr radiálního zkreslení.

$$D_r = \begin{bmatrix} -0.3275 & 0.1122 & 0.0 & 0.0 & -0.0169 \end{bmatrix} \tag{4.21}$$

$$D_l = \begin{bmatrix} -0.3340 & 0.1265 & 0.0 & 0.0 & -0.0234 \end{bmatrix} \tag{4.22}$$

Posledními maticemi, které potřebujeme, jsou koeficienty pro přepočet polohy pixelů. Tyto matice mají stejný rozměr, jako je rozměr obrázku.

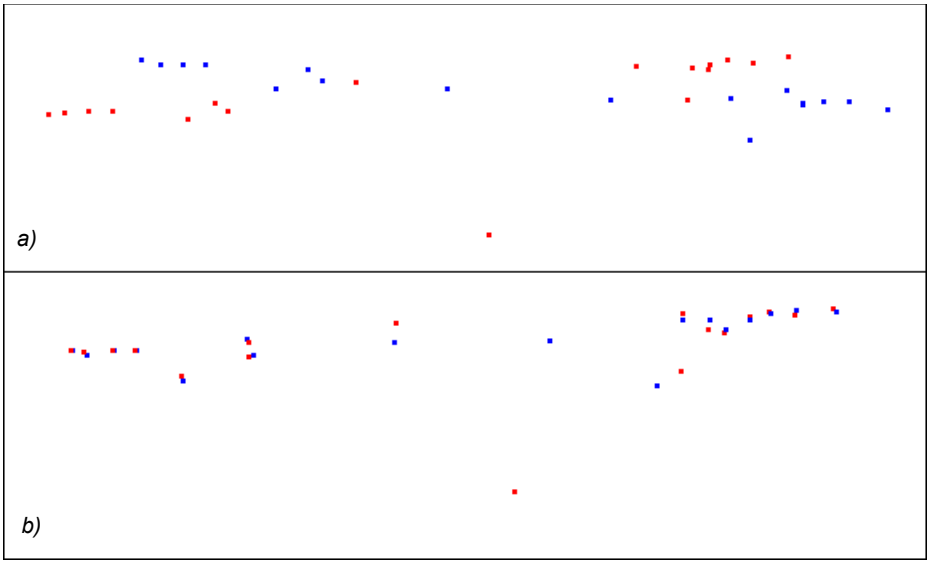
4.3.2 Výpočet polohy bodu

Výpočet polohy bodů se provede pomocí rovnic z kapitoly 3.5. Nejprve se ovšem musí nalézt odpovídající si body v obou snímcích. Význačné body se naleznou pomocí SURF algoritmu, pomocí jehož deskriptorů se také spárují význačné body. Vzhledem k tomu, že spárované body obsahují spoustu chybných korespondencí, je potřeba je nejprve opravit. To se provede např. tak, že se jako správné vezmou jen ty dvojice, jejichž y-souřadnice se liší max. o určitý počet pixelů. Na obr. 4.16 jsou špatně určené korespondence spojeny červenou čarou, ty správné pak modrou. Další případná oprava se provede po spočítání 3D souřadnic. Ty, jejichž z-ová souřadnice vyjde záporná nebo příliš velká, se odstraní z množiny 3D bodů. I přes tuto dvojí kontrolu se ale může stát, že se v množině detekovaných bodů objeví nějaké se špatnými 3D souřadnicemi.

4.3.3 Výpočet posunutí kamer

Jak už bylo zmíněno, posunutí kamer/robota lze určit několika způsoby. Může to být buď pomocí výpočtu fundamentální matice nebo pomocí posunnutí a rotace mračen bodů. Jako nejlepší byla nakonec vybrána metoda pomocí ICP (Iterative Closest Point). Její výpočet byl proveden pomocí knihovny MRPT [2]. Problémem těchto přístupů pro výpočet posunutí je šum, který zapříčiňuje drobné chyby ve výpočtu, které postupně zvyšují odchylku

Obrázek 4.16: Horní dvojice obrázků představuje pár snímků ze stereokamery. Červené body jsou všechny nalezené body SURF algoritmem, červené čáry spojují odpovídající si body. Modře jsou zobrazeny ty body, jejichž korespondence jsou vyhodnoceny jako správné. Svislé bílé čáry spojují ty modré body v novém (horním) snímku a starém (spodním) snímku, které byly vyhodnoceny jako stejné. To, že jsou čáry svislé značí, že kamera se mezi oběma snímky neposunula.



Obrázek 4.17: Na obrázku a) jsou dvě trochu posunutá a pootočená mračna bodů ve 3D prostoru. Na obrázku b) jsou tato dvě mračna již zarovnaná, takže jsou vidět odpovídající si body. U červeného bodu dole uprostřed byly špatně určeny jeho 3D souřadnice, a tak nekoresponduje se svým odpovídajícím si modrým bodem.

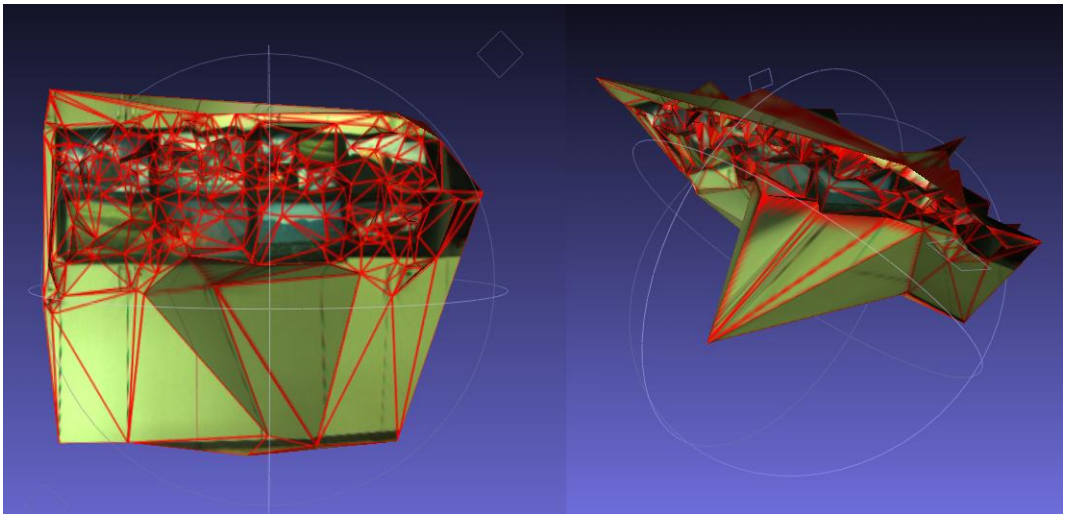
vypočítané polohy kamery od polohy skutečné. Tento problém by se dal vyřešit buď použitím algoritmu typu RANSAC nebo doplněním výpočtu o váhy jednotlivých bodů. Tak by bod, který má vyšší váhu, tzn. jeho polohou jsme si více jisti, přispěl k výpočtu posunutí více, než bod s malou váhou. Podobný přístup je použit např. v [51]. Ukázka zarovnání dvou stejných mračen bodů detekovaných ze dvou různých míst je na obr. 4.17.

4.3.4 Tvorba 3D modelu

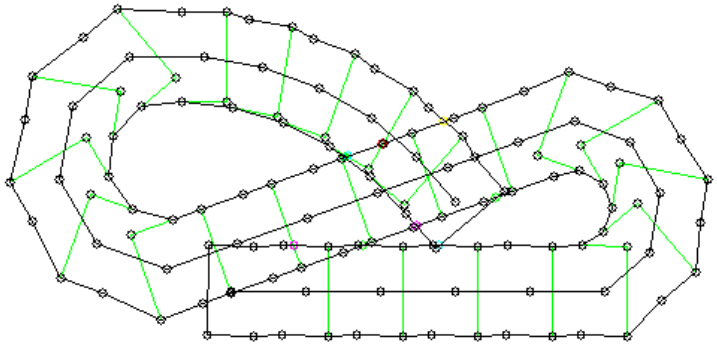
Tvorba 3D modelu z detekovaných bodů patří mezi složitější úkoly. Proto je v práci naznačeno jen vytvoření takového modelu z páru snímků ze stereokamery (obr. 4.18). To se provede pomocí Delaunayho triangulace. Lepší řešení je však v tomto případě použít váhovanou triangulaci (podkapitola 3.7.2), která počítá i s různou vzdáleností bodů od roviny kamery. Tím už v této 2D triangulaci dosáhneme takového vytvoření trojúhelníků, které budou splňovat Delaunayho podmínku i ve 3D prostoru, a které by se v případě použití běžné Delaunayho triangulace musely znovu přetriangulovat.

4.3.5 Vyznačování volného prostoru a plánování pohybu

Základem navržených plánovacích algoritmů je vyznačování volného prostoru a to především správné určení neodstranitelných hran, při pohybu robota vpřed. Ukázka z testování navrženého způsobu je na obr. 4.19. Poloha vyznačených průsečíků se spočítá pomocí algoritmů pro výpočet průsečíku dvou úseček. Zde jsme použili algoritmus uvedený v knize [25].



Obrázek 4.18: Na levém obrázku je model vytvořený z jedné dvojice snímků s naznačenou triangulací. Na pravém obrázku je týž model pootočený tak, aby na něm bylo vidět špatně vypočítané 3D souřadnice jednoho z bodů.



Obrázek 4.19: Na obrázku je ukázána dráha robota s barevně vyznačenými průsečíky se starou cestou. V těchto průsečících dojde k propojení černě vyznačené hranice volného prostoru. Zeleně jsou vyznačeny původní neodstranitelné hrany, které byly v důsledku robotova pohybu vpřed přeznačeny jako smazatelné. Důležité je, aby černě vyznačená hrana tvořila souvislou smyčku a to především v místech, kde robot zatáčí.

Kapitola 5

Závěr

Tato disertační práce představuje algoritmus pro výběr cílů a plánování cesty pro mobilní roboty vybavené pohyblivými stereokamerami.

5.1 Řešená problematika

Abychom robotům umožnili autonomně se pohybovat ve zvoleném prostředí, musejí si nejprve vytvořit mapu tohoto prostředí. V současnosti nejpoužívanějším algoritmem pro tvorbu map je tzv. simultánní lokalizace a mapování (SLAM). Tento obecný algoritmus umožňuje robotovi pomocí senzorů vytvořit mapu svého okolí a zároveň se v ní správně lokalizovat. Senzory, používanými roboty při této činnosti pro měření vzdáleností od překážek, byly nejprve sonary a postupně se začaly více používat laserové skenery a nyní jsou nejpoužívanější kamery. Těmi se detekují na překážkách význačné body, které jsou poté zaneseny do mapy. Mapy se tak skládají z těchto bodů a z hran mezi nimi, čímž se aproximuje tvar povrchu. Mohou také obsahovat texturu, která pomůže lépe rozpoznat jednotlivé části mapy v případě, že by ji měl použít i člověk.

Problémem při automatické tvorbě mapy je rozhodování, kterým směrem by se měl robot vydat a kterou část prostředí zmapovat nejdříve. Na zvoleném způsobu mapování totiž závisí výsledná přesnost mapy a tudíž i následná schopnost robota ji používat.

5.2 Navržená řešení

V případě použití kamer nemůžeme, na rozdíl od sonaru nebo laserového skeneru, automaticky považovat prostor mezi robotem a detekovanou překážkou za průjezdný. V práci je proto navrhnut způsob vyznačování volného prostoru a jeho oddělení od ostatních částí mapy pomocí vynucené Delaunayho triangulace. Volný prostor bude vymezovat území v mapě, ve kterém může robot plánovat svůj pohyb. Zároveň to také bude místo, kde se robot může pohybovat, aniž by při tom musel naplánovanou cestu testovat na možné kolize s překážkami.

Dalším navrženým algoritmem je plánování cesty v tomto volném prostoru a způsob prozkoumávání objevených bodů. Robot totiž volný prostor vytváří především svým pohybem v něm, a tak navržený algoritmus zajišťuje to, aby se robot, pokud je to možné, pohyboval vždy trochu jinou cestou. Dále je zde, vzhledem k nutnosti správného fungování algoritmu SLAM uzavírat smyčky, navržen způsob, jakým robot v neznámém prostředí zmapuje a uzavře nejprve největší smyčku a až teprve poté bude mapovat ostatní prostor.

V poslední části je navržen takový způsob pohybu stereokamer, aby se během mapování minimalizoval rotační pohyb robota a místo toho se využilo umístění kamer na pohyblivé platformě. Kamery se v této části mapování zaměřují na hrany, u nichž je velká pravděpodobnost, že jejich koncové body neleží ve stejné rovině a mezi nimiž by tak mohl existovat další neprozkoumaný prostor.

5.3 Další možné směry výzkumu

Bezprostředně navazující prací by mělo být nahrazení pevně daného bezpečného okolí robota takovým okolím, které by dynamicky měnilo svoji velikost podle nejistoty polohy bodů, kolem kterých by se robot pohyboval. To by také umožnilo jemnější mapování hranic volného prostoru, protože robot by se mohl více přiblížit k bodům, jejichž polohou by si byl více jistý.

Další směry výzkumu v plánování akcí robota závisí především na jeho schopnosti porozumět prostředí, ve kterém se pohybuje. Tomu může napomoci např. označování nejen volného prostoru, ale i dalších částí mapy podle toho, co představují. Mohou tak být např. označeny části mapy jako stěny, okna, dveře a další. V případě, že by se v obraze identifikoval i nábytek a různé předměty, pomohlo by to robotově větší interakci s okolím.

Výhodné také bude umožnit mapování i dynamicky se měnícího prostředí. V tomto případě by se mapa neskládala jen z jedné souvislé části, ale každý jednotlivý předmět, identifikovaný jako pohyblivý, by byl modelován zvlášť.

Oba předchozí návrhy by umožnily naplánovat i akce, při kterých by robot mohl plnit složitější úkoly, než je tomu nyní. Těmito úkoly by byla např. manipulace s konkrétními předměty, přesun konkrétních věcí z místa na místo, atd.

Literatura

[1] *Knihovna CGAL (Computational Geometry Algorithms Library).* Dostupné na: <`http://www.cgal.org`>.

[2] AL., J.-L. B.-C. et. *Knihovna MRPT (Mobile Robot Programming Toolkit).* Dostupné na: <`http://www.mrpt.org`>.

[3] AMENTA, N., BERN, M. a KAMVYSSELIS, M. A new Voronoi-based Surface Reconstruction Algorithm. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques.* New York, NY, USA: ACM, 1998. S. 415–421. SIGGRAPH '98. ISBN 0-89791-999-8.

[4] AMENTA, N., CHOI, S. a KOLLURI, R. K. The Power Crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications.* New York, NY, USA: ACM, 2001. S. 249–266. ISBN 1-58113-366-9.

[5] BARFOOT, T. D. Online Visual Motion Estimation using FastSLAM with SIFT Features. In *International Conference on Intelligent Robots and Systems (IROS).* 2005.

[6] BAY, H., ESS, A., TUYTELAARS, T. et al. Speeded-Up Robust Features (SURF). *Comput. Vis. Image Underst.* June 2008, roč. 110. S. 346–359. ISSN 1077-3142.

[7] BERG, M. de, KREVELD, M. van, OVERMARS, M. et al. *Computational Geometry: Algorithms and Applications.* 2nd edition. [b.m.]: Springer-Verlag, 2000. 367 s.

[8] BONIN FONT, F., ORTIZ, A. a OLIVER, G. Visual Navigation for Mobile Robots: A Survey. *Journal of Intelligent Robotics Systems.* 2008, roč. 53. S. 263–296. ISSN 0921-0296.

[9] BORENSTEIN, J. a KOREN, Y. Histogramic In-Motion Mapping for Mobile Robot Obstacle Avoidance. *IEEE Transactions on Robotics and Automation.* 1991, roč. 7, č. 3. S. 535–539.

[10] BORENSTEIN, J., EVERETT, H. R. a FENG, L. *Navigating Mobile Robots: Systems and Techniques.* Natick, MA, USA: A. K. Peters, Ltd., 1996. ISBN 1568810660.

[11] BRADSKI, G. a KAEHLER, A. *Learning OpenCV.* [b.m.]: O'Reilly, 2008. ISBN 978-0-596-51613-0.

[12] CASANOVA, O. L., IAENG, M., ALFISSIMA, F. et al. Robot Position Tracking Using Kalman Filter. *World Congress on Engeneering.* 2008, roč. 2.

[13] Chew, P. L. Guaranteed-quality Mesh Generation for Curved Surfaces. In *Proceedings of the ninth annual symposium on Computational geometry*. New York, NY, USA: ACM, 1993. S. 274–280. ISBN 0-89791-582-8.

[14] Choset, H., Lynch, K. M., Hutchinson, S. et al. *Principles of Robot Motion*. [b.m.]: The MIT Press, 2005. ISBN 0-262-03327-5.

[15] Clipp, B., Raguram, R., Frahm, J.-M. et al. *A Mobile 3D City Reconstruction System*.

[16] Cuccuru, G., Gobbetti, E., Marton, F. et al. Fast Low-memory Streaming MLS Reconstruction of Point-sampled Surfaces. In *Proceedings of Graphics Interface 2009*. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2009. S. 15–22. ISBN 978-1-56881-470-4.

[17] Davidson, A. J. SLAM with a Single Camera. *SLAM/CML Workshop (ICRA)*. 2002.

[18] Davison, A. J. Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV)*. Washington, DC, USA: IEEE Computer Society, 2003. S. 1403–. ICCV '03, sv. 2. ISBN 0-7695-1950-4.

[19] Davison, A. J., Reid, I. D., Molton, N. D. et al. MonoSLAM: Real-time Single Camera SLAM. *IEEE Trans. Pattern Analysis and Machine Intelligence*. 2007, roč. 29. S. 1–16.

[20] Doucet, A., Freitas, N. de, Murphy, K. P. et al. Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000. S. 176–183. ISBN 1-55860-709-9.

[21] Durrant Whyte, H. a Bailey, T. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *IEEE Robotics and Automation Magazine*. 2006, roč. 2. S. 9.

[22] Elfes, A. Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer*. 1989, roč. 22, č. 6. S. 46–57.

[23] Elinas, P. a Little, J. J. σMCL: Monte-Carlo Localization for Mobile Robots with Stereo Vision. In *Proceedings of Robotics: Science and Systems*. Cambridge, USA: [b.n.], June 2005.

[24] Elinas, P., Sim, R. a Little, J. J. sigmaSLAM: Stereo Vision SLAM Using the Rao-Blackwellised Particle Filter and a Novel Mixture Proposal Distribution. *Proceedings 2006 IEEE International Conference on Robotics and Automation (ICRA)*. 2006. S. 1564–1570.

[25] Ericson, C. *Real-Time Collision Detection*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004. ISBN 1558607323.

[26] Fairfield, N., Kantor, G. a Wettergreen, D. Real-Time SLAM with Octree Evidence Grids for Exploration in Underwater Tunnels. *Journal of Field Robotics*. 2007.

[27] Ferrez, J.-A., Müller, D. a Liebling, T. M. Parallel Implementation of a Distinct Element Method for Granular Media Simulation on the Cray T3D. *Supercomputing Review*. 1996, č. 8. S. 4–7.

[28] Fox, D., Burgard, W., Dellaert, F. et al. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. 1999. S. 343–349.

[29] Fox, D., Burgard, W. a Thrun, S. Markov Localization for Mobile Robots in Dynamic Environments. *Journal of Artificial Intelligence Research*. 1999, roč. 11. S. 391–427.

[30] George, P.-L. a Borouchaki, H. Delaunay Triangulation and Meshing: Application to Finite Elements. 1998.

[31] Hilton, A. Scene Modelling from Sparse 3D Data. *Image Vision Comput.* 2005, roč. 23. S. 900–920. ISSN 0262-8856.

[32] Hoppe, H., DeRose, T., Duchamp, T. et al. Surface Reconstruction from Unorganized Points. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1992. S. 71–78. ISBN 0-89791-479-1.

[33] Horn, B. K. P. Closed-form Solution of Absolute Orientation using Unit Quaternions. *Journal of the Optical Society of America A*. 1987, roč. 4, č. 4. S. 629–642.

[34] Joho, D., Stachniss, C., Pfaff, P. et al. Autonomous Exploration for 3D Map Learning. In *Autonome Mobile Systeme (AMS)*. [b.m.]: Springer, 2007. S. 22–28.

[35] Kalman, R. E. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*. 1960, roč. 82, Series D. S. 35–45.

[36] Koch, R. 3D Surface Reconstruction from Stereoscopic Image Sequences. In *Proceedings of ICCV*. 1995. S. 109–114.

[37] Konolige, K. *Small Vision System Calibration*.

[38] LaValle, S. M. *PlanningAlgorithms*. [b.m.]: Cambridge University Press, 2006. ISBN 0-521-86205-1.

[39] Leonard, J. J. a Durrant Whyte, H. F. Mobile Robot Localization by Tracking Geometric Beacons. *IEEE Transactions on Robotics and Automation*. 1991, roč. 7, č. 3. S. 376–382.

[40] Leonard, J. J. a Durrant Whyte, H. F. Simultaneous Map Building and Localization for an Autonomous Mobile Robot. In *Proc. IEEE Int. Workshop on Intelligent Robots and Systems (IROS)*. 1991. S. 1442–1447.

[41] Leonard, J. J. a Durrant Whyte, H. F. Directed Sonar Navigation. In. [b.m.]: Kluwer Academic Press, 1992. ISBN 0-7923-9242-6.

[42] Liu, J. S., Chen, R. a Logvinenko, T. *A Theoretical Framework for Sequential Importance Sampling and Resampling.* New York: Springer Verlag. ISBN 0-387-95146-6.

[43] Lowe, D. G. Object Recognition from Local Scale-Invariant Features. In *Proceedings of the International Conference on Computer Vision - Volume 2.* Washington, DC, USA: IEEE Computer Society, 1999. S. 1150–. ICCV '99. ISBN 0-7695-0164-8.

[44] Ma, Y., Soatto, S., Kosecka, J. et al. *An Invitation to 3D Vision: From Images to Geometric Models.* [b.m.]: Springer Verlag, 2003. ISBN 978-0-387-00893-6.

[45] Manessis, A., Hilton, A., Palmer, P. et al. Reconstruction of Scene Models from Sparse 3D Structure. In *CVPR.* 2000. S. 2666–2673.

[46] Manseur, R. *Robot Modeling and Kinematics.* [b.m.]: Da Vinci Engeneering Press, 2006. ISBN 1-58450-851-5.

[47] Marton, Z. C., Rusu, R. B. a Beetz, M. On Fast Surface Reconstruction Methods for Large and Noisy Point Clouds. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation.* Piscataway, NJ, USA: IEEE Press, 2009. S. 2829–2834. ICRA'09. ISBN 978-1-4244-2788-8.

[48] Montemerlo, M., Thrun, S., Koller, D. et al. FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping That Provably Converges. *Proceedings of International Joint Conference on Artificial Intelligence.* 2003. S. 1151–1156.

[49] Montemerlo, M., Thrun, S., Koller, D. et al. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence.* [b.m.]: AAAI, 2002. S. 593–598.

[50] Moravec, H. a Elfes, A. High Resolution Maps from Wide Angle Sonar. In *Proceedings of International Conference on Robotics and Automation (ICRA).* 1985. S. 116–121.

[51] Moreno, F. A., Blanco, J. L. a Gonzales, J. Stereo Vision-specific Models for Particle Filter-based SLAM. *Robotics and Autonomous Systems.* 2009, roč. 57, č. 9. S. 955–970.

[52] Moutarliere, P. a Chatila, R. Stochastic Multisensory Data Fusion for Mobile Robot Location and Environment Modeling. In *5th International Symposium on Robotics Research.* Tokyo: [b.n.], 1989. S. 85–94.

[53] Moutarliere, P. a Chatila, R. An Experimental System for Incremental Environment Modeling by an Autonomous Mobile Robot. In *The First International Symposium on Experimental Robotics I.* London, UK: Springer-Verlag, 1990. S. 327–346. ISBN 3-540-52182-8.

[54] Murphy, K. a Russel, S. *Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks.* New York: Springer Verlag, 2001. S. 499–515. ISBN 0-387-95146-6.

[55] Murphy, R. R. *Introduction to AI Robotics.* [b.m.]: MIT Press, 2000. ISBN 0-262-13383-0.

[56] Nistér, D., Naroditsky, O. a Bergen, J. Visual Odometry. In. 2004. S. 652–659.

[57] Nowak, R. a Scott, C. *Kalman Filters.* Dostupné na: <http://cnx.org/content/m11438/latest/>.

[58] Okabe, A., Boots, B., Sugihara, K. et al. *Spatial Tesselations: Concepts and Applications of Voronoi Diagrams.* 2nd edition. [b.m.]: John Wiley, 2000. ISBN 0-471-98635-6.

[59] Overmars, M. H. a Švestka, P. *A Paradigm for Probabilistic Path Planning.* 1996.

[60] Paz, L. M., Pinies, P., Tardos, J. D. et al. Large-Scale 6-DOF SLAM With Stereo-in-Hand. *IEEE Transactions on Robotics.* 2008, roč. 24, č. 5. S. 946–957.

[61] Pollefeys, M., Koch, R., Vergauwen, M. et al. Metric 3D Surface Reconstruction from Uncalibrated Image Sequences. In *3D Structure from Multiple Images of Large Scale Environments. LNCS Series.* [b.m.]: Springer-Verlag, 1998. S. 138–153.

[62] Pollefeys, M., Nistér, D., Frahm, J.-M. et al. Detailed Real-Time Urban 3D Reconstruction from Video. *International Journal of Computer Vision.* 2008, roč. 78. S. 143–167. ISSN 0920-5691.

[63] Rekleitis, I. *A Particle Filter Tutorial for Mobile Robot Localization.* Montreal, Quebec, Canada, McGill University: Centre for Intelligent Machines, 2004.

[64] Rojas, R. *The Kalman Filter.*

[65] Rosell, J. a Iniguez, P. Path Planning using Harmonic Functions and Probabilistic Cell Decomposition. In *International Conference on Robotics and Automation (ICRA).* 2005. S. 1803–1808.

[66] Russell, S. a Norvig, P. *Artificial Intelligence: A Modern Approach.* 2nd edition. [b.m.]: Prentice-Hall, Englewood Cliffs, NJ, 2003. ISBN 0137903952.

[67] Salman, N. a Yvinec, M. Surface Reconstruction from Multi-View Stereo. *Lecture notes in computer science.* 2009.

[68] Salman, N. a Yvinec, M. Surface Reconstruction from Multi-View Stereo of Large-Scale Outdoor Scenes. *The International Journal of Virtual Reality.* 2010, roč. 9, č. 1. S. 19–26.

[69] Schiele, B. a Crowley, J. L. A Comparison of Position Estimation Techniques Using Occupancy Grids. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).* 1994. S. 1628–1634.

[70] Se, S., Lowe, D. a Little, J. Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks. 2002, roč. 21. S. 735–758.

[71] Siegwart, R. a Nourbakhsh, I. R. *Introduction to Autonomous Mobile Robots.* Scituate, MA, USA: Bradford Company, 2004. ISBN 026219502X.

[72] Sim, R., Elinas, P. a Griffin, M. Vision-based SLAM Using the Rao-Blackwellised Particle Filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics.* 2005.

[73] Sim, R., Elinas, P., Griffin, M. et al. Design and Analysis of a Framework for Real-time Vision-based SLAM Using Rao-Blackwellised Particle Filters. In *Proceedings of CRV.* 2006.

[74] Sim, R., Elinas, P. a Little, J. J. A Study of the Rao-Blackwellised Particle Filter for Efficient and Accurate Vision-Based SLAM. *International Journal on Computer Vision.* 2007. S. 303–318. ISSN 0920-5691.

[75] Sim, R. a Little, J. L. Autonomous Vision-based Exploration and Mapping Using Hybrid Maps and Rao-Blackwellised Particle Filters. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (2006).* [b.m.]: IEEE, 2006. S. 2082–2089.

[76] Simmons, R., Thrun, S., Athanassiou, C. et al. ODYSSEUS: An Autonomous Mobile Robot. *AI Magazine.* 1992. extended abstract.

[77] Simmons, R. a Koenig, S. Probabilistic Robot Navigation in Partially Observable Environments. In *Proceedings of IJCAI-95.* [b.m.]: IJCAI, Inc, 1995. S. 1080–1087.

[78] Simon, D. Kalman Filtering. *Embedded Systems Programming.* 2001, roč. 14, č. 11. S. 72–79.

[79] Skrzypczynski, P. Uncertainty Models of the Vision Sensors in Mobile Robot Positioning. *Journal of Applied Mathematics and Computer Science.* 2005, roč. 15, č. 1. S. 73–88.

[80] Smith, R. a Cheeseman, P. *On the Representation and Estimation of Spatial Uncertainty.*

[81] Smith, R., Self, M. a Cheeseman, P. Estimating Uncertain Spatial Relationships in Robotics. In *UAI.* 1986. S. 435–461.

[82] Sonka, M., Hlavac, V. a Boyle, R. *Image Processing, Analysis, and Machine Vision.* [b.m.]: Thomson-Engineering, 2007. ISBN 049508252X.

[83] Stachniss, C., Hähnel, D., Burgard, W. et al. On Actively Closing Loops in Grid-based FastSLAM. *Advanced Robotics.* 2005, roč. 19.

[84] Taylor, C. J. a Kriegman, D. J. Exploration Strategies for Mobile Robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA).* 1993. S. 248–253.

[85] Thrun, S. *Robotic Mapping: A Survey.*

[86] Thrun, S. Probabilistic Algorithms in Robotics. *AI Magazine*. 2000, roč. 21, č. 4. S. 93–109.

[87] Thrun, S., Burgard, W. a Fox, D. *Probabilistic Robotics*. [b.m.]: MIT Press, 2005. ISBN 0-262-201623.

[88] Thrun, S., Fox, D. a Burgard, W. Monte Carlo Localization with Mixture Proposal Distribution. In *Proc. 17th National Conf. on Artificial Intelligence (AAAI-2000). AAAI Press/The*. [b.m.]: MIT Press, 2000. S. 859–865.

[89] Thrun, S., Fox, D., Burgard, W. et al. Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence*. May 2001, roč. 128, 1-2. S. 99–141. ISSN 00043702.

[90] Thrun, S., Hähnel, D., Ferguson, D. et al. A System for Volumetric Robotic Mapping of Abandoned Mines. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2003. S. 4270–4275.

[91] Thrun, S., Thayer, S., Whittaker, W. et al. Autonomous Exploration and Mapping of Abandoned Mines. *IEEE Robotics and Automation Magazine*. 2004, roč. 11. S. 79–91.

[92] Šolc, F. a Žalud, L. *Robotika*. 2002.

[93] Štěpán, P. *Vnitřní reprezentace prostředí pro autonomní mobilní roboty*. 2001. Disertační práce.

[94] Welch, G. a Bishop, G. An Introduction to the Kalman Filter. *IEEE Transactions on Reliability*. 2001.

[95] WillowGarage. *Knihovna OpenCV*. Dostupné na: <`http://opencv.willowgarage.com/wiki/`>.

[96] Yamauchi, B. A frontier-based Approach for Autonomous Eexploration. In *Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*. Washington, DC, USA: IEEE Computer Society, 1997. S. 146–151. ISBN 0-8186-8138-1.

[97] Yamauchi, B. Frontier-based Exploration using Multiple Robots. In *Proceedings of the second international conference on Autonomous agents*. New York, NY, USA: ACM, 1998. S. 47–53. AGENTS '98. ISBN 0-89791-983-1.

Vybrané publikace autora

[1] Rozman, J. Metody plánování cesty robota. In *Workshop Vršov '06*. FEKT VUT, 2006. s. 149–152. ISBN 80-214-3247-0.

[2] Rozman, J. Mobile Autonomous Robot. In *Proceedings of the 13th Conference and Competition STUDENT EEICT*. FIT VUT, 2007. s. 478–482. ISBN 978-80-214-3410-3.

[3] Rozman, J. Grid-Based Map Making using Particle Filters. In *Proceedings of MOSIS '08*. MARQ, 2008. s. 186–192. ISBN 978-80-86840-40-6.

[4] Rozman, J. 2D and 3D Motion Model with Uncertainty. In *Proceedings of XXXIth International Autumn Colloquium Advanced Simulation of Systems*. MARQ, 2009. s. 37–42. ISBN 978-80-86840-47-5.

[5] Rozman, J. Grid-based map making using particle filters. *International Journal of Autonomic Computing*. 2009, roč. 1, č. 2. s. 211–221. ISSN 978-80-86840-47-5.

[6] Rozman, J. Incremental Creation of a 3D Map with a Stereocamera. In *Proceedings of the 10th International Conference on Intelligent Systems Design and Applications (ISDA '10)*. Cairo, EG: IEEE, 2010. s. 4. ISBN 978-1-4244-8135-4.

[7] Rozman, J. Sampling-Based Algorithms for the Motion Planning. In *Proceedings of the 16th International Scientific and Practical Conference MTT '10*. Tomsk, RU: IEEE, 2010. s. 110–112. ISBN 0-7803-8226-9.

[8] Rozman, J. Visualization of a 3D Textured Model for the World Modeling. In *Proceedings of CSE 2010 International Scientific Conference on Computer Science and Engineering*. Košice, SK: TU v Košiciach, 2010. s. 314–319. ISBN 978-80-8086-164-3.

[9] Rozman, J. a Zbořil, F. V. Exploration in the VisualSLAM. In *International Conference on Computer Modelling and Simulation*. v recenzním řízení.

[10] Rozman, J. a Zbořil, F. V. Path Planning and Traversable Area Marking for Stereo Vision-based 3D Map Building. In *International Conference on Computational Vision and Robotics*. v recenzním řízení.

[11] Rozman, J. a Zbořil, F. V. Potential Fields and their use in Robot Navigation. In *Proceedings of XXVIIth International Autumn Colloquium ASIS 2005*. MARQ, 2005. s. 221–224. ISBN 80-86840-16-6.

[12] Rozman, J. a Zbořil, F. V. Trilobot Mobile Robot and its Using in Education. In *Proceedings of MOSIS '05*. MARQ, 2005. s. 188–195. ISBN 80-86840-10-7.

[13] Rozman, J. a Zbořil, F. V. A Concept of a Robot for the Robotour Competition. In *Proceedings of the 7th EUROSIM Congress on Modelling and Simulation.* Praha, CZ: CVUT, 2010. s. 6. ISBN 0-7803-8226-9.

Příloha A

Struktura souborů .obj a .mtl

Abychom si vytvořenou mapu mohli i prohlédnout v běžném 3D programu, je potřeba strukturu uložit v nějakém už známém formátu. Z velkého množství používaných formátů byl vybrán formát .obj společnosti Wavefront především pro svou jednoduchost, která je ovšem pro naše potřeby zcela dostačující a také proto, že se jedná o textový formát.

Pokud bychom chtěli čistě drátěný model, bohatě by nám postačovalo použití pouze formátu .obj. Protože ale budeme chtít náš model doplnit i o texturu, je potřeba použít i souboru typu .mtl, ve kterém jsou uloženy informace o materiálu objektů ze souboru .obj a také cesta k textuře.

Struktura .obj souboru je následující: jako první je uveden odkaz na soubor .mtl. Pak následují řádky s jednotlivými 3D body (vertexy), každý bod je uveden na samostatném řádku, který začíná písmenem *v*. Pak následuje jméno textury (materiálu) a její souřadnice, opět každá na samostatném řádku uvozeném písmeny *vt*. Jako poslední jsou faces (uvozené písmenem *f*), což jsou v podstatě trojice bodů (indexy vertexů, textur a normál) tvořících trojúhelníky. Každý takový bod může být tvořen až třemi hodnotami - vertexem, souřadnicí textury a souřadnicí normály. V našem případě normály používat nebudeme, takže každý trojúhelník budou tvořit tři dvojice hodnot. Souřadnice vertexu a textury jsou navzájem odděleny lomítkem. Mezi poslední dvojicí lomítek by byl index normály. Jednoduchý příklad souboru .obj ukazuje následující výpis:

```
mtllib ./cube.mtl
v 3.249995 78.000008 -167.480835
...
v 0.000000 73.435272 -162.765915

usemtl material_0
vt 0.510417 0.833333
...
vt 0.500000 0.822917

f 1/1// 2/2// 3/3//
...
f 1/1// 4/4// 2/2//
```

V souboru .mtl jsou pro nás důležité jen tři věci. Je to jméno materiálu, které pak bude uvedené u souřadnic textury, nasvícení materiálu a jméno souboru s texturou. Kd je difůzní

světlo, jehož hodnotami R, G, B se bude násobit uvedená textura.

```
newmtl material_0
Kd 1 1 1
map_Kd rect.bmp
```

Oba dva typy souborů mají samozřejmě mnohem více parametrů, hlavně .obj, ty jsou ale pro naši práci nepodstatné, a tak se jimi zde nebudeme zabývat.