VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

NAVIGACE MOBILNÍHO ROBOTA

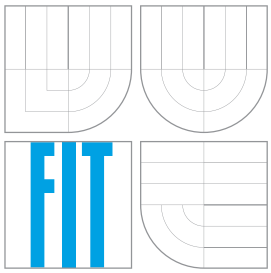
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. TOMÁŠ GOLDMANN

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

NAVIGACE MOBILNÍHO ROBOTA

MOBILE ROBOT NAVIGATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. TOMÁŠ GOLDMANN

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. FILIP ORSÁG, Ph.D.

BRNO 2015

Abstrakt

Když se podíváme do oboru robotiky zjistíme, že existuje několik typů robotů. Někteří z nich využívají pro svou činnost lokální a globální navigace. Cílem této práce je zmapování možností lokální navigace a popis základních používaných technik. Především se budeme zabývat algoritmy pracujícími s optickými senzory, jakými například mohou být kamery, stereokamery nebo laserové skenery prostředí. Praktická část této práce je zaměřená na návrh a implementaci algoritmu pracujícího s lokální navigací pro návrat robota zpět do výchozí pozice. Celá tato práce je spojená s pásovým robotem, který vznikl v rámci výzkumného projektu na Fakultě informačních technologií.

Abstract

When we look at the field of robotics we find that exist a lot of types of robots. Some of tham use location navigation and global navigation for their work. This work aims to map options of location navigation and description of basic technique which used. Especially, we will deal with algorithms which work with optical sensors, for example camera, stereo-camera or laser which scan medium. Practise section this work is focused on the proposal and implementacion algorithm which working with local navigation for robot's return to the starting position. All this work is connecting with tracked robot which formed in the framework one of project realization at Faculty of information technology.

Klíčová slova

robot, mobilní robot, lokální navigace, globální navigace, GPS, SLAM, počítačové vidění, Kalmanův filtr, Dijkstrův algoritmus, odometrie, filtry, ROS

Keywords

robot, mobile robot, locale naviation, global navigation, GPS, SLAM, computer vision, kalman filter, Dijkstra algorithm, odometry, filter, ROS

Citace

Tomáš Goldmann: Navigace mobilního robota, diplomová práce, Brno, FIT VUT v Brně, 2015

Navigace mobilního robota

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Filipa Orsága, Ph.D.

.....
Tomáš Goldmann
27. května 2015

Poděkování

Na tomto místě bych chtěl poděkovat všem, kteří mi jakýmkoliv způsobem pomohli s vypracování této práce. Především bych chtěl poděkovat těm, co mi pomohli při práci s robotem.

Obsah

Kapitola 1

Úvod

S rozvojem robotiky a elektroniky se čím dál více setkáváme s využitím robotů pro realizaci či ulehčení některých činností v mnoha odvětvích. V 21. století se roboti stali nedílnou součástí pokročilé průmyslové výroby. Ovšem využití robotů se začalo uplatňovat i v jiných oborech, například můžeme nalézt roboty v domácnostech, kteří autonomně provádějí vysávání, nebo se můžeme podívat do lékařského odvětví, kde roboti ulehčují či s kontrolou přímo provádějí některé zákroky, tak bychom mohli popsat několik dalších stránek textu o tom, kde se aktuálně uplatňuje obor robotika. Jelikož se neustálé rozšiřují technické možnosti a narůstá pole působností využití robotů, objevují se nové a nové úlohy, které je zapotřebí řešit nebo zdokonalovat. Dá se očekávat, že vrchol tohoto technologického rozmachu nebude v následujících letech dosažen, daleko pravděpodobnější se jeví zvyšování progrese vývoje.

Tato práce je zaměřená na roboty, kteří mají za cíl dostat se na dané místo a následně se vrátit zpátky do výchozího bodu. V ideálním, ovšem prakticky špatně realizovatelném případě, by se robot měl sám za pomocí robustní lokální navigace dostat na místo určení a následně se vrátit zpět do výchozího bodu. A proč je tento úkol špatně realizovatelný? Především proto, že do navigace a odometrie (viz 3.2.1) robota zasahuje mnoho aspektů, z hardwarového hlediska nedokonalost, senzorů jejich náchylnost k vnějším vlivům, a fyzikální limity. Ze softwarového hlediska pak nedokonalost a výpočetní náročnost používaných algoritmů, kterých jsou v autonomních robotech jednotky až desítky a tvoří velmi komplexní systém, podobný menším ekosystémům.

Tato úloha se dá teoreticky využít v robotech, kteří mají za úkol mapovat dané prostředí např. sutiny, spáleniště apod. Praktické využití algoritmu pro autonomní dosažení cílového bodu s možnosti návratu do výchozího je relativně omezené. Dá se ovšem využít jako dílčí část komplexnějšího algoritmu. Jistě by bylo pro hasiče přínosné, kdyby robot dokázal autonomně prozkoumávat oblast a následně se vrátit, přičemž jediným vstupem by bylo určení míst, kterými má projet.

Řešené úkoly v této práci jsou spojené s projektem s názvem: Robot pro hledání osob v závalech a lavinách. V rámci tohoto projektu vznikl robot, kterého v dalších částech této práce budeme nazývat Robotem záchranářem. Jedna se o zajímavý projekt, který byl realizovány Fakultou informačních technologií VUT. Celkovým cílem toho projektu je vývoj systémů pro prohledávání lavin a sutin. Na tento projekt může do budoucna navazovat mnoho dalších projektů. Například se může jedna o projekty spadajících do oblasti lokální navigace, proto je práce zaměřena na popis možností, které robot nabízí v oblastech systémů spojených s lokální navigaci. Z hlediska senzorů použitelných pro navigaci tento robot obsahuje ultrazvukové senzory, kamery, stereokameru, 2D laserový skener prostředí a GPS modul. Mozkem robota je výkonný počítač na kterém běží linuxový operační systém s aplikačním

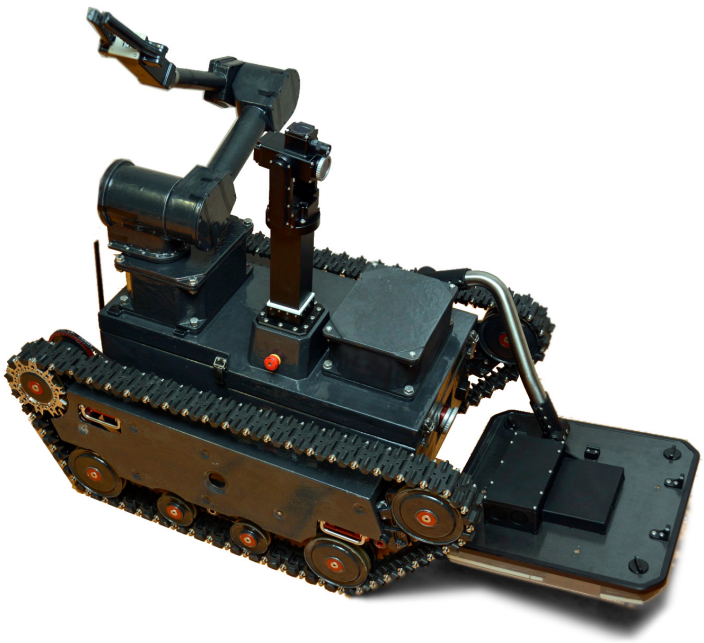
systémem pro řízení, který poskytuje mnoho prostředků pro realizaci různých algoritmů.

Hlavním cílem praktické části této práce je vytvořit algoritmus pro lokální navigaci, založený na datech z optických senzorů, s možností navrácení robota zpět do výchozího bodu. Výsledný algoritmus bude otestovaný na reálném robotu. Na základě získaných dat je možné vyhodnotit s jakou odchylkou robot dokáže kopírovat původní trajektorii. Pro realizaci těchto záležitostí plánuji využít dostupné algoritmy poskytované speciálním systémem pro roboty, který využívá Robot záchranář.

Práce je rozdělená do několika částí, přičemž první část je především zaměřena na obecné hodnocení možností lokální navigace a jejich výhod a nevýhod s důrazem na využití v praxi. V rámci první části si podrobně rozebereme vlastnosti Robota záchranáře. Druhá část práce bude zaměřena na návrh algoritmu pro zpětný návrat robota do výchozí pozice, s využitím již existujících algoritmů lokální navigace.

Při testování robota budu hodnotit úspěšnost návratu do výchozí polohy a celkovou funkci lokální navigace, která je na robotu k dispozici. Na základě těchto pozorování navrhnu další možnosti v oblasti lokální a možná i globální navigace.

Co je přínosem mé práce ? Prvním důležitým předpokládaným přínosem je rešerše možnosti lokální navigace s využitím optických senzoru pro novou robotickou platformu. Druhým potenciálním přínosem by mělo byt zhodnocení možnosti návratu robota z terénu. Ovšem více se dozvídáme na konci této práce, ve které se dostaneme od slova robot až k navigacím.



Obrázek 1.1: Robot záchranář.

Kapitola 2

Systémy v robotice a roboti

Robot je stroj, který vykonává jistou činnost dle daného algoritmu, aby tuto činnost mohl vykonávat potřebuje několik systémů. Základním předpokladem pro správnou činnost robota je získávání informací o svém okolí a možnost jeho ovlivňování, proto si na začátku této kapitoly rozebereme, jaké dva systémy se k této činnosti používají. Komplexnost těchto systémů se odvíjí od jednotlivých typů robotů. Řízení a součinnost systémů je zajištěná pomocí řídící elektroniky. Jakým způsobem je tato elektronika realizovaná záleží na architektuře daného typu robota, např. u Robota záchranáře je činnost řešená pomocí počítače a doplňující elektroniky. Z hlediska funkce jakéhokoliv systému například v robotu, je důležité zajištění zdroje elektrické energie, proto se okrajově podíváme i do této oblasti.

V další části této kapitoly se zaměříme na existující autonomní a poloautonomní roboty, kteří pracují s navigací. Roboti patřící do zmíněných kategorií, musí mít implementované algoritmy tak, aby se s využitím lokální navigace dokázali vyhnout překážkám a aby byly schopni zvolit nejvhodnější cestu ke globálnímu cíli. Ideální lokální navigace by měla zohledňovat jak mechanické aspekty robota, tak i aspekty prostředí, ve kterém se robot pohybuje. Například nemůžeme chtít, aby se klasický podvozkový robot pohyboval po vodní ploše, i když tato cesta neobsahuje žádnou překážku v prostoru. Dále se podrobněji seznámíme s Robotem záchranářem a zhodnotíme jeho možnosti ve vztahu k lokální navigaci, na kterou se zaměřuje tato práce.

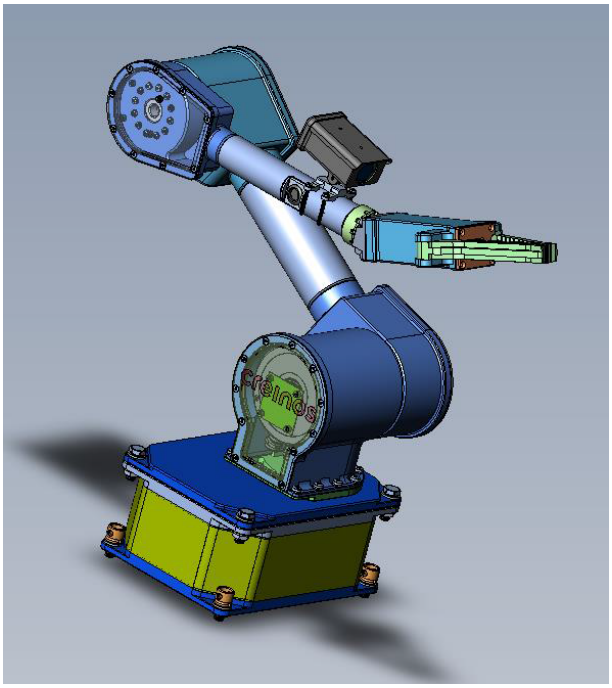
2.1 Systémy v robotice

Jak již bylo v úvodu této kapitoly uvedeno je v robotu nezbytné mít několik systémů. Pro získávání dat z prostředí, ve kterém se robot pohybuje, slouží senzorický systém. Ovšem robotů, kteří by pouze získávaly data a nemohly je následně použít k nějaké činnosti, je minimum. Proto dalším podstatným systémem jsou tzv. efektory. Zařízení takového systému zajišťují provádění interakce robota s okolím.

2.1.1 Efektory

Efektory slouží k provádění interakce robota s prostředím. Mezi typické efektory patří pohony, které zajišťují pohyb zařízení (kol, ramen, atd.). Tato zařízení zajišťují pohyb mechanických částí robota, prostřednictvím kterých dochází k již zmíněné interakci s okolím. Jak již byl napsáno v sekci příklady robotů, má Robot záchranář pohony pro pohyb pásů, výsuvného sloupku a robotického ramena. Ve všech případech se jedná o jeden z druhů efektoru. Efektory jsou ovšem z hlediska lokální navigace takřka nevyužitelné, proto se jimi

zde budeme zabývat jen okrajově. Na následujícím obrázku je ukázané manipulační rameno Robota záchranáře, jedná se o jeden z efektorů tohoto robota.



Obrázek 2.1: Vizualizace ramena Robota záchranáře [1].

Nejznámějšími efektory jsou elektrické motory. Tyto motory se nejčastěji dělí podle technologie, na které je založené otáčení. Výběr vhodného typu motoru je možné provést na základě požadovaného točivého momentu, velikosti a způsobu napájení.

Stejnosměrný motor
Tento typ motoru se skládá ze statoru, na kterém jsou umístěny hlavní póly s budícím vinutím, a pomocné póly umístěné mezi hlavními póly pro zlepšení komutačních vlastností. Další součásti tohoto motoru je rotor, který se otáčí v magnetickém poli. Konstrukce rotoru je provedená z plechů, v jehož drážkách je umístěné vinutí. Jednotlivé cívky jsou připojeny k lamelám, které nazýváme komutátor. Přívod elektrické energie je realizován pomocí kartáčů přiložených ke komutátoru [2].

Střídavý motor
Můžeme rozdělit do dvou kategorií na synchronní a asynchronií. V robotice nacházejí uplatnění synchronií motory. Jedná se o motor jehož hlavní vlastností je shoda otáček rotoru s magnetickým polem statoru. Střídavý proud ve vinutí statoru generuje magnetické točivé pole. Rotor může být z permanentního magnetu se střídavě uspořádanými póly. Další možností je, že má vinutí napájené ze stejnosměrného zdroje a tvoří elektromagnet [2].

Servomotor
Pro některé činnosti se používají tzv. servomotory, jedná se o stejnosměrný malý motor, jehož hřídel je vyvedená tak, aby se pohybovala v jisté kruhové výseči. Důležitou součástí serva je potenciometr, který slouží pro určení polohy výstupního kolečka. Princip je takový, že se roztočí motor a následně se pomocí operačního zesilovače porovnává s hodnotou poža-

dované polohy, jestliže je dosažená, tak se pohyb stejnosměrného motoru zastaví. Pokud by došlo vlivem působení vnější síly k narušení této rovnováhy, servo se bude snažit o opětovný posun do požadované polohy.

2.1.2 Senzory

Senzorický systém je dalším důležitým systémem používaným v robotu. Senzory jsou zařízení, která jsou schopné převádět jistou fyzikální veličinu na elektrický signál, který je následně možné prostřednictvím elektroniky vyhodnotit. Senzory můžeme rozdělit do několika skupin, jelikož Robot záchranář používá optické senzory k navigaci, bude tato kapitola zaměřená především na ně.

Abychom mohli lokální navigaci realizovat, je zapotřebí použít vhodných senzorů k vytváření mapy prostředí a určování polohy robota. Podíváme se na základní principy optických senzorů, které umožňují robotu „vidět“. Obecně můžeme říct, že z každého optického senzoru lze získat určité množství dat o prostředí, ve kterém se daný robot pohybuje. Nejpřesnějším a nejpřínosnějším senzorem pro vytváření 2D i 3D mapy prostředí je laserový skener. Naopak nejméně informací jsme schopni získat z obyčejné RGB kamery. Možnost používání optických senzorů v robotice přišla až s digitálními čipy, dřívější analogové optické senzory nebyly, až na výjimky různých infra závor, použitelné.

RGB a RGBD kamery

Vynalezením digitálního čipu, který dokáže zachytit světlo dané vlnové délky, došlo k prudkému rozvoji v mnoha odvětvích. Vstupem těchto senzorů je světlo, které se následně prostřednictvím čipu převede na elektrický signál. Princip je založený na základě fotoefektu, při kterém foton nárazem do atomů dokáže dostat elektron na vyšší energetickou hladinu. Na tomto základu vzniká na polovodičovém substrátu elektrický náboj, jehož velikost se odvíjí od množství dopadnutého světla. Aktuálně se používají dvě technologie a to CMOS (Complementary Metal–Oxide–Semiconductor) a CCD (Charge-coupled device). Ve spotřebitelském segmentu se nejčastěji používají z důvodu ceny CCD čipy.

CCD senzory jsou sestaveny z matice polovodičových prvků, které jsou schopné uchovat elektrický náboj. Po sejmutí obrazu se tento náboj správně načasovaným přiváděním náboje na elektrody začne posouvat směrem k posuvnému registru. V tomto registru je zachycen a poslán do zesilovače, na jehož výstupu se objevuje elektrický signál. Hlavním problémem u CCD čipu je tzv. *blooming*, jedná se o jev, že na čip dopadne tolik světla, při kterém se začnou ovlivňovat i sousední pixely [3].

CMOS senzor je vyráběn stejným způsobem jako procesory/mikroprocesory. Po zachycení obrazu se zachycený náboj ukládá do paměťové matice, ze které je možné číst podobným způsobem, jaký se používá pro čtení dat z RAM paměti. Jedná se o nákladnější technologii než v případě CCD.

Pokud nám nestačí získávat pouze barevný obraz, existují tzv. RGBD kamery, které přidávají senzor pro získávaní dat o hloubce prostoru. Takovýto senzor se skládá z projektoru a snímače. Projektor provádí promítání obrazu s danou strukturou v neviditelném světelném spektru. Snímač senzoru tento promítnutý obraz zachycuje, na základě jednotlivých části obrazu lze vytvořit hloubkovou mapu [4].

Lokální navigace

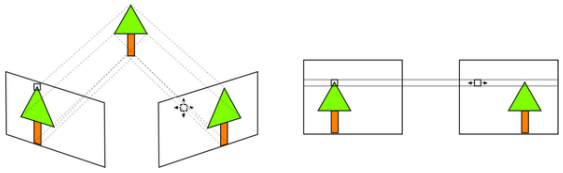
Lokální navigace pomocí RGB kamery může být založená na klasifikaci objektů a textur

v obraze. Pro popis obrazu se v robotice často používají klíčové body (SURF[1], SIFT[2]), 3D histogramy nebo histogramy rozložení gradientů. Mapování prostředí s využitím kamery je možné použít ve známém prostředí, kde máme dopředu informace o rozložení objektů. Tento typ navigace se využívá u některých akademických robotů a robotů určených pro soutěže. V případě, že používáme další senzory pro mapování prostředí, mohou data získaná touto kamerou být součásti fúze s daty z ostatních senzorů.

Výhody/Nevýhody
Z hlediska robotiky jsou hlavním nepřítelem těchto senzorů variabilní světelné podmínky, při kterých v obrazových datech vznikají přepaly a nebo tmavá místa. Tímto dochází ke ztížení správné interpretace informace. Tento neduh se dá částečně eliminovat pomocí filtru na objektivu kamery. Naopak výhoda těchto senzorů spočívá v jejich snadné dostupnosti a ceně.

Stereokamery

Při použití jedné kamery jsme mohli pracovat s obrazem jen ve 2D, při použití dvou kamer je možné provést 3D rekonstrukci prostoru, na který jsou jednotlivé kamery zaměřeny. Nutno však podotknou, že pro správnou funkci těchto kamer je zapotřebí dodržet správnou geometrií. Geometrie konstrukce stereokamery je inspirována biologickým modelem lidského vidění. Díky vzdálenosti mezi očima jsme totiž schopni odhadovat hloubku. Obě oči totiž vidí scénu podobným způsobem ovšem s jistým posunem, tento posun je nepřímo úměrný vzdálenosti mezi očima a vzdálenosti pozorovaného objektu. Často se ve stereo vidění setkáváme s pojmem disparita, jedná se o míru rozdílnosti snímku z jednotlivých kamer.



Obrázek 2.2: Znázornění principu výpočtu hloubkové mapy [5].

S rozvojem výkonnosti hardwaru se mohly začít v praxi používat stereokamery pro výpočet hloubkové mapy. Používané algoritmy pro získání hloubkové mapy budou podrobněji rozebrané v sekci Zpracování obrazu (4.2). Jak již bylo zmíněno v předchozí kapitole, hloubková mapa přináší z hlediska lokální navigace důležité data pro následné získání informací o prostředí, ve kterém se robot nachází.

Výhody/Nevýhody
Jelikož se ke konstrukci stereokamer používají RGB kamery, jsou jejich výhody a nevýhody totožné. K tomu se ovšem přidává další nevýhoda, kterou přináší algoritmy pro výpočet disparity. Není možné spočítat přesnou vzdálenost pro každý pixel v obraze. Především v porovnání s laserovým skenerem je výsledný poměr kvality získané hloubkové mapy vztažený k výpočetní náročnosti velice tristní.

[1] Speeded Up Robust Features - metoda pro popis obrázku pomocí deskriptoru, oproti SIFT rychlejší
[2] Scale-Invariant Feature Transform - metoda pro popis obrázku pomocí deskriptoru

Laserový skener

Jedná se o zařízení určené k provádění mapování prostředí. Princip mapování je založený na měření vzdálenosti k objektům pomocí světelného paprsku. Výsledkem této metody je množina bodů, které můžeme dále zpracovávat. Tato metoda se používá v mnoha odvětvích např. geografii, seismologii, robotice nebo kosmonautice. V této práci budeme slovem LIDAR označovat senzor používaný k této metodě průzkumu.

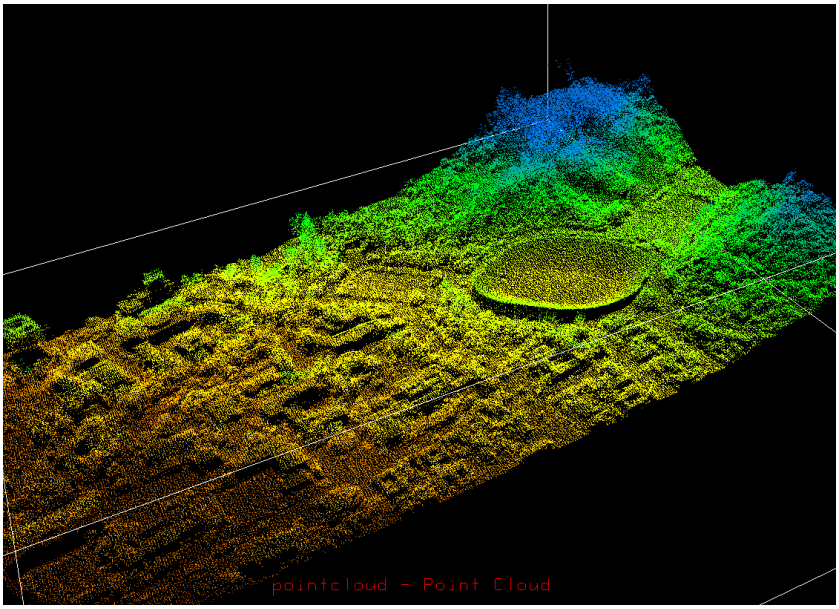
Vlnová délka používaného paprsku se liší podle toho v jakém prostředí se bude daný průzkum provádět. Pro vzdušné prostředí se používá paprsek o vlnové délce mezi 1040-1065 nm [6], v případě, že se průzkum provádí hloubkový průzkum např. ve vodě, použije se parsek o vlnové délce 532 nm, který proniká lépe vodou než paprsek větší vlnové délky. Měření vzdálenosti můžeme být založené na dvou principech, prvním z nich je měření na základě TOF (Time-Of-Fly), výpočet vzdálenosti se provede pomocí [7]:

$$d = \frac{c * t_f}{2} \tag{2.1}$$

kde:

c – rychlost světla,
t_f – doba letu vyslaného signálu,
d – vzdálenost.

Tento způsob se dá použít na vzdálenost několika kilometrů. Druhá možnost měření je založená na detekci změny fáze signálu. Výhodou této možnosti je především rychlost, ovšem za cenu striktnějšího omezení dosahu.



Obrázek 2.3: Mapa vytvořena z dat získaných pomocí laserového skeneru LIDAR [8].

2.1.3 Napájení

Mezí základní zdroje elektrické energie můžeme zařadit akumulátory a zdroje napájené z elektrické sítě. Akumulátory se dělí do dvou hlavních skupin, na primární a sekundární,

přičemž do první z nich patří baterie „na jedno“ použití, kdy po vybití se stávají nepoužitelné. Za to baterie druhé skupiny, sekundární akumulátor, se dají pomocí nabíjecího zařízení znovu dobít. Jelikož primární akumulátory se moc v robotice nepoužívají, podíváme se na základní typy sekundárních akumulátorů (dobíjecích baterií).

Olověné baterie
Olověné baterie jsou jedním z nejstarších zdrojů elektrické energie, Z pohledu robotiky je nevýhodou váha těchto baterií a jejich rozměrnost. Mezi jejich výhody patří cena a malý vnitřní odpor. Z těchto důvodů se používá v automobilech, ve kterých při startování auta potřebujeme krátkodobě mnoho energie. Nabíjejí se jednoduchou nabíječkou, pro nabiti stačí zdroj konstantního proudu. Nevýhodou je, že podléhají stárnutí.

Ni-Mh
Ze sekundárních akumulátorů můžeme Ni-MH baterie zařadit mezi starší typy. Jejich nevýhodou je paměťový efekt a rychlejší degenerace. Jedná se o poměrně tvrdý zdroj elektrické energie. Pro napájení robotů je tento typ nevhodný především z důvodu malé kapacity a velké velikosti.

Li-pol
Li-Pol je poměrně nový typ baterii, které mají jiné vlastnosti než předchozí uvedené typy. Hlavní výhodou je, že nemají paměťový efekt a jsou schopné uchovávat velké množství energie. Nevýhodou těchto baterii je náchylnost k mechanickému poškození, v extrémních případech může dojít i k explozi . V případě vybití těchto baterií pod určitou hodnotu dojde zároveň i k znehodnocení baterie. To samé platí i pro případ, že by došlo k přežití baterie. Baterie jsou vhodné pro pohony a zařízení s velkým odběrem proudu.

2.2 Příklady autonomních/poloautonomních robotů

Cílem této sekce je seznámení se s některými typy autonomních či poloautonomních robotů pracující s lokální navigaci. Mezi autonomní roboty řadíme i některé kvadrokoptéry nebo drony. V těchto robotech se používá především globální navigace se systémem GPS. Pomineme-li různé výzkumné projekty a akademické práce, zjistíme, že tito roboti takřka nevyužívají lokální navigaci. Z tohoto důvodu se zaměříme pouze na roboty pohybující se na zemi, kde je potřeba lokální navigaci řešit.

2.2.1 Robotický vysavač

V úvodu bylo zmíněno, že se rozšiřuje využití robotů v domácnostech. Nejčastěji používaným autonomním robotem v domácnostech je robotický vysavač. Jedná se obvykle o zařízení na kolovém podvozku s metacími kartáči a zásobníkem na odpad. Velikost robota je navržená tak, aby mohl efektivně projíždět po místnosti a dostával se i do hůře dostupných míst.

První generace těchto robotů byla založená na jednoduchém principu, kdy fungují tak, že robot narazí do překážky, otočí se, popojede a následně pokračuje v novém směru. V dalších generacích se algoritmy robota postupně vylepšovaly tak, aby bylo dosaženo optimálního vysání kolem zdi a v rozích. Některé firmy implementovaly algoritmy pro dynamické otáčení, robot se neotáčí stylem stop-rotate-run, ale udělá pohyb po půlkružnici.

Obrázek 2.4: Ukázka robotického vysavače [9].

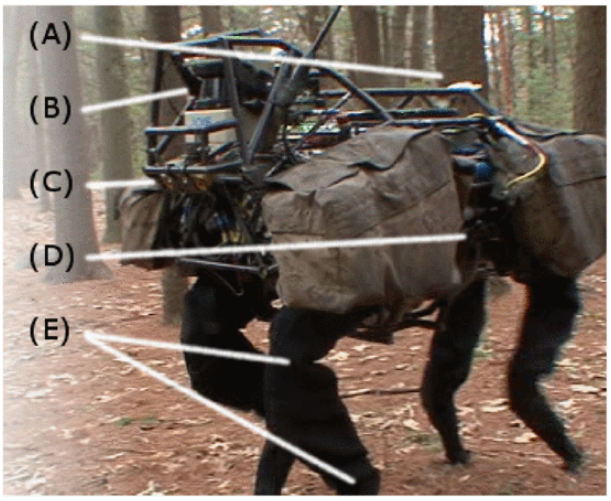
Výhody/Nevýhody
Hlavní výhoda především spočívá v usnadnění úklidu domácnosti, což je činnost ke které je tento typ robota určen. V případě, že je v místnosti více věci na podlaze, nemusí se robot dostat do všech zákoutí a ve výsledku bude potřeba úklid dokončit manuálně. V některých případech se robot může zaseknout nebo se zamotat do přípojných kabelů od elektronických zařízení.

Lokální navigace
Robot využívá jednoduchou lokální navigaci založenou na sonických a kontaktních senzorech. V některých případech roboti provádějí jisté mapování prostředí, aby mohli lépe optimalizovat cestu a nevracet se do stejných míst vícekrát. Někteří výrobci mají implementovanou možnost navrácení zpět do dokovací stanice. Princip funkce spočívá v zachycení IF paprsku a následném následování až do dokovací stanice.

2.2.2 BigDog - robot pro vojenské účely

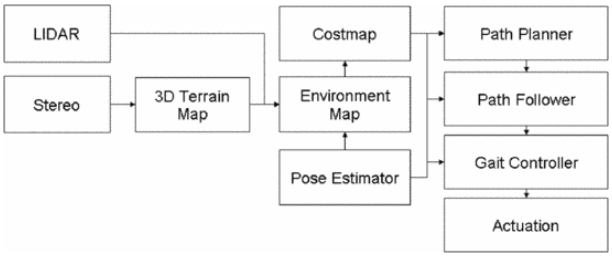
Společnost DARPA ve spolupráci se společností Boston Dynamics už několik let vyvíjí roboty pro vojenské účely. Z hlediska autonomních robotů v poslední době pokročil projekt BigDog. Cílem toho projektu je vyvinout čtyřnohého robota pro pohyb v náročném terénu. Vývoj tohoto robota pokročil do takové fáze, že byl využit i při vojenském cvičení. Robot má být určen k přenosu vybavení pro vojáky. Pohyb robota zajišťuje soustava hydraulických prvků, přičemž hydraulické ústrojí je spojené se spalovacím motorem [10]. Verze robota uvedeného níže na obrázku obsahuje dva počítače, první z nich je postavený na procesoru Intel Pentium M CPU (1.8 GHz) a slouží pro řízení stability a pohybu. Pro zpracování dat z vizuálních senzorů slouží druhý počítač, který je založený na procesoru Intel CoreDuo CPU (1.7 GHz) [11]. Nutno však podotknout, že v dalších verzích bude výpočetní vybavení odpovídat aktuálně dostupným prostředkům.

Výhody/Nevýhody
Hlavní výhoda robota spočívá v možnosti pohybu v horách, lesích a jiných složitých terénech. Nové generace toho typu robota mají pokročilé algoritmy stabilizace a technické možnosti pro navrácení do stabilní polohy v případě převrácení. Hlavní nevýhodou je hlučnost a omezení maximální vzdálenosti, kterou robot může urazit z důvodů omezeného množství paliva.



Obrázek 2.5: BigDog se znázorněnými senzory: a) GPS anténa, b) SICK LIDAR, c) Bumblebee setereo kamera, d) Honeywell IMU, e) Kloubové senzory [10].

Lokální navigace
Tento typ robota má velice robustní navigaci založenou na datech z LIDARu, stereokamer, gyroskopu a senzorech z nohou robota detekující zatížení jednotlivých noh, kontakt se zemí a jejich pozici. Algoritmy lokální navigace jsou schopné reagovat i na převrácení. Při testování v terénu s vojáky, měl jeden z vojáků radiolokátor a robot se vydával za tímto bodem. Pro plánování cesty se používá dynamický plánovač, protože vytvářená mapa má omezený dosah. Robot při své činnosti vytváří 2D cenovou mapu. Další vývoj se bude upírat na vytvoření 3D cenové mapy prostředí [11].



Obrázek 2.6: Architektura systému pro lokální navigaci [10].

2.2.3 Google car

Projekt Google car je jeden z mediálně nejznámějších projektů mající za cíl realizovat plně automaticky řiditelné auto. V podstatě se jedná o autonomního robota, který je kontrolován prostřednictvím řídícího střediska. V případě, že by došlo k problému, auto by zastavilo a operátor by na dálku převzal řízení. Pro dosažení možnosti autonomního řízení musí být vozidlo vybavené automatickou převodovkou, různými a redundantními senzory, bezdrátovou sítí pro vzájemnou komunikaci komponentů v autě, navigací, prvky pro automatické řízení (brzdění, zatáčení, signalizací atd.), serverem a softwarem s vysokými standardy na spolehlivost [12].

Obrázek 2.7: Google car [13].

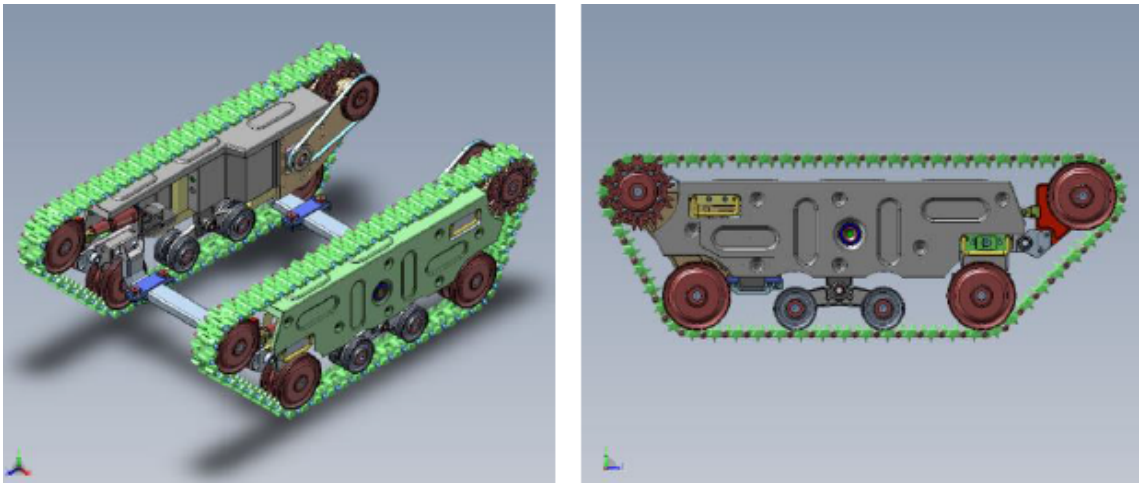
Výhody/Nevýhody
Výhody a nevýhody se mění s postupem vývoje, největší překážkou pro ostré nasazení těchto aut bude legislativa jednotlivých zemí a právní otázka v případě dopravních nehod. Cílem tohoto projektu je dosažení snadného a bezpečnějšího cestování. Dle některých informací ze sdělovacích prostředků, má tento typ auta problémy na kruhových objezdech, tuto informaci se ovšem nepodařilo podložit odborným článkem. Zřejmě se vycházelo ze skutečnosti, že při udělování licence pro provoz, byla vybrána trasa bez kruhových objezdů. Další nevýhodou je potenciální možnost hackerského útoku na vozidlo s cílem převzít jeho řízení nebo provést zásah do interních systémů. Takový útok by mohl mít fatální důsledky.

Lokální navigace
Google car má pokročilou lokální navigaci založenou na LIDARu, kameře (pro detekci objektů), předního zadního radaru a interní jednotky pro určování pozice. Lokální navigace musí být koordinována s algoritmy pro vyhodnocování dopravních situací např. semafory, přednosti před jinými vozidly apod. Mapování prostředí probíhá ve 3D [14].

2.2.4 Robot Záchranář

Projekt Robot záchranář je jedním z realizovaných projektů Fakulty informačních technologií VUT, cílem tohoto projektu byla konstrukce pásového robota pro práci v sutinách a lavinových závalech. Robot disponuje celou řadou senzorů od kamer až po bioradar určený k vyhledávání osob. Pohyb robota je založený na dvou nezávislých pásech, přičemž rozdílem rychlostí pásů se provede otočení robota do strany anebo pohyb po zakřivené trajektorii. Šasi robota je vyrobené z duralu opatřeného speciálním nátěrem, některé nosné části jsou provedené z oceli.

Řízení robota je založené na počítači s procesorem Core i7 (se sníženým navrženým tepelným výkonem) a 8 GB RAM, na kterém běží operační systém Linux (distribuce Ubuntu). Důvod použití tohoto systému spočívá v tom, že je pro něj vyvíjen aplikační systém pro roboty ROS (Robotic Operating System), kterému se ještě budeme v této práci věnovat v sekci 4.1. Interakci s prostředím zajišťuje série senzorů, jejíchž data jsou následně zpracována na výše uvedeném počítači. Mezi optické senzory, které na robotu můžeme najít, patří 2D laserový skener (LIDAR), přední a zadní kamera, stereokamera a náhledová otočná kamera. Robot dále obsahuje zadní a přední ultrasonický senzor tak, aby bylo možné jednoduše detekovat překážku před nebo za robotem. Za zmínku stojí, že tento systém je používaný u aut,



Obrázek 2.8: Podvozek robota s lavinovými pásy [1].

jako parkovací asistent. Robot v podstatě může fungovat ve třech režimech v závislosti na programu, který běží na ROS. V prvním režimu se může chovat autonomně, což je cílem softwaru implementovaného v této práci. V dalších dvou režimech do řízení může zasahovat operátor, v případě, že chceme robota ovládat manuálně, je řízení prováděno výhradně operátorem, pokud robot pracuje polo autonomně, operátor může provádět do jeho chování určité zásahy.

Z technického hlediska je tento robot poháněn pomocí elektromotorů. Jako zdroj elektrické energie se předpokládá využití napájecího kabelu, v případě odpojení od zdroje slouží jako nouzový pohon baterie Ping Battery 48V. Napájení elektroniky je realizované na samostatném okruhu, který je napájen baterií Thundersky 12 V a je od okruhu pro pohon robota galvanicky oddělen. Chlazení procesorů a grafiky je prováděno pomocí kapaliny, která je ochlazovaná v chladiči umístěného na korbě robota. Připojení jednotlivých senzorů je realizováno pomocí SATA, USB, UART a LAN [1].

Pro účely této práce budeme z Robota záchranáře pracovat s optickými senzory (stereokameru, kamery, LIDAR), na kterých založíme autonomní lokální navigaci s možností navrácení do výchozího bodu. LIDAR umístěný na tomto robotu snímá data ve 2D (v trajektorii jedné kružnice). Nevýhodou takového zařízení je, že může ignorovat překážky, které jsou umístěny pod nebo nad úrovní prováděného skenování.

Parametry kamer

Kamera	Rozlišení	Typ senzorů	Snímací režim
Přední a zadní kamera	640x480	CCD	snímání po řádku
Stereokamery	1360x1024	CMOS	celý snímek najednou

Vzdálenost středu kamer tvořící stereokamery je 190mm.

Parametry laserového měřícího systému

Robot obsahuje zařízení pro snímání vzdálenostních bodů v prostoru od firmy Sick, konkrétně verzi LMS100. Toto zařízení dokáže skenovat v kruhové výseči o rozsahu 270° s krokem $0,25°$. Maximální dosah měření se pohybuje do 20 m, přičemž v případě měření vzdálenosti k povrchům se špatnou odrazivosti je tato vzdálenost cca 18 m. Toto zařízení je k robotu připojené pomocí LAN rozhraní.

Výhody/Nevýhody
Robot poskytuje velké množství senzorů použitelných pro mapování prostředí, proto je možné vyzkoušet několik kombinací algoritmů pro lokální navigaci v neznámém prostředí a následně provést jejich vyhodnocení. Robot má robustní konstrukci, díky které by měl zvládat jízdu i ve složitém terénu, sutinách a závalech. V některých okolnostech je robot náchylný k převrácení například při jízdě z vysokých schodů.

Lokální navigace
Protože robot využívá systému ROS (viz 4.1) můžeme pro lokální i globální navigaci využít dostupných balíčku nebo si vytvořit vlastní v kombinaci s již existujícími. Robot ovšem nemá implementovaný balíček s možností zadání cílů a následným vrácením do výchozího bodu. Vytvoření takového balíčku je náplní této práce.

2.3 Shrnutí

V této kapitole jsme si shrnuli klíčové systémy používané v robotice, především pak efektory a senzory. Tato část kapitoly se především orientovala na optické senzory používané v robotice. Zaměřili jsme se na klasické RGB kamery, stereokamery pro odhad hloubkové mapy a na laserový skener LIDAR. Dalším důležitou částí v této kapitole je popis Robota záchranáře. S tímto robotem se budeme setkávat i na následujících stránkách této práce.

Kapitola 3

Navigace v robotice

Jestliže máme roboty, kteří využívají ke své činnosti pohyb, musíme při jejich realizaci řešit úlohy v několika oblastech. Navigace mobilních robotů je realizována propojením několika systémů, které mezi sebou intereagují. Množina systémů, které se budou využívat, závisí na tom, jakou činnost bude robot vykonávat. V případě poloautonomních robotů, pohybujících se v neznámém prostředí, se postupně dostaneme z oblasti realizace podvozku až do programové oblasti. Ta se obvykle dělí do několika vrstev, přičemž v nejnižších vrstvách řešíme ovládání pohonů a základní zpracování dat ze senzorů. Ve vyšších vrstvách propojujeme odometrii s řízením robota. Jestliže robot má realizované všechny tyto části, nastávají z hlediska uplatnění robota nejdůležitější otázky. Jakým způsobem se robot bude dostávat k cíli? Jak budou řešené situace, při kterých může docházet ke kolizím? Jakým způsobem bude robot řízen? Odpovědi na tyto a další podobné otázky jsou klíčové pro tvorbu aplikačního softwaru robota. V případě, že budeme mít robota, který bude plně řízený operátorem, nebudeme muset realizovat žádnou formu navigace, jelikož tato záležitost bude plně závislá na lidském faktoru. V našem případě ovšem máme poloautonomího robota, pro kterého je navigace klíčová. Software, který je vykonáván na tomto typu robota, má za úkol vyhodnocovat, zpracovávat data ze senzorů a poskytovat zpětnou vazbu nutnou pro jeho řízení.

Navigaci můžeme rozdělit do tří základních kategorií. První kategorii tvoří tzv. lokální navigace. Tento druh navigace určuje relativní pozici robota ke stacionárním nebo mobilním objektům v jeho okolí a umožňuje korektní interakci (nejen) s těmito objekty. Dalším typem je globální navigace, která určuje absolutní pozici robota v celosvětovém měřítku (případně i menším, dle potřeb) a dokáže ve své doméně navigovat robota do cílové pozice. Posledním typem je tzv. osobní navigace, která určuje polohu různých části robota ve vztahu k sobě sama. Protože máme mnoho druhů různých robotů, od nanorobotů po letadla, je zapotřebí správně zvolit fyzické měřítko navigace, které si můžeme definovat, jako míru přesnosti potřebné pro daný typ aplikace robota [15].

Pro lepší porozumění si uvedeme jeden příklad ze života. Pro horské turisty, kteří rádi dobývají náročné či méně obtížné horské cíle, je známy lavinový vyhledávač. Z pohledu robotiky se jedná o globální absolutní bod, ke kterému se má robot za úkol dostat. Stejným případem jsou i body určené pomocí systému GPS. Ačkoliv robot díky těmto bodům má přesně definovaný cíl, kam se dostat, tak mu chybí znalost o možné cestě k tomuto bodu tak, aby nenarazil do překážky, nezpůsobil škodu nebo sám sebe neohrozil. Tento problém řeší lokální navigace, která mapuje prostředí a vyhodnocuje s jistou pravděpodobností cestu robota za daným cílem.

3.1 Globální navigace

Globální navigace ma za cíl určovat polohu robota v globálním měřítku např. celosvětově a následně ho navigovat v rámci své domény. Když se na globální navigaci podíváme podrobněji, zjistíme, že pro potřeby globální navigace potřebujeme vymezit prostor, kde se za každé situace může teoreticky robot vyskytnou. Velice často se pro určení pevného globálního bodu, ke kterému se robot bude pohybovat, používá systém GPS, který je dostupný na každém místě země.

Obdobným systémem k GPS je ruský systém GLONASS, který funguje na stejném principu jako GPS. Evropská unie společně se soukromými firmami vyvíjí vlastní satelitní navigační systém Galileo. Administrativní sídlo tohoto systém se nachází v Praze [16].

3.1.1 GPS (Global Positoning System)

Systém GPS slouží pro určení absolutní polohy na zemi s určitou přesností. Za vývojem tohoto systému stojí Spojené státy americké, které tento systém původně navrhly ryze pro vojenské účely. Vývoj tohoto systému byl započat v roce 1973, v roce 1983 prezident Ronald Reagan rozhodl, že tento systém bude po dokončení dostupný i v civilním sektoru [17]. Určování polohy je založené na měření času letu signálu z jednotlivých družic k přijímači, který reprezentuje bod jehož polohu chceme zjistit.

Určování polohy se provádí pomocí matematické metody zvané triangulace, na základě které jsme, za předpokladu, že známe polohu jednotlivých družic a dobu letu signálu k přijímači, schopni určit polohu. Jednotlivé družice se sebou nesou atomové hodiny, jejichž cena a přesnost je oproti časovacím polovodičovým obvodům s krystalovým oscilátorem používaných v GPS přijímačích řádově vyšší. V případě, že by bylo použité nepřesné časování, došlo by k velké chybě měření a v případě, že bychom provedli několik triangulací postupně s daty ze čtyř družic, dostali bychom pravděpodobně rozdílné body. Proto se k třem družicím potřebným pro určení pozice přidává družice čtvrtá, která slouží pro určení časového ofsetu. Když signály dorazí ze všech družic (4) do přijímače, zpracují se a když víme, že byly vyslány ze všech družic ve stejný čas, můžeme vůči času z čtvrté družice vypočítat pomocí offsetu dobu letu a tím pádem vzdálenost k jednotlivých družicím [18].

Jednotlivé družice jsou umístěný na oběžné dráze, tudíž i přijímače přesně vědí kde jednotlivé družice jsou. Ovšem vlivem gravitace a „tlakem“ slunečního záření na satelity mohou vznikat tzv. efemeridové chyby. V případě, že s pozicí družice něco není v pořádku, provede se přidání informace o této chybě do vysílané zprávy, na základě které je možné provést při výpočtu korekce. V následující tabulce je uvedený přehled jednotlivých druhů chyb vznikající při přenosu signálu.

Zdroj chyby	**Potenciální chyba**	**Typická chyba**
Ionosféra	5m	0,4m
Troposféra	0,5m	0,2m
Efemeridová data	2,5m	0m
Posun času na satelitu	1,5m	0m
Vícecestný	0,6m	0,6m
Šum	0,3m	0,3m
Celkem	15m	10m

Tabulka 3.1: Přehled GPS chyb [18].

Nutno ovšem podotknout, že nejen tyto chyby se podílejí na nepřesném určení pozice. Další odchylky jsou způsobeny na straně přijímače, který se může nacházet v místě, kde dochází k slabému pokrytí signálem nebo odrazům signálu. Obecně platí, že čím více družic zachytíme, tím přesněji bude určená poloha.

Výše jsme si stručně popsali princip fungování a určování pomocí systém GPS. Nyní si popíšeme podrobněji jednotlivé segmenty tohoto systému. Dělení je následující:

- kosmický segment,
- řídící segment,
- uživatelský segment.

Jak již bylo uvedeno na začátku, klíčovým prvkem celého systému jsou družice, které řadíme do kosmického segmentu. Cílem kosmického segmentu je zajištění vysílání signálu GPS a příjem řídících signálu z operačního střediska. V rané koncepci se počítalo s využitím 24 družic pro možnost určení pozice na jakémkoliv místě na zemi, ovšem počet družic postupně narůstá, aktuálně se využívá 31 družic. Družice jsou umístěny na oběžné dráze ve výšce 20 350 km na 6 kruhových drahách. Každá z družic je velice komplexním zařízením skládajícím se z mnoha částí, pro zjednodušení uvedeme jen několik z nich. Pro určování přesného času se používají atomové hodiny nejčastěji s rubidiovým oscilátorem. Ve starších generacích družic se používal oscilátor cesiový. Na novějších družicích se používají atomové hodiny založené na vodíkovém maseru. Další důležitou částí je rádiový vysílač pracující na frekvenci v pásmu L (1000-2000 MHz) s anténami, které mají pravotočivou kruhovou polarizaci (RHCP). Napájení družice je zajištěno ze solárních panelů. S vývojem GPS systému vzniklo několik generací družic. Družice první generace vysílaly signál, který nebyl přístupný pro civilní účely. Jejich životnost byla naplánována na 4,5 let, ovšem reálná životnost dosáhla v průměru 8,76 let. V dnešní době již nepracují žádné družice první generace. Aktuálně se používají družice „typu“ blok 2 a připravují se družice typu blok 3 [19].

Cílem řídícího segmentu je monitorování vesmírného segmentu a řízení jednotlivých družic. V současnosti funguje na zemi několik středisek na různých kontinentech, hlavní středisko se nachází v Americkém městě Colorado Springs. Nejčastěji se v řídících střediscích provádí prostřednictvím příkazu na družicích korekce atomových hodin, predikce dráhy družice a sledování „zdravotního“ stavu. Celý systém je postavený tak, aby v případě, že budou zničena řídící střediska dokázal ještě nějakou dobu fungovat.

Z oboru robotiky je ovšem klíčový uživatelský segment, kdy pomocí GPS přijímače zachytíme signál a určujeme globální polohu vztaženou vůči mapě země. Pro civilní účely se využívá služby Standard Positioning Service (SPS), která je volně dostupná pro obyvatelstvo, co si zakoupí zařízení k příjmu GPS. Další poskytovanou službou je Precise Positioning Service (PPS), která je dostupná pouze uživatelům s dekódovacími klíči. Tato služba je využívaná v armádě (USA) pro podporu velení a navádění střel.

GPS moduly

Jak již víme, abychom mohli přijímat GPS signál potřebujeme GPS přijímač. Základem takového zařízení je GPS obvod, ke kterému se připojuje anténa. Pro řízení takového se obvykle používá mikrokontroléru nebo převodníků. Komunikace je obvykle realizována po sériové licence (RS232), po které se posílají zprávy definované protokolem NMEA. Tento protokol se původně používal pro sjednocení komunikace mezi zařízeními na lodi. Jedná se o poměrně jednoduchý protokol s kontrolním součtem bez kódování nebo komprese. NMEA

věta vyslána z GPS obvodu může vypadat například takto:

```
$GPGLL,4926.45,N,12311.12,W,225444,A,*1D
```

Obrázek 3.1: Ukázka GPS modulu a antény.

Pokud ovšem nepotřebujeme GPS integrovat do elektroniky, mohou nám vystačit různé moduly, které umožňují komunikaci na vyšší úrovni například přes GPS nebo LAN. Komunikační protokoly na takových vyšších úrovních nejsou nijak standardizovány a liší se výrobce od výrobce.

3.2 Lokální navigace

V úvodě jsme si definovali, co lokální navigace znamená, nyní se podíváme na její podstatné části. První nutnou části pro lokální navigaci je získání dat ze senzorů. Pro tento účel budeme využívat senzory, které jsou z fyzikálního hlediska schopné mapovat okolí robota. Pro účely lokální navigace se používají různé typy senzorů, nejčastěji však senzory kontaktní, sonické nebo optické. Každý z těchto senzorů poskytuje určitý druh dat. V případě kontaktních senzorů budeme pracovat s daty, ze kterých lze získat jednoduchou informaci, jestli robot narazil do překážky nebo ne. Dále máme senzory, které provádí měření, například ultrazvukové senzory měří vzdálenost k překážce. Z obecné podstaty měření víme, že mohou vznikat chyby (hrubá, soustavná nebo náhodná chyba). V případě, že takových senzorů máme více, získáme množství zašuměných signálů tzn. žádný z těchto signálů nebude přesně korespondovat s realitou. Pro agregaci nepřesných signálů z různých zdrojů se používají tzv. filtry.

Jestliže jsme úspěšně získali data ze senzorů (přímým měřením nebo fúzi) je zapotřebí provést jejich reprezentaci tak, aby bylo možné z nich získat potřebné informace. Velice často se využívá cenová mapa, která se vytváří absolutně k pozici robota. Aby bylo možné takovou mapu vytvářet, je zapotřebí mít dostatečně přesné senzory. V případě že by chybovost senzoru přesáhla určitou hranici, začala by se vytvářet nepřesná mapa a celá lokální navigace by začala vykazovat velkou chybovost. Hranice únosnosti chybovosti senzorů je závislá jak na fyzickém měřítku navigace, tak na požadovaném úkolu robota. Nad touto mapou je možné provádět výpočet cesty s nejmenší cenou, například pomocí Dijkstrova algoritmu.

Další možnou reprezentací prostředí jsou topologické mapy. Jedná se o grafovou reprezentaci, která zaznamenává místa na mapě v podobě vrcholů a jejich vzájemné vztahy pomocí

hran. Výhodou této metody je menší výpočetní náročnost cesty pro robota než v případě předchozí mapy.

3.2.1 Relativní měření polohy

Jedná se o proces odhadu polohy robota v daném čase na základě rychlosti, směru pohybu a předchozí známe polohy. Toto ovšem přináší jedno velké úskalí v podobě kumulativní chyby, kdy se chyba zvětšuje úměrně s počtem odhadu poloh. Pro realizaci takového odhadování se používají data z odometrie a interní měřící jednotky, která je založená na senzorech pro určení směru pohybu apod.

Odometrie

Nyní se podíváme na další, v robotice často používaný proces. Jen v této práci se s tímto možná pro laika neznámým pojmem setkáme několikrát. Odometrie je dalším z důležitých pojmů v oblasti robotiky. Jedná se o spojení dvou řeckých slov *hodos*(cestovat) a *metro*(měřit). Jedná se o proces, který řeší problematiku transformace dat z měřících senzorů na změnu pozice a orientace robota. Tento proces je důležitý, abychom mohli určit, jakým směrem se robot vydal a v nejlepším případě jakou urazil vzdálenost. Ovšem tato data jsou obvykle hodně nepřesná.

Prvním krokem tohoto procesu je získání dat z odmotrických senzorů. Jedná se o různé enkódery, které jsou úzce spojeny s efektory robota. Troufnu si tvrdit, že pro podstatu odmeotrie je naprosto klíčové propojení senzorického a efektorického systému. Jako příklad senzoru pro odometrii můžeme použít rotační enkóder, který poskytuje data o dokončených či nedokončených otáčkách motorů. Na základě rozměrů kola a této informace můžeme pomocí jednoduché matematiky vypočítat ujetou vzdálenost. Jak již jistě, tušíte odometrie je důležitou součástí takřka každých algoritmů pro lokální navigaci.

Možná se někomu může zdát, že odometrie je pouze spojená s pohybem robota v prostředí, ale není tomu tak. Odometrie se používá i s jinými efektory, například u manipulačního ramena potřebujeme vědět polohu jednotlivých motorů apod.

Interní měřící jednotka

Interní jednotka je dalším senzorem, který nám může pomoci v lokální navigaci. V literatuře se pro tuto jednotku používá zkratka IMU (Internal Messure Unit). Tato jednotka může obsahovat řadu senzorů. Mezi hlavní součástí můžeme zařadit akcelerometr nebo gyroskop. Jedná se o senzor zachycující zrychlení, jednotlivé typy těchto senzorů se liší především v hodnotě maximálního možného zrychlení a počtem os ve kterém se toto zrychlení měří. Jestliže máme senzor měřící zrychlení ve 3 osách, můžeme určit, kterým směrem se robot pohnul. A právě tato informace je důležitá pro lokální navigaci a doplňuje odometrii. Data z této jednotky se používají i pro SLAM, který si rozebereme v sekci 3.2.2.

Technologicky je akcelerometr vyroben pomocí technologie, která umožňuje integraci mechanických a elektronických struktur, tuto technologií nazýváme zkratkou MEMS (Micro-Electro-Mechanical Systems). Při výrobě těchto čipů dochází ke spojení technologie pro výrobu polovodičových čipů a technologie pro selektivní leptání nebo také implementování dalších vrstev [20]. Akcelerometr je pak realizován na základě principu nelineární závislosti kapacity C na vzdálenosti elektrod. Pokud jednu z elektrod uděláme pohyblivou a její pohyb bude závislý na zrychlení, dostaneme akcelerometr [21]. Jednotlivé druhy akcelerometru se

liší především v rozsahu měřeného zrychlení, citlivosti, reakční době, odolnosti proti přetížení a napájecím napětí.

Jednotlivé interní měřící jednotky se značně liší, jak množstvím senzorů tak i velikosti. Tyto jednotky se kromě robotiky využívají i v letectví, astronautice a v dalších mnoha odvětvích.

3.2.2 Lokalizace pomocí SLAM

SLAM (Simultaneous Localization And Mapping) je oblast, ve které se zabýváme vytvářením mapy v okolí robota a zároveň řešíme, kde se robot v této mapě nachází (lokalizuje). Tato lokalizace je klíčová z pohledu lokální navigace, jestliže nebudeme schopni v rámci nějaké odchylky určit polohu robota na mapě, nebude lokální navigace realizovatelná nebo bude vykazovat velkou chybovost, což by mohlo mít pro robota fatální důsledky. SLAM je úzce spojený s odometrií, pokud bychom na základě odometrických dat mohli provést přesné určení pozice robota, nemuseli bychom SLAM řešit. Ovšem odometrická data jsou velice nepřesná, například si představme, že se kolový robot pohybuje na šikmé plošině, která má povrch tvořený ledem a na druhé šikmé plošině, která má asfaltový povrch. Při stejném počtu otočení kol bude pravděpodobně ujetá vzdálenost rozdílná. Z tohoto důvodů, aktuální algoritmy realizující SLAM využívají pro určování polohy různá data. Velký význam se přikládá laserovým skenerům, které se vyznačují vysokou přesností a z jejichž dat se dá dobře odhadovat pozice robota a zároveň vytvářet mapa prostředí. Z hlediska informatiky je SLAM poměrně obecný pojem, definuje jakého cíle chceme dosáhnout a popisuje pomocí jakých metod ho dosáhneme. V některých publikacích je ovšem možné najít verze, kde jsou upřesněné použité filtry např. EKF SLAM. SLAM je možné použít jak pro 2D, tak i 3D pohyb.

Problém SLAMu můžeme popsat jako $p(x_t, m_t|z_{1:t}, u_{1:t})$, kde x_t značí polohu robota čase t, m_t je mapa orientační bodů v čase t, $z_{1:t}$ je série dat naměřených ze senzorů a $u_{1:t}$ je série řídících signálu.

SLAM zahrnuje několik částí, každá část může být realizována prostřednictvím rozsáhlé škály různých metod a algoritmů. Výběr algoritmů je závislý na prostředí a typu robota pro kterého SLAM řešíme. Základní části, které tvoří SLAM jsou následující [22]:

- extrakce orientačních bodů prostředí,
- asociace dat,
- odhad stavu,
- aktualizace stavu,
- aktualizace mapy prostředí.

Pro popis prostředí používáme tzv. orientační body, jedná se o charakteristické rysy v prostředí, které jsou odlišitelné a zároveň jsou snadno znovuobjevitelé. Protože jsou takové body základem pro sebelokalizaci robota stávající se klíčovým prvkem pro lokální navigaci. Objekt použitelný pro orientační bod by měl splňovat následující vlastnosti:

- Orientační body by měly být stacionární tzn. jejich poloha by v závislosti na čase měla být invariantní.
- Orientační body by měly být bohatě zastoupeny v daném prostředí.

- Orientační body různých typů by měli být od sebe snadno rozlišitelné.
- Orientační body by měl robot jednoduše znovu rozpoznat.

Extrakce orientačních bodů z prostředí se provádí na základě dat získaných ze vstupních senzorů robota. Pro určování polohy robota jsou vhodné senzory, které dokážou měřit vzdálenosti k daným bodům. V poslední době se rozšiřuje používání laserových skenerů. Pro extrakci orientačních bodů z prostředí s využitím laserového skeneru můžeme použít dva algoritmy nazývané Spikes a RANSAC [23]. Spikes algoritmus je určený pro na nalezení bodů, kde dochází k výraznějším rozdílu vzdálenosti. Druhý algoritmus (RANSAC) slouží k projektivní transformaci mezi obrazy, tento algoritmus využíváme, jestliže potřebujeme zjistit, jak se změnila poloha laserového skeneru.

Pokud jsme provedli extrakci bodů z prostředí, máme sice nové data, které potřebujeme zařadit do kontextu. Bude proto nutné provést asociaci nových dat s již existujícími daty. Prakticky ovšem budeme narážet na několik problému. Prvním z problému je, že robot může pracovat s orientačním bodem, který neexistuje v každém čase, takový orientační bod je ovšem špatný a cílem je takové body eliminovat. Dále se může stát, že robot vyhodnotí něco jako orientační bod, ale již tento bod nikdy nezaregistruje. Hlavním problémem, na který je zapotřebí myslet, když se snažíme navrhnout vhodný algoritmus je přiřazení orientačního bodu k orientačnímu bodu, který byl viděn v minulosti.

Další část je úzce spojená s filtry uvedené v předchozí sekci této práce. Cílem této části je odhadnout novou pozici robota a aktualizovat jeho polohu na mapě. Vstupními daty pro tento odhad, který se provádí pomocí variant Kalmanových filtrů, jsou data z odometrie, orientační body a předchozí pozice. Protože při extrakci mohly přibýt další orientační body, je zapotřebí je přenést do nové mapy.

3.3 Filtry

Filtry umožňují prostřednictvím modelů daného systému vypočítat ze zašuměných dat, co nejpřesnější hodnoty pro výstupní signál. Matematickým základem pro pravděpodobností filtry je Bayesův filtr [24] .

3.3.1 Bayesův filtr

Cílem tohoto filtru je vypočítat nejpravděpodobnější polohu robota v čase, přičemž využívá, jak aktuálně naměřených dat, tak i dat z minulých měření. Algoritmus se skládá ze dvou části, z předpovědi a aktualizace. Předpověď dané veličiny se provádí na základě modelu a ve fázi aktualizace se tato hodnota upřesňuje z naměřených hodnot. Podstatou filtru je odhadnout rozložení pravděpodobnosti přes stavový prostor v závislosti na datech ze senzorů. Protože se jedná o pravděpodobností filtr pracující se stavovým prostorem, modeluje se Bayesův filtr pomocí Markovových modelů.

Rovnice reprezentující Bayesův filtr je následující:

$$bel(x_t) = \eta P(z_t|x_t) \int P(x_t|u_t, x_{t-1}) bel(x_{x-1}) dx_{t-1} \tag{3.1}$$

Algoritmus 1: Pseudokód algoritmu Baysova filtru [24].

Vstup:
$bel(x_{t-1})$ - hodnota bayesova filtru v čase t-1
u_t - akce v čase t
z_t - měření v čase t
Výstup:
$bel(x_t)$ - hodnota bayesova filtru v čase t
for *all* x_t **do**
  $\overline{bel}(x_t) = \int P(x_t|u_t, x_{t-1})bel(x_{x-1})dx_{t-1}$
  $bel(x_t) = P(z_t|x_t)\overline{bel}(x_t)p(z_t)^{-1}$
end
return $bel(X_t)$

3.3.2 Kalmanův filtr

Kalmanův filtr, pojmenovaný po svém autorovi Rudolfovi E. Kalmanovi, je jeden z nejznámějších a nejčastěji používaných nástrojů při filtrování zašuměných dat [24]. I přestože jsou principy tohoto filtru přes 50 let známe, stále patří přímo on nebo jiné filtry odvozené od něho k důležitým prvkům moderní robotiky, dá se říct, že žádný mobilní robot se bez něj neobejde. Zjednodušeně můžeme říct, že Kalmanův filtr provádí fúzi dat ze dvou a více senzorů, přičemž tyto senzory poskytují signály v časové oblasti a Kalmanův filtr vytváří z těchto signálu jeden čistý signál. Pro příklad si představme dva senzory měřící vzdálenost (laserový, ultrazvukový), Kalmanův filtr na základě modelu provede transformaci těchto dvou signálu do jednoho.

Pro možnost využití Kalmanova filtru v systému musí být splněné čtyři podmínky. První z těchto podmínek je nutnost, aby modelovaný systém byl lineární. To znamená, že nový stav systému získáme vynásobením starého stavu matici přechodů. Dalšími podmínkami jsou, že šum měření musí být bílý (tzn. nekoreluje v čase) a že se tento šum musí dát modelovat Gaussovou funkcí. Poslední podmínkou je, že počáteční rozložení pravděpodobnosti systému musí být taky Gaussovo.

Model založený na Kalmanově filtru předpokládá, že se stav systému v čase t-1 vyvinul podle následující rovnice, dle publikace [25]:

$$x_t = F_t X_{t-1} + B_t u_t + w_t \tag{3.2}$$

$$z_t = H_t X_{t-1} + v_t \tag{3.3}$$

kde:

x_t – stavový vektor zahrnující například pozici a rychlost,
u_t – vektor zahrnující prvky potřebné k řízení,
F_t – matice, která převádí stav systému na měření,
H_t – matice, která převádí stav systému na měření,
B_t – řídicí matice aplikovaná na každý parametr ve vektoru, u_t
w_t, v_t – náhodné veličiny představující šum procesu.

Náhodné veličiny w_t, v_t mají normální rozložení pravděpodobnosti se středem v nule a s kovariančními maticemi Q a R [24].

$$p(w) \sim N(0, Q_t) \tag{3.4}$$

$$p(v) \sim N(0, R_t) \tag{3.5}$$

Algoritmicky se Kalmanův filtr skládá ze dvou z částí predikce a korekce, jedná se o totožné pojmy jako v případě Bayesova filtru.

Rovnice, které reprezentují část pro predikci jsou následující

$$\hat{X}_{t|t-1}{}^{1} = F_t \hat{X}_{t-1|t-1} + B_t u_t \tag{3.6}$$

$$P_{t|t-1} = F_t P_{t-1|t-1} F_t^T + Q_t \tag{3.7}$$

kde:

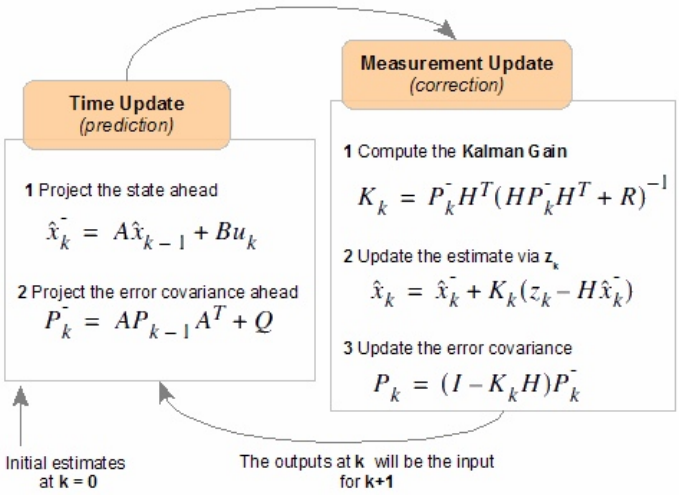
Q_t – kovarianční matice šumu.

Část která zajišťuje korekci a výpočet výsledné hodnoty Kalmannova filtru v čase t, je určená následujícími rovnicemi:

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - H_t \hat{x}_{t|t-1}) \tag{3.8}$$

$$P_{t|t} = P_{t|t-1} - K_t H_t P_{t|t-1} \tag{3.9}$$

$$K_t = P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + R_t)^{-1} \tag{3.10}$$



Obrázek 3.2: Rekurzivní schéma Kalmanova filtru [25].

[1] Stříška je použitá jako notace pro odhad.

3.3.3 Rozšířený Kalmanův filter (EKF)

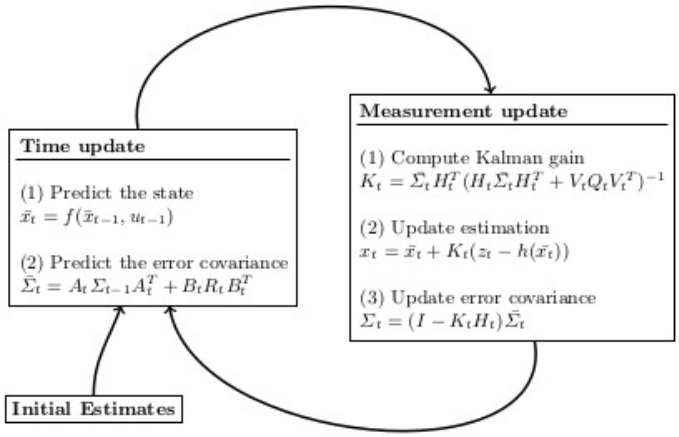
Málokdy můžeme systém popsat pomocí lineárních rovnic, v případě, že máme nelineární systém je zapotřebí použít rozšířený Kalmanův filtr EKF (Extended Kalmann Filter). Tento filtr pracuje se systémem, který je popsaný nelineárními stochastickými diferenciálními rovnicemi.

Rovnice ze základního Kalmannova filtru, které reprezentují model (zvýrazněny modře), budou nahrazeny dle následujícího schématu, na kterém je možné vidět, že dojde k nahrazení části, založené na linearitě, funkcí, která je závislá na řídícím vektoru a předchozím stavu. Stejné to je i v případě rovnice pro modelování měřící části Kalmannova filtru, podstatnou části této rovnice je nově funkce h, jejíž řídící proměnou je stavový vektor x_t.

$$x_t = F_t X_{t-1} + B_t u_t + w_t \Rightarrow x_t = g(u_t, x_{t-1}) + w_t \tag{3.11}$$

$$z_t = H_t x_t + v_t \Rightarrow z_t = h(x_t) + v_t \tag{3.12}$$

Předchozí rovnice ovšem řeší jen reprezentaci systému, při výpočtu hodnoty Kalmannova filtru se vychází z toho, že v případě lineárních systémů bylo možné využívat Gaussovo rozložení, ovšem u systému popsaných diferenciální rovnici Gaussovo rozložení přímo použít nemůžeme. Tato situace komplikuje algoritmus EKF, jelikož je zapotřebí provádět linearizaci [2] diferenciální funkce, použitá hodnota této funkce tudíž bude aproximována. Linearizace se může například provést pomocí prvního řádu Taylorova rozvoje.



Obrázek 3.3: Rekurzivní schéma rozšířeného Kalmanova filtru [26].

3.4 Plánování cesty

Dalším podstatou úlohou z hlediska navigace, ať už globální či lokální, je plánování cesty. Vstupme toho procesu je jistá reprezentace prostředí, která určuje, kde se potencionálně nacházejí překážky a volný prostor. Možnostem jakým může získat tyto informace jsem se

[2] Linearizace (někdy také lineární aproximace) je nahrazení části křivky (nebo průběhu funkce) přímkou.

věnovali v kapitole 3.2. Výstupem této úlohy bude reprezentace trajektorie k cíli s podmínkou, že je dosažitelný. Jednotlivé algoritmy, které jsou uvedené v následující sekcích dokáži vyhledávat nejkratší možnou cestu.

3.4.1 Dijkstra algoritmus

Jeden z nejznámějších algoritmů určených pro vyhledávání nejkratší cesty v grafu. Autorem toho algoritmu je nizozemský informatik Edsger Dijkstra, který tento algoritmus popsal v roce 1959 [27]. I přesto, že se jedná o půl století starý algoritmus, využívá se v mnoha oborech informatiky např. v sítích nebo geografických informačních systémech.

Algoritmus je založený na prohledávání grafu do šířky. Časová složitost závisí na implementaci algoritmu, v případě, že bude použité sekvenční prohledávání seznamu bude časová složitost O(n), v případě použití haldy bude tato složitost $O((m+n)ln(n))$, kde n je řád grafu a m velikost grafu [28].

Algoritmus 2: Dijkstrův algoritmus [28].

Vstup:
$G = (V, E)$ - orientovaný nebo neorientovaný graf, který nemá vlastní smyčky
s - značí počáteční uzel
Výstup:
Seznam D vzdálenosti k jednotlivým uzlům z s do v
Seznam P hran rodičů, např. P[v] je rodičem v
D:= []
D[s] := 0
P :=[]
Q :=V uložení uzlu do fronty
while *length(Q) > 0* **do**
  najdi $v \in Q$ jehož D[v] je minimální Q := odstraň(Q,v) **for** *každou* $u \in sousedic(v) \cap Q$ **do**
    if $D[u] > D[v] + w(vu)$ **then**
      $D[u] := D[v] + w(vu)$
      $P[u] := v$
    end
  end
end
return *D,P*

Protože algoritmus pracuje s grafovou reprezentací, budeme v případě použití mapy obsazenosti brát jako uzly jednotlivé body mřížky.

3.4.2 A* algoritmus

Jedná se o algoritmus založený na prohledávání stavového prostoru, opět tento algoritmus hledá nejkratší cestu v grafu. Samotný algoritmus je založený na algoritmu BFS (Best First Search), liší se ovšem v používané ohodnocovaní funkci.

Ohodnocovací funkce je definována jako $f(n) = g(n) + h(n)$, kde h(n) musí být spodním odhadem skutečné cesty od ohodnocovaného uzlu k cíli [29].

Algoritmus 3: Algoritmus A* [29].

Vstup:
Stavový prostor uzly
Výstup:
Cesta s nejlepším ohodnocením

OPEN:= [] uzly, které mají být prohledané
CLOSED := [] uzly, které už byly navštívené
while *obsahuje-li OPEN uzel* **do**
  vlož uzel do CLOSE **if** *uzel je cílovy* **then**
    return *cestu k počátečnímu uzlu*
  end
  Vybraný uzel expanduj a všechny jeho bezprostřední následníky ulož do OPEN, v případě, že se uzly vyskytují vícekrát, vyber nejlépe ohodnocené.
end
return *nebyl dosažen cílový uzel*

3.4.3 Algoritmus mravenčí kolonie

Oproti předchozím dvou algoritmům se jedná o méně rozšířený algoritmus. Algoritmus je inspirovaný biologickým modelem, kdy mravenci po cestě za potravou vylučují po cestě feromony, mravenci se následně vydávají po cestě, která má nejsilnější feromonový zápach. Experimentálně bylo dokázáno, že takto mravenci najednou nejkratší cestu. Tento algoritmus se používá jak pro optimalizaci cesty, tak i pro její plánování. Z pohledu plánování cesty v robotice existuje několik variant tohoto algoritmu, které se liší jak ve výpočtu pravděpodobnosti cesty, tak i v rychlosti zpracování.

Pravděpodobnost cesty mravence z místa i do místa j se vypočte dle následujícího vzorce [30]:

$$P_k(i,j) = \begin{cases} \frac{[\tau(i,j)]^\alpha [\eta(i,j)]^\beta}{\sum_{q \in povolen(i)} [\tau(i,q)]^\alpha [\eta(i,q)]^\beta} & j \in povolen(i) \\ 0 & jinak \end{cases} \tag{3.13}$$

kde:

$povoleno(i)$ – jsou dostupné uzly v bezprostředním okolí uzlu i,
α – parametr vlivu intenzity feromonu,
β – parametr vlivu vzájemné viditelnosti míst.

Protože se intenzita feromonů v čase t (popř. v každé iteraci) mění, je zapotřebí provádět aktualizaci hodnoty této intenzity

$$\tau(i,j) = \rho\tau(i,j) + \Delta\tau(i,j) \tag{3.14}$$

$$\Delta\tau(i,j) = \sum_{k=1}^{m} \Delta\tau_k(i,j) \tag{3.15}$$

$$\Delta\tau_k(i,j) = \begin{cases} Q/L_k & if (i,j) \in cesta(ant_k) \\ 0 & jinak \end{cases} \tag{3.16}$$

kde:

Q – konstanta,
L_k – délka cesty v dané iteraci.

Algoritmus 4: Pseudokód ACO algoritmu [30].

Vstup:
Matice reprezentující prostředí M - počet mravenců Startovní pozice
Výstup:
Inicializuj hodnotu feromonu pro každý z uzlu v mapě obsazenosti.
while *dokud nebylo dosaženo max. počtu iterací* **do**
    Umístí M mravenců na startovní uzel.
    Každý z mravenců si vybere další uzel v mapě z kterého je dosažitelný cílový uzel.
    Vypočítej délku použitelné cesty, kterou našel každý mravenec a pokud je to možné optimalizuj řešení.
    Aktualizuj hodnotu feromonu.
    Aktualizace ceny cesty.
end

3.4.4 Evoluční algoritmy

Vyhledávat cestu je taky možné pomocí evolučních algoritmů. Tyto algoritmy jsou inspirovány biologickou evolucí. Abychom mohli s evolučními algoritmy pracovat musíme vhodně zakódovat problém, v tomto případě budeme do chromozomu kódovat cestu z bodu A do bodu B. Dále si musíme vytvořit ohodnocovací funkci, která bude pracovat s mapou obsazenosti a cestou v chromozomu. Následně můžeme spustit algoritmus uvedený níže, pro který si definujeme populací (množství cest), maximální počet generací popř. si můžeme definovat typ křížení nebo mutace. Pomocí těchto algoritmu jsme schopni hledat jak existující cestu, tak i nejkratší cestu.

Algoritmus 5: Kostra evolučního algoritmu [31].

Vstup:
Vstupem může být jakákoliv struktura nad kterou se dá definovat cesta mezi dvěma body.
Výstup:
Nejlepší získané řešení.

Vygeneruj počáteční populaci.
Ohodnoť každého kandidáta z populace pomocí fitness funkce.
while *dokud nebude splněná ukončující podmínka* **do**
    Vyber vhodné jedince z populace a pomocí vhodných operátorů (křížení, mutace apod.) vytvoř jejich potomky.
    Z původní populace a z těchto potomků vytvoř novou populaci.
    Ohodnoť každého kandidáta z populace pomocí fitness funkce.
end
return *řešení s maximální fitness*

3.5 Shrnutí

Seznámili jsme se s typy navigací, které se používají nejen v robotice. Důležitou části této kapitoly je SLAM, který se používá pro lokální navigaci s vizuálními senzory. Algoritmy SLAM jsou zajímavé tím, že provádějí sebelokalizaci a mapování zároveň. Dále byly popsány základní filtry pro fúzi dat z různých senzorů, které své uplatnění nacházejí uplatnění i mimo robotiku, například v elektrotechnice. Mezi nejdůležitější filtry řadíme Kalmanův filtr.

Kapitola 4

Implementace

V předchozí části jsme se seznámili jak s robotickými systémy, tak i se základy technik a algoritmů používaných v lokální navigaci. Jak již bylo několikrát napsáno, jedná se o komplexní algoritmy, které prošly dlouhým vývojem. Implementace algoritmů je poměrně náročná a přesahuje rámec této práce. Naštěstí existuje systém pro roboty, který obsahuje množství balíků (implementovaných programů) pro realizaci navigace. Algoritmus pro návrat robota do výchozí pozice bude tyto balíky využívat. Návrat zpět bude řešen couváním robota. A to proto, že v praxi může nastat situace, kdy robot zajede do místa, kde nebude možné jeho otočení. Robot by měl při návratu zpět kopírovat původní trajektorii. Aby robot mohl provádět případné korekce při pohybu po původní trajektorii, byl implementován korektor založený na vizuálních datech.

4.1 Robot Operating System (ROS)

ROS je flexibilní framework používaný při vytváření softwaru pro roboty. Jedná se o kolekci nástrojů, knihoven a konvencí, které mají za cíl zjednodušit vytváření komplexního a robustního chování robota pro širokou škálu robotických platforem [32]. Troufnu si říct, že v oboru robotiky je ROS nejpoužívanější systém pro řízení komplexních robotů a to především z důvodů, že se kolem tohoto systému vytvořila široká komunita, ve které dochází ke sdílení ovládačů, či softwaru pracující s daným hardwarem na určité úrovni abstrakce. Příkladem může být ovladač pro zařízení LIDAR, jehož výstupem jsou Point Cloud (4.1.4) data. Pro vývojáře je tady toto výrazné usnadnění práce, jelikož člověk zaměřující se zpracování obrazu, v tomto případě spíš na práci s knihovnou PCL, nemusí řešit, jaké nízkoúrovňové data LIDAR posílá do počítače apod.

ROS už vyšel v několika verzích, jelikož prochází relativně rychlým vývojem, jsou mezi některými menší či větší rozdíly. Aktuální verze je *ROS Indigo Igloo*.

4.1.1 Konstrukce a konvence

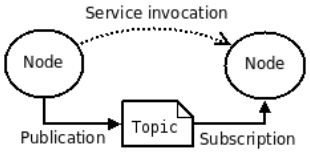
Abychom mohli lépe pochopit, jakým způsobem funguje ROS, zkusíme si v následující kapitole rozepsat základní konvence a používané konstrukce. ROS implementuje několik druhů komunikace, patří mezi ně například komunikace založená na vzdáleném volání procedur (RPC), asynchronní streaming dat pomocí ROS kanálů nebo ukládání dat na parametr server.

Mezi základní konstrukce používané v systému ROS patří uzly (*node*), ke každému uzlu se může vázat další používaná konstrukce, kterou je tzv. *topic*, tato konstrukce tvoří komuni-

kační kanál mezi dvěma uzly. Prostřednictvím tohoto kanálu si dva uzly mezi sebou vyměňují zprávy *message*. V některých případech se používá pro komunikaci volání služeb. Poskytující uzel nabízí danou službu pod názvem, následně uzel žádající zprávu zašle požadavek a uzel poskytující službu mu vyhoví. Tento způsob komunikace je podobný vzdálenému volání procedur RPC. Celý systém je pak organizován pomocí tzv. balíku (*packages*), každý takový balík obsahuje buď jeden uzel nebo několik uzlů, konfigurační soubory jednotlivých uzlů, knihovny a konfigurační soubor balíků. V tomto souboru je možné definovat výčet balíků na kterých je konfigurovaný balík závislý, definovat způsob překladu balíku, licenci a jiné, z hlediska funkčnosti méně podstatné, údaje. Pro překlad balíku se používají dvě nezávislé prostředí, první z nich *catkin* je založený na *CMake* makrech a *Python* skriptech, výhoda tohoto prostředí spočívá v lepší podpoře mezi jednotlivými platformami. Další možností je využití prostředí *rosbuild*, ovšem jeho podpora už je velice omezená a postupně všechny balíky přecházejí na *catkin*.

Další zajímavou konstrukcí resp. strukturou je *bag*. Jedná se o soubory obsahující množství zpráv, které zasílal nějaký uzel na daný *topic*. Následně je možné tyto zprávy přehrát a vysílat na zvolený *topic*. Využívá se to především pro testování algoritmů bez nutností fyzického zkoušení na hardwaru.

Základ systému ROS tvoří *roscore*, jedná se o základní uzly a kolekci programů, která zajišťuje fungování ostatních uzlů, komunikaci a mnoho dalších aspektů pro úspěšný chod celého systému.



Obrázek 4.1: Základní koncepce ROS znázorněná na dvou uzlech.

Nodlet
Jedná se o způsob jakým spustit několik algoritmů v jednom procesu, výhoda spočívá v tom, že nedochází k přenosu dat mezi uzly. *Nodlet* se tváří jako samostatný uzel i přestože je součástí jednoho uzlu. Jádro řídící jednotlivé uzly typu *nodlet*, postupně provádí dynamické volání tříd v daném uzlu.

Vizualizace zpráv
Pro vizualizaci se používá nástroj napsaný v programovacím jazyce *Python rviz*. Tento program dokáže vizualizovat data několika datových typů od proměnných po mapy prostředí. Navíc tento program dokáže vzdáleně komunikovat s ROS serverem, takže můžeme zkoumat data na ovládací stanici robota. Některé vizualizace z robota, co v této práci byly použité jsou získané vzdáleně.

4.1.2 Balíčky pro realizaci lokální navigaci

ROS obsahuje několik balíků pro vytvoření algoritmů lokální navigace, pojďme se podívat na některé z nich, které by bylo možné použít i v této práci.

octomap
Tento balík slouží pro vytvoření 3D mapy z optických senzorů, za výchozí data se považují data získaná z LIDAR laserových skenerů, ovšem je možné pracovat i s daty získa-

ných z hloubkových dat. Tato data ovšem bude zapotřebí převádět do formátu definovaného pomocí knihovny PCL. Vstupem do vstupního *topicu* je zpráva definována pomocí *PointCloud2*. 3D mapa je založená na stromové struktuře *octree*. Výhoda použití této struktury spočívá v menší paměťové náročnosti než při použití plného PCL modelu [33]. Pomocí dalšího balíku *Octomap server* můžeme provést převod této 3D mapy do 2D cenové mapy. Tato mapa může být následně použitá pro vytvoření SLAM algoritmu.

costmap_2D
Jedná se o další balík pro vytvoření ohodnocovaní mapy z dat získaných z prostředí reálného světa. Každá z buněk této mapy může nabývat 255 hodnot. Prakticky jsou ovšem důležité tří kategorie, buď je buňka obsazená, neobsazená a nebo jde o buňku neurčitou. Četnost aktualizace této mapy se nastavuje pomocí parametrů *update frequency*. Každý cyklus probíhá tak, že jsou přijatá data ze senzorů, následně je provedeno zpracování těchto dat a nakonec se tyto data promítnou do cenové mapy. Poslední krokem aktualizace je výpočet inflace v okolí překážek.

navfn
Funkce tohoto balíku spočívá v nalezení cesty s minimální cenou ze startovacího bodu do cílového bodu umístěného na cenové mapě. Výpočet cesty probíhá pomocí Dijsktrova algoritmu. V plánu je také implementovat výpočet pomocí A* heuristiky.

collider
Balíček pro slučování point cloud dat z více zdrojů do jednoho, výstupem je zpráva strukturovaná dle *PointCloud2* a zpráva strukturovaná dle *CollisionMap*. Tento balíček je ovšem experimentální a nebyl otestován na všech dostupných verzích.

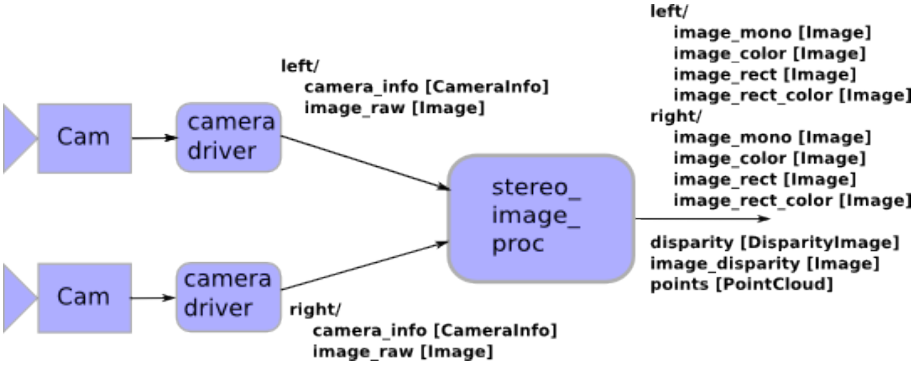
depthimage_to_laserscan
Jedná se o balík, který transformuje data z hloubkové mapy do zprávy určené pro data laserového skeneru (viz 4.1.4). Pro správnou funkci je zapotřebí nastavit parametry maximální a minimální hloubky, dále pak i rozsah skenování.

pointcloud_to_laserscan
Jedná se o balík, který transformuje data z formátu PCL (viz 4.1.4) do zprávy určené pro data.

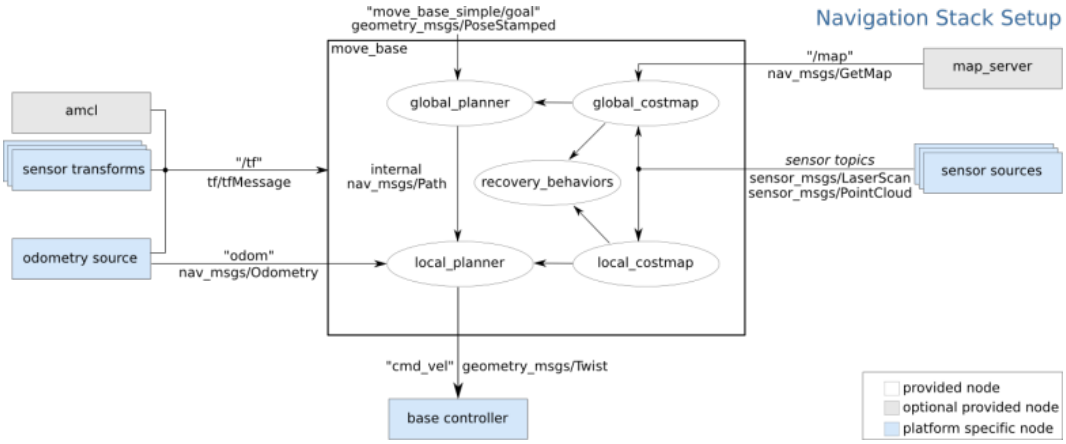
stereo image proc
Balík pro zpracování dat ze stereokamery jeho výstupem jsou snímky z jednotlivých kamer převedené do stupňů šedi, korelované snímky, celková disparita a korespondujících bodů.

Navigation stack
ROS dále obsahuje podsystém pro lokální navigací využívající data z LIDARu, který se nazývá *Navigation stack*. Na níže uvedeném obrázku je možné vidět jednotlivé části podsystému. Modře označené části je zapotřebí implementovat dle platformy robota, jedná se o uzly potřebné k ovládání robota, zisk dat z LIDARu a uzly pro transformaci dat.

Robotino
Balíček *Robotino* je postavený na předchozím balíčku. Balíček umožňuje autonomní navigaci robota k určenému bodu. Princip funkce je takový, že si nejdříve vytvoříme mapu prostředí pomocí řízeného pohybu robota. Následně takto získanou cenovou mapu použijeme k auto-



Obrázek 4.2: Vyobrazení rozhraní balíku *stereo_image_proc* [34].



Obrázek 4.3: Navigation stack [34].

nomní lokální navigaci, cílový bod zadáme do mapy prostředí pomocí programu *rviz*.

Lokální navigace řešená touto práci je založená primárně na datech ze tří optických senzorů tzn. kamery, stereokamery a laserového skeneru LIDAR. V následující kapitole jsou rozebrány základní metody používané v oboru počítačového vidění pro extrakci informací z obrazových dat. Tato data jsou klíčové pro vytvoření algoritmu pro lokální navigaci.

4.1.3 Balíčky pro SLAM

V sekci zabývající se lokální navigací jsme rozebrali základní principy metody SLAM. Tato metoda je pro lokální navigaci využívající optické senzory klíčová, proto se zaměříme v této části pouze na existující možnosti realizace SLAM metody v ROS.

gmapping

Tento balíček je založený na algoritmu gmapping z knihovy OpenSLAM [1], která sdružuje algoritmy pro realizaci SLAM. Algoritmus je založený na částicovém filtru Rao-Blackwellized, kde každá částice reprezentuje mapu prostředí. Filtr pracuje s adaptivními technikami pro snížení počtu částic. V predikci daného kroku se využívají jak data získaná z odometrie, tak i z aktuálního snímání prostředí. Výhodou tohoto řeší je redukce nejistoty určení po-

[1] OpenSLAM

zice robota. Vstupem pro tento balíček jsou data z laserového skeneru a odometrie robota. Primárním výstupem je mapa (ve 2D), která je tvořená mřížkou obsazenosti.

hector_slam
Jedná se o meta balíček, který spojuje několik další nutných pro realizaci SLAM. Pro mapování prostředí slouží balíček *hector_mapping*, jehož vstupem jsou data z laser skeneru. Na základě těchto dat se primárně vytváří mapa a zároveň určuje poloha robota. Na rozdíl od předchozího balíčku nevyžaduje výstup z odometrie. Další součásti *hector_slam* je balíček *hector_trajectory_server*, který ukládá trajektorii po které se robot pohyboval. Na závěr ještě zmíním jeden z hlavních balíčku s názvem *hector_geotiff*, který slouží pro ukládání mapa a trajektorie do obrázků.

rgbdslam
Tento balíček pro realizaci SLAMu vychází z algoritmu, který je uvedený v knihovně OpenSLAM. Algoritmus pro mapování je založený na odhadu změny pozice pomocí SIFT a SURF bodů. Vstupem pro tento algoritmus je obraz ze stereokamery nebo jiného RGBD senzoru. Výstupem je graf pozic s RGB body v prostoru resp. se jedná o prostorovou projekci založenou na 2D snímku a hloubkové mapě.

4.1.4 Reprezentace prostorových dat

V této práci se bude často setkávat s daty obsahujících množinu bodů v prostoru, proto se podíváme v jakých formátech se taková data uchovávají. V následujícím textu budeme zmíněná data nazývat prostorovými daty i přesto, že je tento název formálně nepřesný. Zaměříme se na formáty, které využívá systém ROS.

Podívejme se na proces zpracování dat z laserového skeneru. Surová data, které vycházejí z laserového skeneru nejsou ustanoveny žádnou normou a určuje si je každý výrobce sám. Většinou jsou data posílána pomocí síťového rozhraní formou UDP paketů. Tyto pakety obsahují hlavičku, data z měření a další informace např. čas nebo pozici GPS. Když se podíváme na to nejdůležitější část, data z měření, zjistíme, že obsahují naměřenou vzdálenost, vertikální úhel a v případě 3D měření i horizontální úhel.

Po přijetí takovýchto dat je třeba provést převod do struktur s kterými dokáže pracovat program, pro který jsou tato data určena, v našem případě se jedná o systém ROS.

Struktury knihovny PCL

Pro reprezentaci a práci s body v prostoru se používá knihovna PCL (Point Cloud Library). Tato knihovna zároveň definuje struktury pro ukládání bodů v prostoru. Z hlediska ROS systému jsou nejdůležitější struktury typu *PointXYZ* a *PointXYZI*. První z nich definuje bod v prostoru a druhá k této definici přidává i intenzitu bodu. Jednotlivé položky v této struktuře jsou datového typu *float*.

Výhodou použití tohoto formátu spočívá v široké škále možnosti zpracování těchto dat s využitím knihovny PCL. Tato knihovna podporuje i ukládání těchto dat do souboru.

Zpráva typu Laserscan

Jedná se o ROS zprávu, která má definovanou strukturu pro data získaná z laserového skeneru. Tento druh zprávy přijímají základní SLAM balíky uvedené v . Zpráva obsahuje data o úhlu počátku a konci skenování, úhel mezery mezi jednotlivými skeny, maximální

a minimální vzdálenost, čas mezi jednotlivými skeny, čas mezi měřením a nakonec pole dat reprezentujících vzdálenost. Z těchto informací se dá sestavit prostorová mapa prostředí a nebo vytvořit PCL data [32].

Hloubková mapa

Hloubková mapa prostředí je reprezentace prostoru pomocí intenzity barev v černobílém obrázku. Pro zpracování se jedená o jeden z nejhorší formátů prostoru. Většinou se hloubková mapa převádí na jeden z výše uvedených formátu. ROS pro takový převod nabízí minimálně dva balíky (viz 4.1.2).

4.2 Zpracování obrazu a algoritmy

Od dob, kdy vznikl první digitální snímač obrazu, bylo zapotřebí řešit problematiku následného zpracování dat. Jakmile obrazový snímač dokončí proces snímání, jsou data uložena do paměti, v případě snímacích čipu CMOS je paměť součást senzoru, zatímco v případě CCD je paměť mimo výkonný čip. Jelikož je fyzikálně a technologický nemožné získat data reprezentující dokonale snímaný vzor, musí se provést následné zpracování. Toto zpracování můžeme rozdělit do dvou části. V první části, která se odehrává částečně na hardwaru a částečně softwarově, je zapotřebí z obrazových dat odstranit šum a provést komprimaci dat. V případě běžné spotřební elektroniky se tento krok označuje slovem *postrpocesing*, protože se jedná o konec procesu získávání obrazu ze snímače.

V oboru počítačového vidění se však jedná o první krok dlouhého procesu získávání požadovaných informací z obrazu. Z pohledu počítačového vidění je předzpracování proces, při kterém se snažíme z obrazových dat odstranit nepotřebné informace např. vysokofrekvenční složky obrazu, přičemž chceme zachovat nebo extrahovat takové informace, které jsou potřebné pro další části procesu.

4.2.1 Gaussův filtr

Mezi nejčastěji používané filtry pro odstranění vysokofrekvenčních složek (šumu, malý objektů, atd.) slouží filtry vycházející z Gaussové funkce. V následující rovnici je uvedená modifikace vzorce pro obraz indexovaný pomocí písmen x a y [35]:

$$g(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\pi\sigma^2}} \tag{4.1}$$

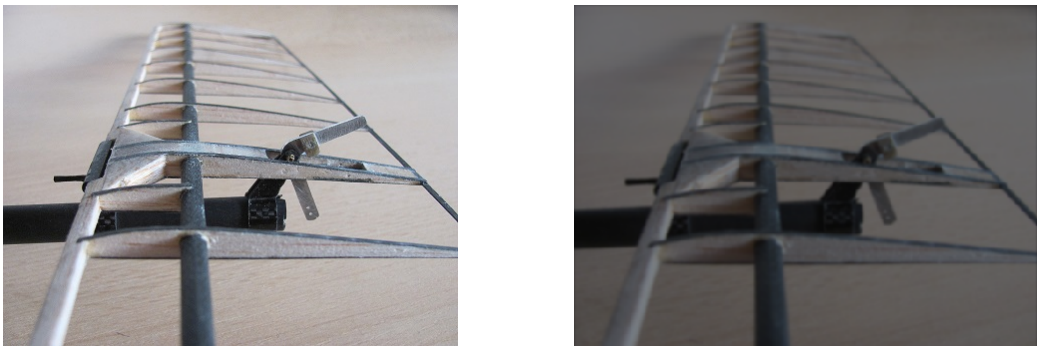
kde:

σ – určuje šířku „kopce“, x a y jsou souřadnice v obraze.

Tato funkce se používá pro výpočet koeficientů konvoluční matice (4.2), která se aplikuje na obraz. Z důvodů, že funkce má mít vrchol v jedním z polí matice, používá se pro výpočet, matice o lichém počtu sloupců. Rozměrem matice se určuje na jak velkým okolím je hodnota pixelu ovlivněna.

$$A = \begin{bmatrix} 1/16 & 1/8 & 1/16 \\ 1/8 & 1/4 & 1/8 \\ 1/16 & 1/8 & 1/16 \end{bmatrix} \tag{4.2}$$

Matice s koeficinety pro Gaussův filter.



(a) Zdrojový obraz (b) Aplikace filtru eroze

Obrázek 4.4: Ukázka několikrát aplikovaného filtru Gaussian.

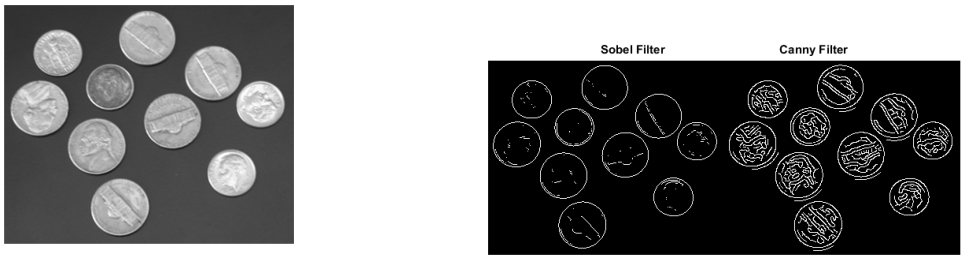
4.2.2 Detekce hran

Jestliže potřebujeme detekovat hrany v obraze, nabízí se nám několik možností. Nejpoužívanější možností je použití tzv. Sobelova operátoru. Tento operátor provádí výpočet první derivace v obraze. Výhodou tohoto filtru je rychlý výpočet pomoci konvoluce.

Mezi další často používané metody pro detekci hran patří Canny detektor pojmenovaný podle svého stvořitele jménem John F. Canny. Na rozdíl od detekce hran pomocí Sobelova filtru se tento algoritmus skládá ze tří kroků.

1. Odstranění šumu (nejčastěji podle Gaussova algoritmu).
2. Použití metody Non-maximum k odstranění pixelů, které nejsou součástí .
3. Prahování hran.
4. Odstranění nevýrazných hran.

Tento algoritmus se snaží extrahovat hrany, které jsou v obraze v kontextu, zatímco hrany, které jsou minoritní se snaží ignorovat. Z hlediska lokální navigace se hranový detektor dá použít k hledání hranice objektů např. překážek.



(a) Zdrojový obraz (b) Aplikace hranového filtru

Obrázek 4.5: Porovnání hranových filtrů [36].

4.2.3 Morfologické operace

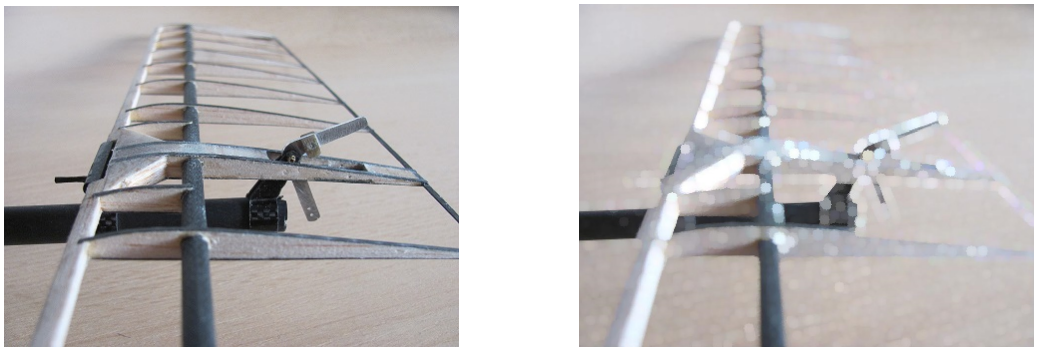
Tyto operace jsou založené na matematické morfologii, která vychází z vlastností bodových množin. V oboru počítačového vidění se morfologické operace používají k zjednodušení obrazu, doplnění nespojitých části nebo k rekonstrukci obrazu. Morfologické operaci jsou realizovány jako relace mezi dvěma bodovými množinami. V případě, že pracujeme s binárním obrazem jednotlivé body množiny náleží prostoru Z^2, zatímco při práci s obrazem v odstínech šedi jsou jednotlivé body množiny vymezeny v prostoru Z^3 [37]. Matematické definice následujících operací se vztahuje k binárnímu obrazu v prostoru Z^2.

Dilatace

Morfologický filtr dilatace rozšiřuje elementy obrazu. Názorně to můžeme vidět na obrázku níže, kde je znázorněn otisk prstu. Po aplikaci tohoto filtru začnou papilární linie splývat.

$$A \oplus B = \bigcup_{b \in B} (X)_b \tag{4.3}$$

kde A je množina obrazových bodů a B je symetrický strutkurní element.



(a) Zdrojový obraz. (b) Aplikace filtru dilatace.

Obrázek 4.6: Ukázka filtru dilatace.

(a) Zdrojový obraz. (b) Aplikace filtru dilatace.

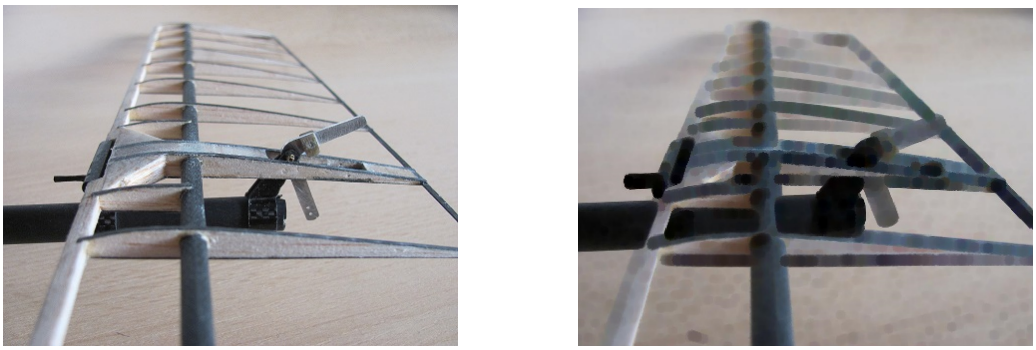
Obrázek 4.7: Ukázka filtru dilatace na černobílém obrázku.

Eroze

Morfologický filtr dilatace dokáže z obrazu odstranit detaily. Na obrázku otisku prstu můžeme vidět odstranění části obrazu, které byly příliš tenké.

$$A \ominus B = \bigcap^{b \in B} (X)_{-b} \tag{4.4}$$

kde A je množina obrazových bodů a B je symetrický strutkurní element.



(a) Zdrojový obraz. (b) Aplikace filtru eroze.

Obrázek 4.8: Ukázka filtru eroze.

(a) Zdrojový obraz (b) Aplikace filtru eroze.

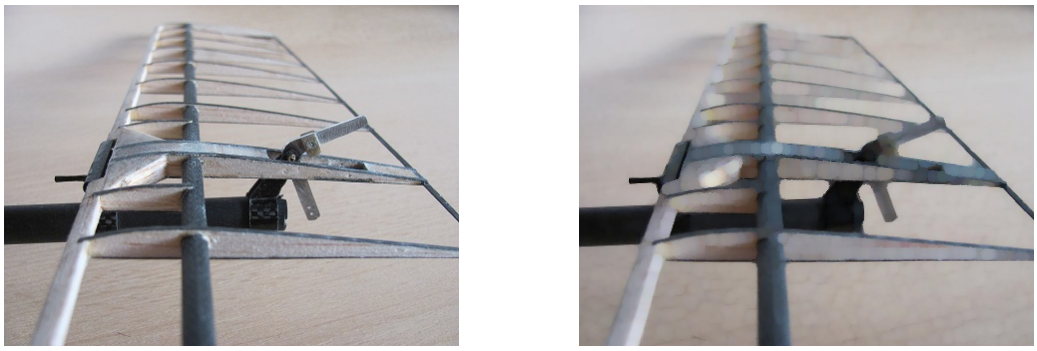
Obrázek 4.9: Ukázka filtru eroze na černobílém obrázku.

Otevření

V mnoha aplikacích jsou operace dilatace a eroze používaný sekvenčně pro dosažení požadovaných změn v obraze. Jednou takovou operaci je operace otevření, která z geometrického hlediska provádí vyhlazování kontur a odstraňování malých ploch [38].

$$A \circ B = (A \ominus B) \oplus B \tag{4.5}$$

kde A je množina obrazových bodů a B je symetrický strutkurní element.



(a) Zdrojový obraz. (b) Aplikace filtru otevření.

Obrázek 4.10: Ukázka filtru otevření.

(a) Zdrojový obraz. (b) Aplikace filtru otevření.

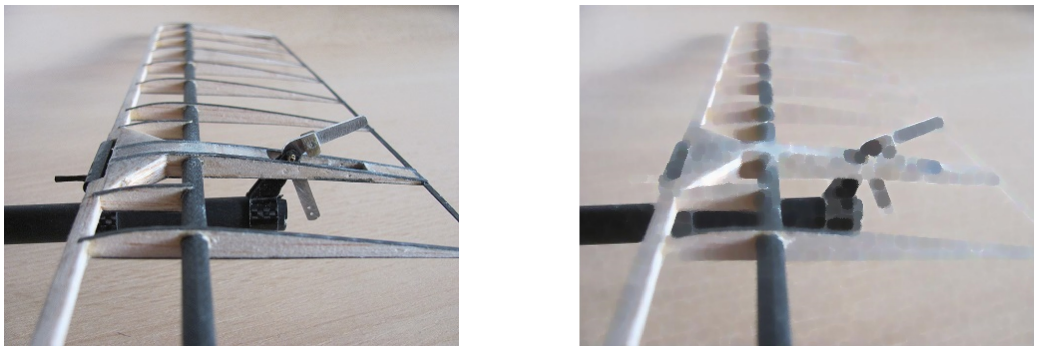
Obrázek 4.11: Ukázka filtru otevření na černobílém obrázku.

Uzavření

Operace uzavření je opět sekvenční metoda aplikace morfologických operací a z geometrického hlediska se projevuje uzavíráním kanálu a malých děr sousedící s plochou [38].

$$A \bullet B = (A \oplus B) \ominus B \tag{4.6}$$

kde A je množina obrazových bodů a B je symetrický strutkurní element.



(a) Zdrojový obraz (b) Aplikace filtru uzavření

Obrázek 4.12: Ukázka filtru uzavření.

(a) Zdrojový obraz (b) Aplikace filtru dilatace

Obrázek 4.13: Ukázka filtru uzavření na černobílém obrázku.

4.2.4 Výpočet disparity

Získávání hloubkové mapy probíhá na základě měření rozdílu pozice dvou totožných bodů na dvou různých snímcích zachycující tutéž scénu [39]. Čím větší je tento skok mezi dvěma stejnými body, tím je objekt blíže kamerám.

V ideálním případě by mělo platit, že bod na pozici v levém obraze $I_l(x, y)$ se musí nacházet i v pravém obraze na souřadnicích $I_r(x + d, y)$, kde x a y jsou souřadnice v obraze a d značí disparitu. V praxi se ovšem stává, že korespondující bod je změněn vlivem světelných podmínek nebo nepřesnosti snímacího čipu. Algoritmy pro výpočet disparity se snaží pomocí aproximace a algoritmů zaměřující se na vztah bodu k okolí tento problém částečně eliminovat.

BM algoritmus
Jedná se o základní používaný algoritmus, který funguje na základě vyhodnocování podobnosti bloku kolem pixelu p obrazu I_1 s bloky v epipolární linii obrazu I_2. Odkaz na tento princip je zahrnutý v názvu, kde BM znamená porovnávání bloků (Block Matching). Výhoda tohoto algoritmu spočívá v rychlosti zpracování. Nevýhodou je relativně velká chybovost. Algoritmus je součásti knihovny OpenCV verze 2.

SGBM algoritmus
Název SGBM (Semi Global Block Matching), že se jedná o algoritmus, který bude porov-

návat bloky v globálním rámci. Tento algoritmus je založený nejen na základě porovnání rozdílů v oblasti jednoho bloku, ale přidává informaci o rozdílech v sousedních blocích, tímto se zvyšuje šance, že se blok přiřadí na správné místo. Oproti předchozímu algoritmu má tento algoritmus menší chybovost cca o 10 % [40]. Algoritmus je součásti knihovny OpenCV verze 2.

Cílem algoritmu je minimalizace energie E snímku disparity D, podle tohoto vzorce:

$$E(D) = \sum_{p} \left(C\left(p, D_p\right) + \sum_{q \in N_p} P_1 I\left[|D_p - D_q| = 1\right] + \sum_{q \in N_p} P_2 I\left[|D_p - D_q| > 1\right] \right) \tag{4.7}$$

kde:

$E(D)$	– energie mapy disparity,
p, q	– reprezentují pixel v obraze,
N_p	– soused pixelu p,
$C(p, D_p)$	– cena pixelu s disparitou D,
P_1, P_2	– penalizace pro sousední pixel, přičemž musí platit $P_2 > P_1$,
$I[]$	– funkce která vrací 1 pokud je argument pravdivý, jinak 0.

Tato funkce dokáže teoreticky vytvořit perfektní mapu disparity, hlavní problém je ten, že minimalizace ve 2D prostoru je výpočetně NP problém. Pro řešení se používá metoda dynamického programování, při které se snažíme řešený problém rozdělit na podproblémy. Vyhodnocovací funkce SGBM algoritmu je založená na minimalizaci 2D prostoru pomocí 1D cest.

Celková cena pixelu je daná součtem minimálních cest ve směru r.

$$S(p, d) = \sum_{r} L_r(p, d) \tag{4.8}$$

Výpočet minimální ceny cesty se provádí pomocí [41]:

$$\begin{aligned} L_r(p, d) = C(p, d) + \min(&L_r(p - r, d), L_r(p - r, d - 1) + P_1, L_r(p - r, d + 1) + P_1, \\ &\min_i L_r(p - r, i) + P_2) - \min_k L_r(p - r, k) \end{aligned} \tag{4.9}$$

Pro výpočet ohodnocovací funkce $C(p, d)$ je možné použít, jak absolutní minimální rozdíl hodnot intenzit pixelu p v obraze jedna s pixelem q posunutým o parametr d. Další poněkud složitější možností je vypočítat ohodnocení pomocí vzájemné informace (MI - Mutal Information).

4.2.5 Extrakce oblastí objektů

V počítačovém vidění je často zapotřebí získat oblast, kde se objekt nachází. Na základě této informace můžeme dále provádět další zpracování. Cílem této kapitoly je najít tzv. oblast zájmu, v literatuře označované jako ROI (Region of Interest).

Connected-component labeling

Jedná se o algoritmus založený na teorií grafů. Zkoumáme, jak je pixel připojený k ostatním, dá se předpokládat, že pokud je pixel přípojený k jinému, bude stejného označení

jako pixel druhý. Takto můžeme postupně označit pixely všech objektů, v množině bodů reprezentujících objekt můžeme najít maximální a minimální souřadnice x a y.

Watershade transformace
Jedná se o jednu z nejstarší segmentačních technik se kterou přišli pánové Beucher a Lantuejoul. Tato segmentační metoda se dá zařadit mezi takzvané region based segmentační přístupy. Obrázek v odstínech šedi je chápan jako topografický element. Každý element je postupně zaplavován „virtuální vodou" od svého minima. Na rozhraní dvou těchto elementů (jezer) se postaví tzv. hráz. Jednotlivé hráze oddělují jednotlivé regiony a tím dochází k segmentaci obrazu. Procesy jak k tady toto provádět jsou popsány v publikacích Beucher (1990) a Soille (1992). Výhodou toho algoritmu je jeho nízká výpočetní náročnost v porovnání s ostatními metodami. Ovšem je zapotřebí myslet na předzpracování, kdy musíme převést obrázek do odstínu šedi a odstranit z něho vysokofrekvenční objekty, které by mohly narušit proces segmentace [42].

4.3 Algoritmus pro návrat do výchozí pozice

Nyní se zaměříme na implementaci algoritmu umožňující robotu návrat do výchozí polohy. Jak již víme z předchozích sekcí, klíčovým prvkem pro lokální navigaci je algoritmus realizující SLAM. Z analýzy možnosti systému ROS provedené v sekci Balíčky pro SLAM, víme, že ROS nabízí několik možností pro mapování a lokalizaci. Vstupem pro tyto balíčky jsou buď data z laserového skeneru, obrazová data ze stereo kamer nebo jiná data reprezentující prostor. Výstupem z takového balíčku je odhadnuta poloha robota a mapa prostředí.

Základem pro navigaci založenou na optických senzorech je SLAM balík *gmapping*, jaké má vlastnosti jsme si rozebrali v části 4.1.4. V systému ROS ovšem není balík pro získávání dat ze stereokamer od výrobce Prosilica, který je zapotřebí, kromě samotného algoritmu pro návrat zpět, také implementovat. Základním vstupem pro SLAM balík jsou data z laserového skeneru, která jsou ve formátu PCL(4.1.4). Ze SLAM balíku obvykle získáváme dva základní výstupy, prvním z nich je odhadnuta poloha a druhým z nich je mapa obsazenosti. Tento balík nám umožní získat mapu obsazenosti kolem robota a odhadnutou polohu na základě pozorování a odometrie.

Veškeré uvedené části programu jsou napsány v programovacím jazyce C++ s využitím možností standardní knihovny. Pro obrazové operace se používá výhradně knihovna OpenCV. Dále je potřeba zmínit, že program jako takový je realizovaný jako aplikace, která funguje pouze se systémem ROS.

4.3.1 Ovladač pro stereokamery

Systém ROS obsahuje pouze ovladač pro jednu stereokameru, v rámci projekt byla vytvořena knihovna pro získávání dat ze stereokamer. Tato knihovna zajišťuje synchronní získávání snímků, které je možné dále publikovat. Její hlavní nevýhodou je, že je implementovaná mimo systém ROS, tudíž data z těchto kamer nelze použít pro další balíčky apod. Proto byl vytvořený *node*, který realizuje komunikaci s knihovnou a následně obrazová data s využitím knihovny OpenCV publikuje.

Program byl vytvořený jako standardní balík pro ROS v programovacím jazyce C++ s využitím kódů z programu *StereoCamToPng*. Program je dělaný tak, že v zadaném intervalu získává data z kamer a publikuje je na dva kanály (*topic*), na dalších dvou publikuje

camera_ info zprávy. Tyto zprávy obsahují informace o obrazu s kamer a kalibrační informace. Práce s kalibrací však nebyla otestována z důvodu chybějících kalibračních souborů.

Pro kalibraci stereo kamer se používá program *camera_ calibaration*, který vygeneruje potřebné soubory pro ovladač. Jestliže ovladač tyto soubory nebude mít k dispozici, vypíše uživateli hlášku a nebude tyto zprávy poskytovat.

Publish		
	/left/image_raw	Obraz z levé kamery
	/right/image_raw	Obraz z pravé kamery
	/right/camera_info	Informace o obrazu z pravé kamery (netestováno)
	/left/camera_info	Informace o obrazu z levé kamery (netestováno)

Tabulka 4.1: Parametry ROS uzlu, **Subscribe** značí, které zprávy tento uzel přijímá a **Publisher** značí zprávy, které tento uzel vysílá.

Nutno ovšem podotknou, že zřejmě z důvodu nízkoúrovňového přístupu ke kamerám, musí být uzel spuštěný s vyšším uživatelským oprávnění. V případě Ubuntu (12.04) se spuštění provádí pomocí příkazu *sudo -i*. Pokud by program nebyl tímto způsobem spuštěný, nenašel by jednotlivé kamery.

4.3.2 Pohyb po trajektorii

Nyní se podíváme na část zajišťující pohyb robota, následující popis se bude vztahovat k trajektorií, kterou jsme si ukládali v předchozím kroku. Řízení robota probíhá pomoci zasílání zpráv typu *geometry_ msgs/Twist*. Tyto zprávy se zasílají uzlu, který zajišťuje řízení a přímé ovládání pohonů robota. Po přijetí zprávy tento ovladač provede na základě informací v této zprávě pohyb po specifikovanou dobu. Každá tato zprávy obsahuje hodnotu přímých a úhlových rychlostí. Pro pohyb pásového robota se definuje pouze rychlost ve směru y a úhlová rychlost ve směru y.

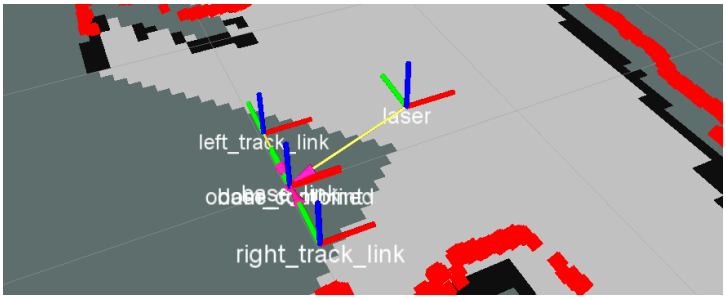
Trajektorie pohybu robota je tvořena body, četnost bodu přímo závisí na zvoleném intervalu ukládáni. ROS používá pro ukládáni trajektorie pole zpráv typu *nav_ msgs/Pose*. Trajektorie se postupně aktualizuje s průchodem robota mapou. Program jsem navrhl tak, aby ukládal jak trajektorii generovanou SLAM programem *gmapping*, která je ovšem dle dokumentace určena pouze k navigaci. Proto jsem si interně vytvářel další trajektorii založenou na transformačních zprávách. Program si následně postupně bere data o pozici robota z transformačních zpráv a ukládá je do struktur c++ vektor.

Aby se robot mohl vydat na svou cestu ve svém směru je zapotřebí znát vektor polohy robota. Vektor je možné získat výpočtem z transformačních zpráv reprezentujících model robota. Tyto zprávy obsahuji informaci o pozici přední části robota a jeho zadní části.

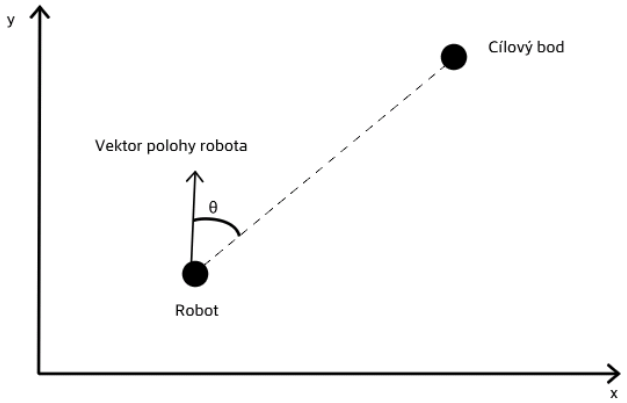
Získaná data o poloze jsou dostačující pro určení vektoru polohy robota. Následně je nutné spočítat na základě definované rychlosti, kterou se robot bude pohybovat do cíle a vzdálenosti, úhlovou rychlost. Jelikož máme dva vektory výpočet takového úhlu (Θ viz obrázek 4.15) je otázkou základních vzorců z trigonometrie.

Algoritmus je navržený tak, aby robot nikdy nepřekonával větší úhel než $\pi(180°)$. Pokud dojde při výpočtu k překročení této hodnoty, robot se vydá k cíli v opačném směru pohybu po kruhové výseči tzn. použije inverzní úhlovou rychlost.

Úhlovou rychlost program vypočítá tak, aby požadovaného úhlu bylo dosaženo s dosažením cíle. Když známe lineární rychlost, kterou si můžeme zvolit v rozumné rozsahu



Obrázek 4.14: Model robota tvořený transformacemi. Na základě polohy bodů *laser* a *base_link* je možné určit polohu robota.



Obrázek 4.15: Znázornění situace pro výpočet úhlu mezi vektorem reprezentujícím robota a cílový bodem.

libovolnou je možné na základě podílu cílového úlu a času v sekundách, který robot potřebuje k dojezdu k danému bodu spočítat úhlovou rychlost v rad/s.

Program provádí posíláni zpráv po dobu spočítaného času cesty. Po této činnosti se spočítá odchylka od předpokládaného bodu dojezdu, cesta k dalšímu se bude odvíjet od nového bodu, aby se minimalizovala chyba. V experimentech zhodnotíme velikost výchylky na různých cestách.

Samotný algoritmus funguje tak, že v případě, že uživatel zadá pokyn k návratu robota (prostřednictvím zprávy). Začne postupně brát body z trajektorie, vypočítá vzdálenost mezi body. Jelikož se může stát a taky se bude stávat, že robot zůstane stát na místě, je zapotřebí provést filtraci bodů a body mezi kterými je příliš malá vzdálenost jednoduše ignorovat. Vzdálenost je dále jedním z parametru programu, dalším měnitelným parametrem programu je rychlost pojezdu robota.

Korekce trajektorie

Při pohybu robota k cíli jsme měli k dispozici tři optické senzory, z kterých byl nejpřesnější laserový skener a nejméně přesná byla náhledová kamera. Při couvání nám zůstal pouze jeden optický senzor a tím je zadní náhledová kamera. Při pohybu robota dopředu si algoritmus postupně ukládal na mapu rozložení jednotlivých objektů v prostředí. Tyto objekty se při návratu robota zpět do výchozí pozice snaží robot detekovat v náhledové kameře a základě

nich provádět korekci dráhy.

Při průjezdu robot si robot vytvoří profil, který následně promítne do mapy. Proces vytvoření profilu je založený na základě metod počítačového vidění, které jsme si probrali v sekci 4.2 .Takovýto profil je následně promítnutý do interní mapy. Jestliže se robot vrací zpět snaží se provést porovnání s takovým profilem. Jestliže robot najde při zpáteční cestě před sebou podobný profil objektů nachází se pravděpodobně na stejné pozici, jako v předchozím případě.

Jestliže bude shoda mimo danou mez, algoritmus provede posun bloků mezi sebou o hodnotu offsetu. Tento posun se provede do obou stran, jestliže se dojde při porovnávání k zlepšení připraví se korekční posun robota v příštím kroku.

Algoritmus tyto profily porovnávaná pomocí metody vzájemného rozdílu obrazu s následným výpočtem chyby. Čím se obrazy více liší, tím tato chyba bude větší. Tento způsob prorvání byl použitý především z důvodu výpočetní rychlosti. Dalším důvodem použití tohoto způsobu bylo, že robot pro návrat zpět nepotřebuje vědět přesné rozložení objektů. Budeme li předpokládat, že se roboto při cestě k cíli a při cestě zpět bude dívat ze stejné pozice, bude vzdálenost k objektům tohoto profilu stejná, tudíž i výsledný profil by měl být z pohledu na obě strany podobný. Jakým způsobem tady toto funguje, zhodnotíme v kapitole experimentování.

Nyní se podíváme na tuto část z implementačního hlediska. Bude se jednat o knihovnu, která bude úzce komunikovat s knihovnou *RobotMove.h*. Výsledkem této knihovny bude hodnota o kterou by měl robot provést korekci nové polohy.

Ukládání vizuální profilů probíhá zároveň s ukládáním bodů trajektorie, před každým příkazem k pohybu robota se provede volání metody *compareWithActualProfile* třídy *VisualCorrection*. Tato metoda vrátí hodnotu korekce. Tato hodnota říká o kolik metrů by se měl provést posun vůči hodnotě získané z odometrie.

Bezpečností kontrola

Protože program pracuje s pohonem robota a Robot záchranář patří mezi větší typy robotů, které můžou způsobit nějakou škodu či zranění, je před každým odesláním prováděná kontrola parametrů řídící zprávy. Kontrolou zpráv se rozumí zjištění, zda některá ze zadaných rychlosti nepřesahuje povolené meze. Tyto meze jsou definovány jako ROS parametry.

4.3.3 Detekce objektu

V situaci, že operátor pošle robotu příkaz, aby se vrátil, robot se bude snažit po stejné trajektorii dojet do cíle. Ovšem nikde není zaručené, že cestu nezablokoval nový objekt. V tomto případě má robot dvě možnosti, buď zastavit anebo pokračovat v cestě. V případě druhé možnosti ovšem může dojít k poškození robota s fatálními důsledky, někdy se ovšem vyskytne objekt, který robot může přejet. Příkladem takového objektu může být spadlá suť, kterou měl by pásový robot bez problému překonat nebo obyčejná papírová krabice. Detekce výskytu překážek v prostoru před nebo za robotem se provádí pomocí sonických senzorů. Tato práce je ovšem zmařená na zpracování dat z optických senzorů, proto s využitím metod počítačového vidění budeme provádět detekci překážky z obrazu. Detektor je možné udělat několika způsoby, nejlepší možností ovšem je mít detektor natrénovaný přímo na konkrétní překonatelný typ překážky.

Jak jsem již výše uvedl, robot při návratu zpátky couvá, tudíž může využívat jediný optický senzor a tím je zadní náhledová kamera. Při cestě k cíli si robot pomocí balíku

gmapping vytvořil mapu prostředí. Na základě dat z mapy prostředí jsme schopni odhadnout, jak daleko jsou od robota překážky, které tam byly v čase, kdy se robot vydával k cíli.

V rámci jedné z mých předchozí prací [43] vznikl detektor osob v obraze, kterým jsem se rozhodl doplnit algoritmus návratu zpět. Tento detektor dokáže detekovat osoby. V případě, že by program detekoval pohybující se osobou před robotem, neprovedl by vyslání příkazu k pohybu robota.

4.3.4 Shrnutí

V předchozích kapitolách jsme se podívali, jak jsou navržené a realizované jednotlivé části algoritmu. Nyní se podíváme, jak jednotlivé části programu mezi sebou spolupracují. Program je tvořen třemi základními třídami *NavigationMap*, *RobotMove* a *VisualCorrection*. Další používanou třídou je *Detector*, která zajišťuje detekci osob v obraze.

Třída *NavigaionMap* vytváří interní mapu překážek, se kterou je možné dále pracovat. Původním cílem této třídy bylo mapování a ukládání deskriptorů obrazových segmentů prostředí. Ve výsledném algoritmu se tato třída používá pouze na propojení s detektorem osob.

Propojení mezi těmito třídami je takové, že třída *RobotMove* volá metodu *isOccupied* třídy *NavigationMap*. Tato třída následně zjistí, zda se na interní mapě, která je úzce propojená s mapou z *gmapping* vyskytuje objekt nebo ne. Pokud funkce třídy *NavigationMap* vyhodnotí, že robot má v cestě překážku, nepovolí robotu další pohyb. Robot následně zůstane stát a prostřednictvím výpisu informuje operátora o vzniklé situaci tak, aby operátor mohl provést zásah do řízení.

Pro algoritmus je ovšem mnohem důležitější třída *VisualCorrection*, která provádí korekci trajektorie při návratu zpět.

Další souborem používaným v tomto projektu je soubor obsahující struktury datových typů a různé jiné pomocné *inline* funkce s názvem *DataTypes.h*. Společně s výše uvedenými soubory tvoří základě celého projektu, ostatní soubory tvoří nastavení pro ROS a podobně.

V následující tabulce je uvedený přehled z kterých *topics* výsledný program odebírá a z kterých přijímá zprávy. Další informace se získávají z transformačních zpráv, které program získává prostřednictvím *tf_listener*.

Subscribe		
	/map	Mapa prostředí získaná z gmapping.
	/path	Trajektorie
	/v4l/camera_rear	Obraz ze zadní náhledové kamery
	/v4l/camera_front	Obraz z přední náhledové kamery
	/to_target	Pokyn k návratu robota zpět
Publish		
	/cmd_vel_safe	Zprávy pro řízení motorů
	/trajectory_to_target	Trajektorie od startu k cíli
	/trajectory_to_start	Trajektorie z cíle ke startu

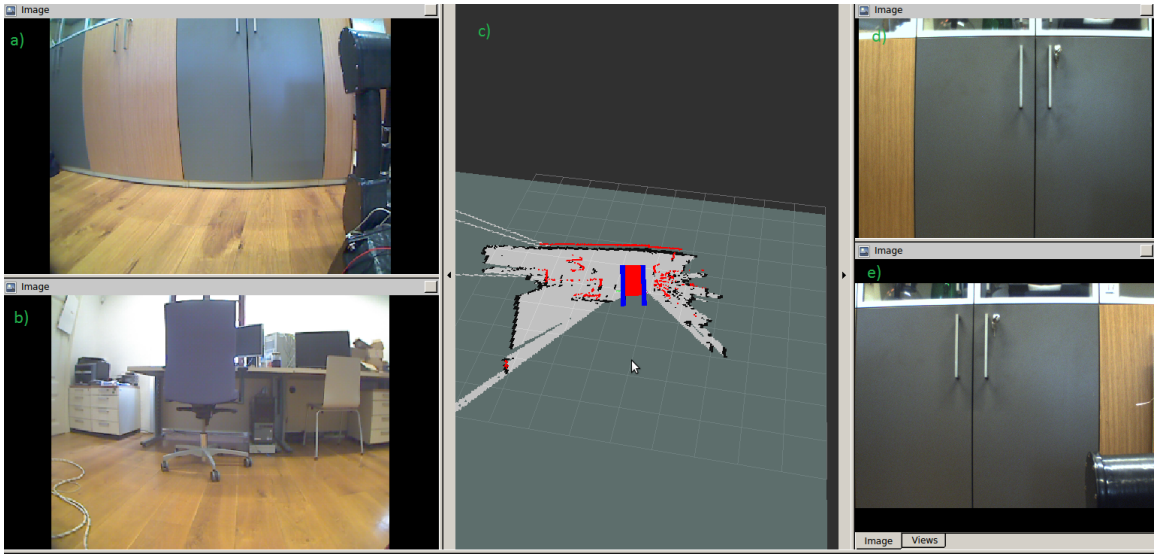
Tabulka 4.2: Parametry ROS uzlu, **Subscribe** značí, které zprávy tento uzel přijímá a **Publisher** značí zprávy, které tento uzel vysílá.

Kapitola 5

Experimenty a testování

Testování probíhalo s využitím SLAM balíku *gmapping*, který prováděl základní mapování prostředí s využitím dat z laserového skeneru. Jelikož obsluha robota s přesunem veškerého vybavení je organizačně náročná, probíhalo testování na ploše v areálu Fakulty informačních technologií. V rámci testování jsme se především zaměřili, jestli je robot schopný se vrátit po stejné trajektorii do výchozí pozice a s jakou odchylkou od původní trajektorie.

V další části jsem testoval detektor překážek, který má za cil detekovat objekty, které může přejet. Pro otestování tohoto detektoru jsem využil stejného prostředí, jako v předchozím případě.



Obrázek 5.1: Ukázka přijmu dat z optických senzorů určených pro navigaci: a)přední náhledová kamera, b)zadní náhledová kamera, c)mapa prostředí s daty z laserového skeneru, d),e) obrazy ze stereokamery.

5.1 Návrat do výchozí pozice

Testování probíhalo na trávníku a betonovém povrchu. Překážky jsme simulovali pomocí krabic a židlí. Robot byl řízený pomocí operátorské stanice, která umožňuje uživateli pohyb s robotem. Prostřednictvím joysticku byl robot naveden do cíle. Po dosažení cíle byla vyslána

zpráva vytvořenému ROS balíčku (programu) pro návrat do výchozí pozice. Jelikož tento program vytváří druhou trajektorii, která je založená na novým pozičních datech, lze tyto dvě trajektorie porovnat.

Jako metodu pro výpočet odchylky, od plánové (původní) trajektorie, jsem použil průměrovaný výpočet rozdílu vzdálenosti bodů trajektorie, kterými měl robot projet a které projel ve skutečnosti. Zjednodušeně řečeno jsem využil bod na mapě, kam měl robot dojet a spočítal jsem vzdálenost k bodu kam robot dle odometrir dojel. Postupně jsem počítal celkový průměr odchylky a detekoval minimální a maximální odchylku. Na základě tohoto můžeme zhodnotit s jakou přesností se robot pohybuje k plánovanému cíli.

5.1.1 Návrat bez korekce trajektorie

První experimenty probíhaly s vypnutím korekce trajektorie. Robot byl pomocí operační stanice vyslán na místo určení, následně dostal ROS zprávu s požadavkem pro návrat zpět do výchozí pozice. Minimální krok algoritmu byl nastavený na 0,3 metrů. Rychlost robota byla nastavená na 0,25 m/s.

Jakmile roboto přijel požadavek pro návrat zpět, začal couvat. Ačkoliv robot prostřednictvím příkazů dostával přesné informace, jakým směrem se má vydat, s dosažením některých bodů měl z důvodu prokluzu pásů problémy. Při testování jsem zjistil, že robot má problémy při větším otáčení. Problém spočívá v tom, že robot má relativně velkou hmotnost. Váha robota pak působí na podvozek, tím pádem se robot otáčí nebo jede pomaleji než o proti předpokladům.

Při experimentování byla vyhodnocována především průměrná odchylka od trajektorie. Celkem byl robot „požádán“ o návrat zpět třikrát. Průměrná odchylka se měnila s profilem trajektorie. V nejlepším případě její hodnota byla 0,12 metrů, v nejhorším případě pak 0,6 metrů. Ačkoliv jsem původně plánoval měření maximální a minimální odchylky

Testování bylo limitované délkou napájecího kabelu robota. Pohon na baterie nebyl v době experimentování funkční.

5.1.2 Návrat s korekcí trajektorie

Cílem tohoto experimentování je získat informace o funkci korektoru trajektorie. Nastavení robota pro tyto experimenty bylo stejné jak v předchozím případě. Korektor měl nastavenou korekci o 0,2 metru/pixel.

Správnost funkce korektoru byla především ovlivněna prostředím ve kterém se robot pohyboval. V případě, že se kolem robota nacházely objekty výrazně se lišící, dokázal provést určitou korekci trajektorie. Pokud ovšem kamery zachycovaly především vysokofrekvenční složky v obraze, docházelo k chybné korekci.

Odchylky se pohybovaly na stejné úrovni jako v předchozím případě. Při průjezdu po mírně zakřivené trajektorii, robot dokázal udržet odchylku na úrovní 0,1 metru. V ostatních případech se odchylka pohybovala řádově v desetinách metru. V situaci, že byl na cestě štěrk byla odchylka větší.

5.2 Detekce lidí

Dále jsem provedl testování detektoru lidi před robotem. Pro tento účel jsem využil natrénovaný detektor lidí a jako překážky jsem použil židle. Testování detektoru probíhalo tak, že se osoba postavila do zorného pole zadní náhledové kamery. Toto ovšem přinášelo problém, aby

se osoba do zorného pole vešla musela být příliš daleko od robota. Další problém se objevil i s detekcí, jelikož detektor potřeboval, aby osoba byla uprostřed obrazu. Celkem se z 10 pokusů o detekci povedlo detekovat osobu před kamerou pouze dvakrát. Funkci detektoru osob ve spojení s náhledovou kamerou na Robotu záchranáři hodnotím jako velmi špatnou.

5.3 Návrh pro další vývoj

Jak již bylo zmíněno v části zaobírající se vlastnostmi Robota záchranáře, jedná se o velice komplexní zařízení, na kterém lze realizovat nepřeberné množství projektů. Systém ROS poskytuje základní prostředky pro realizaci navigace jak ve venkovním prostředí, tak i ve vnitřních prostorech. Poznatky získané při práci na algoritmu pro návrat zpět, který má doplňovat lokální navigaci je možné použít pro vytvoření komplexní navigace robota. V rámci této práce byla vytvořená aplikace, která pracuje s metodami počítačového vidění s cílem provádět korekci dráhy robota při návratu do výchozího bodu. Pro další vývoj lokální navigace je možné využít ovládač pro stereokamer, jež vznikl při této práci.

Komplexní navigace by dokázala analyzovat prostředí kolem robota a na základě různých klasifikátoru by určovala typy jednotlivých objektů. Taková navigace by pak měla spoustu výhod v tom, že by robot mohl některé objekty ignorovat jako překážky a vydat se na cestu přes ně. Představme si situaci, že by robot mohl operovat v oblasti, kde je vysoká tráva. Jestliže by tráva zasahovala do oblasti ve které pracuje laserový skener, byla by automaticky vyhodnocená jako překážka.

Takovýto algoritmus by byl výpočetně náročný, proto se nabízí jedna z dalších možností, kterou poskytuje robot záchranář a tou je využití grafické karty pro paralelní výpočty. V sekci 2.2.4 jsme si uvedli, že počítač robota obsahuje grafickou kartu od výrobce NVIDIA.

Dalším možností rozšířené navigace, kterou by bylo možné realizovat na Robotu záchranáře je navigace založená na dalších typech senzorů, například by se nabízela možnost využití sonických senzorů k detekci překážek.

5.4 Shrnutí

Tato kapitola se zaobírala testováním algoritmů pro návrat zpět. Důležitým poznatek získaným při testování programu a experimentováním s robotem bylo zjištění, že robot při zatáčení má relativně velký prokluz pásů vůči povrchu, což způsobuje odchylku v pozici, kam dojel vůči pozici, kam měl dojet. Korekce polohy robota prostřednictvím vizuálních senzorů funguje jen v některých prostředích. Na konci této části byly shrnuty další možnosti v rámci vývoje navigace pro Robota záchranáře.

Kapitola 6

Závěr

Tato práce se zaměřila především na analýzu možností lokální navigace. Byly zhodnocené základní aspekty a používané algoritmy, jak pro lokální navigaci, tak i pro zpracování obrazu. V první kapitole jsem se zaměřil na roboty využívající navigací. Někteří z těchto robotů se používají v praxi, někteří slouží pro výzkumné účely. Obecně lze tvrdit, že s rozvojem výpočetní techniky se zlepšily možnosti pro lokální navigaci, které se proto stávají dostupnějšími a v budoucnosti mohou zasahovat do konzumního segmentu trhu.

Během projektu jsem několikrát pracoval s Robotem záchranářem. Robot obsahuje velké množství senzorů i výpočetních prostředků, což představuje širokou škálu možností pro realizaci, a to nejen lokální navigace. Výhodou robota je že používá systém ROS, který nabízí balíčky pro řešení různých úloh (navigace, řízení robota, atd.). Jedná se o novou platformu, na které je možné v budoucnosti realizovat další projekty. Během řešení jsem se potýkal s problémy nejen v rámci vývoje programu pro návrat do výchozí pozice, ale i problémy přímo s Robotem záchranářem. Nutno však podotknout, že se dařilo jakékoliv nečekané situace nebo problémy úspěšně řešit s realizačním týmem projektu. V době kdy jsem prováděl řešení práce, se realizační týmu podařilo odstranit spoustu nedostatky a výrazně zlepšit funkční vlastnosti robota. V prvotní fázi byla realizace programu pracujícího s lokální navigaci velice obtížná, avšak s postupným vývojem se dal program vyzkoušet v reálném prostředí a získat výsledky k celkovému zhodnocení programu. Z hlediska lokální navigace robot ve finále poskytoval názorné grafická data prostřednictvím programu *rviz*.

Z hlediska prostředků pro řešení lokální i globální navigace je Robot záchranář zcela na úrovní špičkových moderních robotů. Při srovnání vycházím z analýzy dostupných robotů, kterou jsem provedl v úvodu této práce. Například robot BigDog (viz 2.2.2) má pro řešení navigace stejné prostředky jako Robot záchranář, a to se jedná o robota vyvíjeného firmou patřící ke světové špičce v oboru robotiky. Z této analýzy vyplývá, že na Robotu záchranáři je možné v budoucnu realizovat různé projekty, které by se mohly potencionálně dostat do praxe a zvýšit tak význam robotiky v určitém odvětví.

Dále jsem se podíval na prostředky určené pro realizaci lokální navigace s využitím optických senzorů. Přesnost lokální navigace se v posledních desetiletích značně zlepšila díky laserovým skenerům, které jsou schopny přesně měřit vzdálenost k překážkám ve svém dosahu. Na základě algoritmu pro porovnávání shody jednotlivých laserových měření, jsme schopni určit o kolik metrů se robot posunul nebo spočítat jeho novou polohu. Další možnost lokalizace je s využitím dat ze stereokamery. Získaná data však už nejsou tak přesná. Navíc je výpočetně náročné získat z nich hloubkovou mapu, která se navíc musí pomocí metod pro zpracování obrazu upravit tak, aby neobsahovala například šum.

V rámci praktické části této práce vytvořen program pro návrat robota do výchozí pozice.

Tento program byl otestován v prostředí, které bylo podobné klasickému městském prostředí, kde se nejčastěji vyskytuje asfalt, beton a občas travnatá plocha. Po spuštění programu se robot vrátil s jistou odchylkou zpátky na místo ze kterého vyjel. Průměrná odchylka od původní trajektorie byla v rozmezí 0,33-0,6 m. Odchylka je způsobená více faktory, prvním faktorem je nepřesnost pohybu na pásech, mezi další faktory je určitě zapotřebí zařadit chyby vzniklé špatnou lokalizací. Pro účel zlepšení lokalizace byl vyzkoušen korektor trajektorie založený na porovnávání obrazu. Správnost funkce tohoto korektoru především záležela na prostředí a nastavené korekční hodnotě.

Literatura

[1] ORSÁG, F.; DRAHANSKÝ, M.; a spol: Robot pro hledání osob v závalech a lavinách. Technická zpráva, Vysoké učení technické v Brně, Fakulta informačních technologií, 2015.

[2] SMUTNÝ, L.: Akční členy [online]. 2014, [cit. 30.4.2015].
Dostupné z: <http://www.e-automatizace.cz/ebooks/ridici_systemy_akcni_cleny/Akc_el.html>

[3] NIXON, M.; AGUADO, A.: *Feature Extraction & Image Processing, Second Edition.* Academic Press, druhé vydání, 2008, ISBN 0123725380, 9780123725387.

[4] MACCCORMICK, J.: How does the Kinect work? [online]. 2011, [cit. 1.2.2015].
Dostupné z: <http://users.dickinson.edu/~jmac/selected-talks/kinect.pdf>

[5] BHATTI, A.: *Current advancements in stereo vision.* InTech, 2012, ISBN 9789535106609.

[6] GATZIOLIS, D.; ANDERSEN, H.-E.; Pacific Northwest Research Station (Portland, O.: *A guide to LIDAR data acquisition and processing for the forsts of the Pacific Northwest [electronic resource] / Demetrios Gatziolis and Hans-Erik Andersen.* U.S. Dept. of Agriculture, Forest Service, Pacific Northwest Research Station Portland, OR, 2008, 32 s. .

[7] HERITAGE, G.; LARGE, A.: *Laser Scanning for the Environmental Sciences.* Wiley, 2009, ISBN 9781444311945.

[8] UNAVCO: UNAVCO IDV: Displays of LIDAR and TLS Point Clouds [online]. 2014, [cit. 23.5.2015]. Obrázek ve formátu PNG.
Dostupné z: <http://www.unavco.org/software/visualization/idv/IDV_datasource_point_cloud.html>

[9] MACMANUS, C.: Samsung teases robotic vacuum cleaner with a twist [online]. 2013, [cit. 23.5.2015]. Obrázek ve formátu JPEG.
Dostupné z: <http://www.cnet.com/news/samsung-teases-robotic-vacuum-cleaner-with-a-twist/>

[10] BigDog - Most Advanced Rough-Terrain [online]. 2014, [cit. 2.1.2015].
Dostupné z: <http://www.bostondynamics.com/robot_bigdog.html>

[11] WOODEN, D.; MALCHANO, M.; BLANKESPOOR, K.; aj.: Autonomous navigation for BigDog. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, ISSN 1050-4729, s. 4736–4741.

[12] LITMAN, T.: Autonomous Vehicle Implementation Predictions. 2015, [cit. 26.4.2015]. Dostupné z: <http://www.vtpi.org/avip.pdf>

[13] ETHERINGTON, D.: Google's Self-Driving Car Project Is A World's Fair Fantasy Turned City Street Reality [online]. 2014, [cit. 23.5.2015]. Obrázek ve formátu JPEG. Dostupné z: <http://techcrunch.com/2014/05/14/googles-self-driving-car-project-is-a-worlds-fair-fantasy-turned-city-street-reality/>

[14] KNIGHT, W.: Driverless Cars Are Further Away Than You Think [online]. 2013, [cit. 5.1.2015]. Dostupné z: <http://www.technologyreview.com/featuredstory/520431/driverless-cars-are-further-away-than-you-think/>

[15] ORSÁG, F.: Robotika. 2011, [interní opora do předmětu Robotika].

[16] DOBEŠ, K.: Sídlo GSA v Praze [online]. 2015, [cit. 19.4.2015]. Dostupné z: <http://www.czechspaceportal.cz/2-sekce/agentura-gsa/sidlo-gsa-v-praze/>

[17] SULLIVAN, M.: A brief history of GPS [online]. 2012, [cit. 22.4.2015]. Dostupné z: <http://www.techhive.com/article/2000276/a-brief-history-of-gps.html>

[18] COOKSEY, D.: Understanding the Global Positioning System (GPS) [online]. 2012, [cit. 24.5.2015]. Dostupné z: <http://www.montana.edu/gps/understd.html>

[19] SICKLE, J. V.: *GPS for land surveyors*. CRC Press, 2011.

[20] VOJÁČEK, A.: MEMS - díl 1. - Co to je a jak to vypadá ? [online]. 2006, [cit. 23.5.2015]. Dostupné z: <http://www.hw.cz/clanek/2006111901>

[21] VOJÁČEK, A.: Jak pracují nové 3D MEMS akcelerometry Freescale ? [online]. 2007, [cit. 23.5.2015]. Dostupné z: <http://www.hw.cz/soucastky/jak-pracuji-nove-3d-mems-akcelerometry-freescale.html>

[22] RIISGAARD, S.; BLAS, M. R.: SLAM for Dummies. *A Tutorial Approach to Simultaneous Localization and Mapping*, ročník 22, č. 1-127, 2003: str. 126.

[23] FISCHLER, M. A.; BOLLES, R. C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, ročník 24, č. 6, Červen 1981: s. 381–395, ISSN 0001-0782, doi:10.1145/358669.358692. Dostupné z: <http://doi.acm.org/10.1145/358669.358692>

[24] ROZMAN, J.: *Navigace mobilních robotů*. Dizertační práce, Vysoké učení technické v Brně, 2011. Dostupné z: <http://www.fit.vutbr.cz/research/view_pub.php.cs?id=9825>

[25] FARAGHER, R.: Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation [Lecture Notes]. *Signal Processing Magazine, IEEE*, ročník 29, č. 5, Sept 2012: s. 128–132, ISSN 1053-5888.

[26] HASPER, P.: Indoor pedestrian navigation based on recursive filtering [online]. 2013, [cit. 23.5.2015]. Obrázek ve formátu JPEG.
Dostupné z: <http://bilgin.esme.org/BitsBytes/KalmanFilterforDummies.aspx>

[27] DIJKSTRA, E.: A note on two problems in connexion with graphs. *Numerische Mathematik*, ročník 1, č. 1, 1959: s. 269–271, ISSN 0029-599X, doi:10.1007/BF01386390.
Dostupné z: <http://dx.doi.org/10.1007/BF01386390>

[28] JOYNER, D.; NGUYEN, M. V.; PHILLIPS, D.: Algorithmic Graph Theory and Sage [online]. 2013, [cit. 10.5.2015].
Dostupné z: <http://code.google.com/p/graphbook/>

[29] ZBOŘIL, F.; ZBOŘIL, F.: Základy umělé inteligence, 2012, interní oporad předmětu Základy umělé inteligence.

[30] DONG, J.; LIU, B.; PENG, K.; aj.: Robot Obstacle Avoidance based on an Improved Ant Colony Algorithm. In *Intelligent Systems, 2009. GCIS '09. WRI Global Congress on*, ročník 3, May 2009, s. 103–106, doi:10.1109/GCIS.2009.307.

[31] SEKANINA, L.: Evoluční návrh analogových obvodů a antén, vliv prosředí, 2015, [cit. 20.4.2015].
Dostupné z: <https://www.fit.vutbr.cz/study/courses/BIN/private/prednasky/bin2015_p07.pdf>

[32] SMITH, T.: About ROS [online]. 2014, [cit. 2.12.2014].
Dostupné z: <http://www.ros.org/about-ros/>

[33] HORNUNG, A.; WURM, K. M.; BENNEWITZ, M.; aj.: OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, ročník 34, č. 3, 2013: s. 189–206, ISSN 0929-5593.

[34] SMITH, T.: ROS [online]. 2014, [cit. 5.5.2014].
Dostupné z: <http://www.ros.org/>

[35] Robert FISHER, A. W. E. W., Simon PERKINS: Gaussian Smoothing [online]. 2003, [cit. 30.4.2015].
Dostupné z: <http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm>

[36] MathWorks, T.: MATLAB Examples [online]. 2015, [cit. 10.3.2014]. Obrázek ve formátu PNG.
Dostupné z:
<http://www.mathworks.com/examples/image/2102-detect-edges-in-images>

[37] GONZALEZ, R. C.; WOODS, R. E.: *Digital Image Processing*. Pearson/Prentice Hall, 2008, ISBN 9780131687288.

[38] LOUI, A.; VENETSANOPOULOS, A.; SMITH, K.: Flexible architectures for morphological image processing and analysis. *Circuits and Systems for Video Technology, IEEE Transactions on*, ročník 2, č. 1, Mar 1992: s. 72–83, ISSN 1051-8215.

[39] NIKOLOVA, I.; NIKOLOV, A.; ZAPRYANOV, G.: Depth Estimation Using Shifted Digital Still Camera. 2011: s. 234–240.

[40] HIRSCHMÜLLER, H.: Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, ročník 2, June 2005, ISSN 1063-6919, s. 807–814 vol. 2, doi:10.1109/CVPR.2005.56.

[41] Corporation, N. I.: Semi-Global Block-Matching Algorithm [online]. 2015, [cit. 10.3.2014].
Dostupné z: <http://zone.ni.com/reference/en-XX/help/372916M-01/nivisionconceptsdita/guid-53310181-e4af-4093-bba1-f80b8c5da2f4/>

[42] Lamia Jaafar Belaid, W. M.: Image segmentation: a watershed transformation algorithm. *Image Analysis and Stereology*, ročník 28, č. 2, 2011: s. 93–102, ISSN 1854-5165.
Dostupné z: <http://www.ias-iss.org/ojs/IAS/article/view/852>

[43] GOLDMANN, T.: *Sledování hlídaného prostoru a detekce narušení bezpečnosti kamerovým systémem.* Diplomová práce, Vysoké učení technické v Brně, 2013.

Příloha A

Obsah CD

- **src** - zdrojové kódy programu
- **doc** - text diplomové práce
- **tex** - zdrojové soubory k textu práce
- README - pokyny pro práci s programy
- stereo.launch - ROS skript pro spuštění stereokamery
- return_to_start.launch - ROS skript pro spuštění programu pro návrat zpět

Příloha B

Manuál

K práci jsou přiložené dva programy, jedním z nich je ovládač pro stereokameru Prosilica. Druhým programem je implementace algoritmu pro návrat robota do výchozí pozice. Programy jsou určeny pro ROS verze Hydro. Překlad se provádí v prostředí *catkin*. Na CD jsou k dispozici pro oba programy *launch file*. Zde uvádím jen základní informace, další je možné nalézt na CD v souboru README.

B.0.1 Ovládač pro stereokameru

Ovládač pro stereo kamery je zapotřebí spouštět s vyšším uživatelským oprávněním. Po spuštění tento program začne získávat data ze stereokamery a publikovat je na potřebné kanály. Seznam kanálu je uvedeny v části této práce zabývající se implementací.

Program se spouští pomocí:

```
rosrun stereo_prosilica_driver stereo_prosilica_driver
```

B.0.2 Návrat od výchozí pozice

Tento program po startu začne mapovat trajektorii, po které se robot pohybuje. Stejně jako v předchozím případě je seznam kanálu (*topic*) uvedeny v části této práce zabývající se implementací. Pro správnou funkci programu je nezbytné, aby své data publikovala, jak zadní náhledová kamera, tak i přední kamera. Program je zapotřebí spustit předtím než se zahájí cesta s robotem k cíli.

Program se spouští pomocí:

```
rosrun robot_return robot_return
```

Příkaz pro zaslání zprávy zpět:

```
rostopic pub -1 /to_target stds_msgs/Bool '{data: true}'
```