VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

BEZKOLIZNÍ NAVIGACE MOBILNÍHO ROBOTU

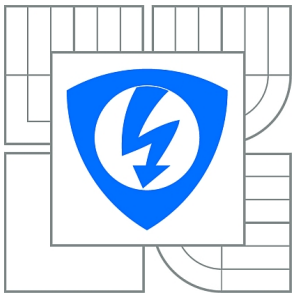
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. VLADIMÍR STŘÍTESKÝ
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

BEZKOLIZNÍ NAVIGACE MOBILNÍHO ROBOTU

MOBILE ROBOT NAVIGATION WITH OBSTACLE AVOIDANCE

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

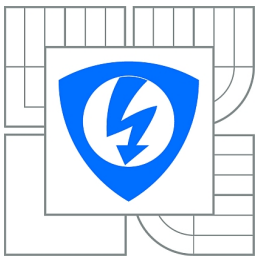
AUTOR PRÁCE — Bc. VLADIMÍR STŘÍTESKÝ

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. LUDĚK ŽALUD, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student: Bc. Vladimír Stříteský ***ID:*** 134409
Ročník: 2 ***Akademický rok:*** 2014/2015

NÁZEV TÉMATU:

Bezkolizní navigace mobilního robotu

POKYNY PRO VYPRACOVÁNÍ:

Zabývejte se tématem autonomní jízdy mobilního robotu s detekcí překážek v naplánované trajektorii jízdy a její přeplánování v případě kolize překážek a trajektorie. Pro reálnou aplikaci vyberte vhodná měřicí zařízení z dostupného vybavení na UAMT.

1. Vytvořte rešerši na zadaná témata.
2. Vyberte a použijte pro danou aplikaci vhodná měřicí zařízení.
3. Navrhněte a implementujte algoritmus detekce kolizních překážek.
4. Navrhněte a implementujte algoritmus přeplánování trasy.
5. Navrhněte a implementujte komunikaci s již existujícím modulem autonomní navigace.
6. Navržené řešení sestavte pro provedení reálných experimentů.
7. Body 3 až 5 ověřte v simulaci i v reálných venkovních podmínkách.
8. Zhodnoťte dosažené výsledky a definujte podmínky funkčnosti navrženého řešení.

DOPORUČENÁ LITERATURA:

HOLLAND, J. Designing Autonomous Mobile Robots. Newnes, c2004. 335 s. ISBN 0-7506-7683-3.

Termín zadání: 9.2.2015 ***Termín odevzdání:*** 18.5.2015

Vedoucí práce: doc. Ing. Luděk Žalud, Ph.D.
Konzultanti diplomové práce: Ing. Tomáš Jílek

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Práce se zabývá autonomní jízdou robotu z hlediska vyhýbání se překážkám v již naplánované trase. Jsou zde shrnuty běžně používané snímače pro detekci překážek a algoritmy pro hledání cesty. Na těchto základech je navrženo vlastní řešení problému využívající změnu naplánovaných bodů trasy. Pro test funkčnosti řešení je vytvořena simulace v prostředí MATLAB. Pro test v reálném prostředí je navržená metoda aplikována na reálný robot. V závěru práce jsou shrnuty dosažené výsledky a návrhy na zlepšení.

KLÍČOVÁ SLOVA

detekce překážek, měření vzdálenosti, TOF kamera, RTK GNSS, hledání cesty, MATLAB simulace, autonomní pohyb

ABSTRACT

Thesis deals with automatic guided mobile robot focused on obstacle avoidance during ride on planned route. There are summaries of usually used obstacle detecting sensors and algorithms used for path finding. Based on this, own solution is designed. It uses waypoints changes to pass obstacle. MATLAB simulation is created for tests of new designed method. This method is implemented to real robot for real world testing. Reached goals and upgrade possibilities are summarized in bottom of thesis.

KEYWORDS

obstacle detection, distance meassurement, TOF camera, RTK GNSS, pathfinding, MATLAB simulation, autonomous movement

STŘÍTESKÝ, Vladimír *Bezkolizní navigace mobilního robotu*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2015. 77 s. Vedoucí práce byl doc. Ing. Luděk Žalud, PhD.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Bezkolizní navigace mobilního robotu“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Děkuji především konzultantovi ing. Tomáši Jílkovi a vedoucímu doc. ing. Luďkovi Žaludovi za ochotnou spolupráci a cenné rady a připomínky při tvorbě této práce. Dále pak všem, jenž mi byli nápomocni při realizaci výsledného programu.

Brno ....

podpis autora(-ky)

OBSAH

SEZNAM OBRÁZKŮ

1 ÚVOD

Prvním cílem práce je prostudovat stávající běžně používané metody pro autonomní navigaci. Požadavek na schopnost autonomní jízdy vyplývá z účelu konstrukce robotu. Nejedná se o klasický problém typu najdi cestu ze stávající pozice do cíle, ale o mírně komplikovanější variantu.

Robot, pro který je metoda vyvíjena, je vývojová verze robotu Orpheus. Úkolem tohoto robotu je prozkoumávat oblasti, jenž jsou pro člověka nebezpečné. Jde tedy například o chemicky či radiačně zamořené oblasti. Předpokládá se, že robot bude řízen operátorem z bezpečné vzdálenosti. Při cestě do cílové zóny budou pravidelně ukládány souřadnice bodů projeté cesty. Po následném splnění úkolu a prozkoumání oblasti se musí robot navrátit zpět do výchozího bodu.

Právě v této situaci nastává moment, kdy je již manuální řízení robotu v podstatě zbytečné. Stačí, když robot bude umět následovat zaznamenané body trasy v opačném pořadí. Prakticky však může na trase dojít ke změnám, jenž způsobí její neprostupnost. Může se jednat například o spadlý strom, trosky zhroucené budovy či jen zaparkovaný automobil.

Z tohoto důvodu je vhodné, aby robot byl opatřen takovým systémem, jenž dokáže překážku detekovat a naplánovat objízdnou trasu. Objízdná trasa by měla být co nejkratší. Zároveň by se měla napojit zpět na původní trasu, co nejdříve je to možné.

Prvním úkolem je vybrat dostupný snímač vhodný pro detekci překážek. Spektrum běžně používaných snímačů je opravdu široké. Jak z hlediska jejich fyzikálního principu, tak metody samotného měření. Totéž pak platí pro typy překážek, které se kromě tvaru liší i v materiálu. Ne úplně obvyklým, ale neopomenutelným typem překážky v reálném prostředí je i hlubší díra, jako například chybějící poklop kanalizace.

Souvisejícím úkolem je také zhodnotit používané algoritmy pro vyhýbání se překážkám. Některé typy algoritmů jsou přímo závislé na nějakém typu senzoru. Algoritmů je opět celé spektrum. Liší se jak výpočetní náročností, tak schopností od objetí jednoduché překážky, až po schopnost najít cestu skrz neznámé bludiště.

Návrh algoritmu pro detekci překážek, současně s algoritmem pro přeplánování trajektorie jsou nejdůležitější částí práce. Jejich funkčnost bude nejprve ověřena počítačovou simulací.

Pro ověření funkčnosti v reálném prostředí je třeba implementovat navržené algoritmy pomocí počítačového programu. Tento program bude následně komunikovat s modulem zajišťujícím autonomní průjezd bodů trasy. Tomuto modulu bude předávat informace o nutných změnách v trajektorii, aby nedošlo ke kolizi.

2 TEORETICKÝ ROZBOR

V první části práce je proveden rozbor na základě studia literatury zabývající se autonomní jízdou mobilních robotů, především pak vyhýbáním se překážkám. Jsou zde podrobně popsány existující metody pro řešení obdobných zadání. Pro přehlednost, je celý problém rozdělen do dvou hlavních částí. První se týká problematiky detekce překážek, což je problematika týkající se především vhodných snímačů pro jejich detekci a lokalizaci. Druhá část se pak zabývá popisem existujících algoritmů používaných pro objetí překážky či hledání trasy.

2.1 Snímače pro detekci překážek

Snímače pro detekci překážek jsou obvykle zvláštním způsobem použité snímače vzdálenosti. Při jejich známém umístění na robotu můžeme z jejich dat vypočítat polohu překážky. Tyto snímače je možné rozdělit do skupin podle několika kritérií.

Množství dat

První a možná také nejdůležitější dělení snímačů je dle získávané informace, či spíše dle množství dat, které lze daným snímačem získat. Záleží, zda na výstupu snímače získáváme pouze jedinou hodnotu, například vzdálenost nejbližšího objektu v daném směru. V absolutně nejjednodušším případě získáváme pouze informaci o přítomnosti/nepřítomnosti objektu v oblasti, tedy binární výstup. Typicky sem patří kapacitní, indukční či taktilní senzory, jenž se běžně používají v průmyslu, ale pro mobilní roboty nejsou příliš vhodné.

Naopak u složitějších snímačů jde o soubor většího množství informací, například řádková či dvourozměrná matice vzdáleností. S množstvím získávaných dat, ale také roste výpočetní náročnost jejich zpracování. Zatímco data z jednoduchého snímače vzdálenosti lze zpracovat velmi snadno, i bez výpočetní techniky. Data z maticových snímačů jsou obvykle v číslicovém formátu a potřebují tak více, či méně složité číslicové zpracování. Před rozvojem digitální techniky se i vícerozměrné signály zpracovávaly analogově, například u leteckého radaru.

Každopádně s větším množstvím dat roste množství informací o okolním prostředí. Tím se rozšiřují i možnosti plánování trasy. Lze pak použít komplexnější algoritmy, které dosahují lepších výsledků.

Fyzikální princip

Další dělení je podle použitého fyzikálního principu. Velmi časté je z tohoto hlediska měření TOF, nebo-li doby letu elektromagnetické či zvukové vlny k objektu a zpět.

Při použití tohoto principu existuje několik možných způsobů měření, od prostého vyslání impulsu a čekání na odražený signál, až po měření fáze odraženého signálu. V tomto případě se již jedná o konkrétní způsob měření daného fyzikálního principu. Speciální kategorií zde může být například laserový interferometr, jenž je velice přesný i na velké vzdálenosti, ale pro použití v mobilní robotice se nehodí.

Další možností měření jsou triangulační metody, jenž obvykle pracují s laserovým paprskem a maticovým či řádkovým snímačem.

Dle fyzikálního principu lze některé snímače označit za bodové, protože jejich snímací úhel je velmi úzký. Jde především o snímače využívajících laserový paprsek. Jiné snímače naopak úmyslně snímají širší okolí.

Dle výše popsaných principů, se významně liší použitelnost jednotlivých snímačů v různých prostředích a aplikacích. Jednotlivé snímače a jejich přesné principy a možnosti použití jsou popsány dále.

2.1.1 Ultrazvukový dálkoměr

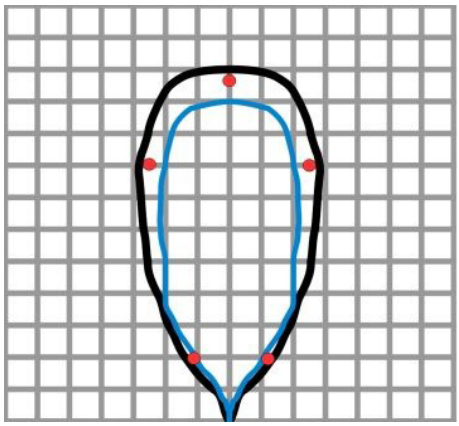
V oblasti drobné elektroniky, malých robotů či hraček je ultrazvukový dálkoměr jedním z nejpoužívanějších. Vděčí za to především nízké ceně a kompaktním rozměrům. Pracuje na principu měření TOF. K vytváření a přijímání ultrazvukových vln slouží piezokrystalové prvky. V některých snímačích je vysílací a přijímací člen realizován jediným prvkem.

Existují dvě základní modifikace, dle výstupního signálu. První je digitální, který poskytuje informaci o přítomnosti/nepřítomnost objektu. Takovýto výstup je pro mobilní robotiku obvykle nedostatečný. Proto se zde v převážné míře používá snímač s analogovým výstupem, jenž poskytuje údaj o vzdálenosti objektu. Přesněji je to údaj o vzdálenosti nejbližšího objektu v zorném poli. Některé snímače mají pokročilejší rozhraní, kde je možné naměřenou hodnotu vyčítat pomocí digitální komunikace, například po sériové lince RS232.

Další vlastnosti ultrazvukového snímače je poměrně široký zorný úhel (obr. 2.1). Dále schopnost měřit malé vzdálenosti, někdy i od nuly a schopnost detekovat i čiré překážky, například skleněné dveře.

Nevýhodou tohoto snímače je neschopnost detekovat šikmé a hladké překážky, od kterých se vyslaný signál neodrazí zpět ke snímači. Také objekty s velkým akustickým útlumem, jako například molitan, jsou obtížně detekovatelné. Efektivní zorný úhel snímače je závislý především na povrchové struktuře překážky.

Jelikož tento dálkoměr získává o okolním prostředí jen velmi málo informací, je často umístěn na otočnou plošinu se známým úhlem natočení. Pomocí takového mechanismu lze získat výrazně více informací o poloze překážek v okolním prostředí.



Obr. 2.1: Zorný úhel ultrazvukového snímače [10]

Obr. 2.2: Příklad ultrazvukového snímače [10]

Obdobným řešením může být také použití většího počtu snímačů, například v kruhovém uspořádání. U tohoto uspořádání je třeba dát pozor na překrytí zorných polí jednotlivých snímačů, aby nedocházelo k chybným detekcím [25].

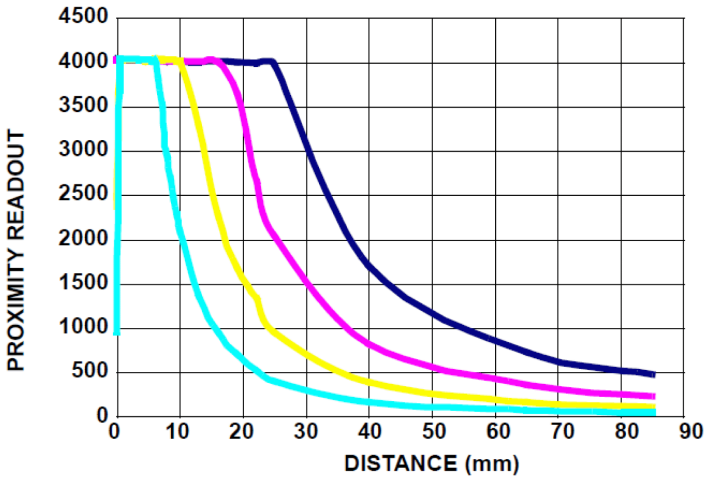
Běžně se ultrazvukové snímače používají i jako parkovací snímače u automobilů. V průmyslu pak pro měření výšky hladiny, přítomnosti či vzdálenosti objektů.

2.1.2 Infračervený dálkoměr

Podobně jako ultrazvukový snímač je infračervený dálkoměr velmi levný a kompaktní. Běžně používá velice zjednodušenou metodu pro měření, kde se měří pouze intenzita odraženého záření. Princip spočívá v tom, že čím je překážka blíže, tím více záření se odrazí zpět ke snímacímu prvku. Tato metoda se používá například v mobilních telefonech. Je velice jednoduchá, ale nepřesná, nevhodná pro detekci malých překážek a náchylná na příjem rušivého záření z okolí. Závislost výstupního

signálu na vzdálenosti překážky je nepřímo úměrná (graf 2.3). S rostoucí vzdáleností klesá hodnota výstupního signálu [17].

Výstupní signál je taktéž závislý na velikosti objektu či překážky, je-li objekt malý, je jeho vzdálenost vyhodnocena chybně.



Obr. 2.3: Charakteristika infračerveného snímače [17]

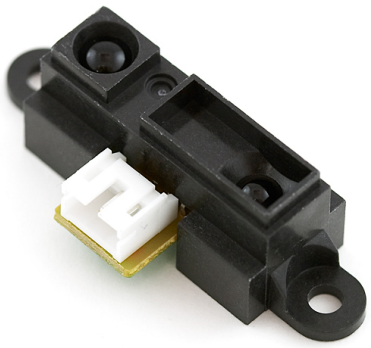
Na rozdíl od ultrazvuku, není tento snímač schopný pracovat pod přímým slunečním světlem, kdy dochází k saturaci fotocitlivého elementu. Ten pak není schopen rozeznat změny intenzity způsobené změnami vzdálenosti. Také může mít problém s detekcí čirých či průsvitných překážek.

Další nevýhodou oproti ultrazvuku je nutnost pracovat v čistém prostředí. Pokud by byl vysílací či snímací prvek znečištěn, došlo by ke zkreslení měřené hodnoty. Vzhledem k výše zmíněným nevýhodám je tento snímač předurčen především do vnitřního prostředí budov.

Tento typ snímače je nejlevnější a parametricky také nejhorší snímač vzdálenosti. Jeho použití pro robotiku je sice možné, ale jen ve vnitřním prostředí bez dopadu slunečního záření a to pouze pro detekci velkých překážek, třeba zdí.

2.1.3 LIDAR

LIDAR, z angl. light detection and ranging (také LiDAR či LADAR) je název pro způsob měření vzdálenosti, využívající měření doby letu světelného paprsku k cíli a zpět (TOF). Obvykle se realizuje pomocí laseru v infračerveném spektru a adekvátního snímacího prvku, například PIN diody [14]. Světelným zdrojem nemusí být nutně laser, používá se také laserových diod či rychlých infračervených LED. Pro



Obr. 2.4: Příklad infračerveného snímače [7]

přesné měření je ale nutné, aby vyzařovaný a snímací kužel byl co nejužší. Při širším vyzařovacím a snímacím úhlu by docházelo k příjmu paprsků z různých směrů a vzdáleností. To by způsobilo nepřesné měření a maximální použitelný dosah by byl velmi omezen.

Obr. 2.5: Lidar Lite [14]

Na uvedeném obrázku 2.5 je Lidar Lite firmy PulsedLight. Dle konfigurace má snímací úhel 3° a dosah dle prostředí až 60m. Pro měření TOF se obvykle používá metoda měření fáze odraženého paprsku. Aby bylo dosaženo vysoké přesnosti měření na zvolenou vzdálenost, musí světelný zdroj i přijímač pracovat na vysokém kmitočtu kvůli vysoké rychlosti světla. Jde řádově o jednotky až desítky MHz (dle vzorce 2.1).

$$\lambda = \frac{c}{f} \qquad [m, ms^{-1}, Hz] \tag{2.1}$$

Aby byl výsledek měření jednoznačný, musí se měřená vzdálenost nacházet v rozsahu $0 - \frac{\lambda}{2}$. Z čehož plyne, že pro měřící rozsah 10 m je potřeba kmitočet 15 MHz.

Laserový dálkoměr je na rozdíl od předchozích snímač bodový, tedy jeho zorný úhel je velmi malý. Při použití jediného snímače je tak k dispozici údaj o vzdálenosti jediného bodu. Právě tato vlastnost je u tohoto snímače unikátní. Je zřejmé, že použití jediného snímače by bylo nedostatečné, vhodné pouze k několika málo typům měření. Proto se v mobilní robotice používá především v modifikované verzi, jako 2D nebo 3D scanner.

2.1.4 Laserový scanner

Laserový scanner principiálně vylepšená aplikace laserového dálkoměru. Princip měření vzdálenosti je stejný. Laserový svazek je zde rozmítán pomocí pohyblivého zrcátka či hranolu. Výsledkem je snímač schopný měřit ve velmi širokém zorném poli, někdy dokonce celých 360°, a získávat informace o velkém množství bodů. Výstupní data jsou tedy souborem všech naměřených vzdáleností, obvykle v řádkové nebo maticové formě, někdy se také nazývají úhlová data.

Konstrukčně jednodušší je 2D scanner. Zde dochází k rozmítání laseru pomocí rotujícího zrcátka. Vhodně zvoleným vzorkováním měření vzdálenosti, vzhledem k poloze zrcátka, se docílí požadovaného rozlišení a velikosti zorného úhlu. U 3D scanneru musí docházet kromě rotačního pohybu také k vychylování v kolmém směru.

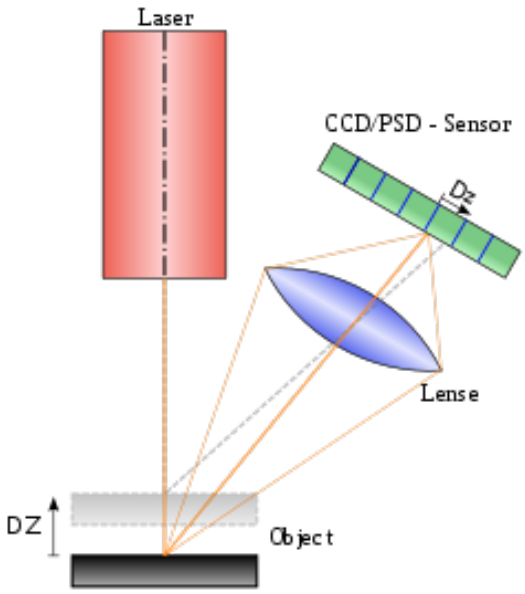
Obr. 2.6: Laserový scanner SICK [12]

Laserový scanner je již vcelku složitý snímač, poskytuje také výrazně více informací než předchozí snímače. Kvůli velkému množství poskytovaných dat je nutné pokročilejší a výkonnější komunikační rozhraní. Obvykle pracují bezproblémově i ve

venkovním prostředí i pod intenzivním slunečním zářením. Poměrně velkou nevýhodou těchto měřicích systémů je jejich vysoká cena.

2.1.5 Triangulační snímač

Pomocí laseru se dá vzdálenost měřit kromě TOF také triangulační metodou. Nevýhodou tohoto měření je nutnost oddělení vysílací a přijímací části. Čím dále jsou od sebe tím přesnější měření lze získat.



Obr. 2.7: Triangulační metoda [1]

Výsledná vzdálenost od přijímače se vypočte dle vzorce 2.2.

$$d = a \cdot tan(\alpha) \qquad [m, m, rad] \tag{2.2}$$

Kde a představuje vzdálenost vysílače a přijímače, α naměřený úhel od roviny s vysílačem a d je výsledná vzdálenost od vysílače. Místo bodového laseru může být použit lineární.V takovémto případě při použití adekvátního 2D snímače lze získávat přímo 2D strukturu povrchu.

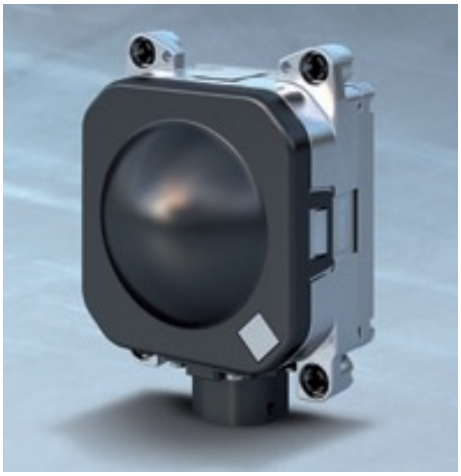
3D model pak lze získat například posuvem nebo rotací celého snímače. Další možností je využití strukturovaného světla na místo laseru. V obraze z kamery se pak vyhledávají známé struktury. Dle zjištěné deformace promítané struktury je následně možné vypočítat 3D model.

Nevýhodou triangulační metody je nutnost rozměrnější konstrukce a také tvorba tzv. stínů. Jedná se o místa, která jsou viditelná z pozice přijímače, ale ne již z pozice vysílače. Ve výsledném modelu informace o těchto místech chybí.

2.1.6 Radar

Radarový měřič vzdálenosti pracuje velmi podobně jako laserový (měření TOF). Rozdíl je především v použitém záření. Zatímco laser využívá vlnové délky v řádově nm až μm (infračervené spektrum) radar využívá mikrovlnné spektrum s vlnovými délkami v řádu cm.

Z výše popsaných snímačů má tato technologie nejvyšší dosah, díky čemuž se používá především v letectví. Další nespornou výhodou je schopnost pracovat i ve výrazně zhoršených optických podmínkách, jako je mlha či kouř. V poslední době se také začínají radarové snímače používat v automobilovém průmyslu, především jako brzdový asistent. V některých případech slouží i jako snímač pro následování vozidla.

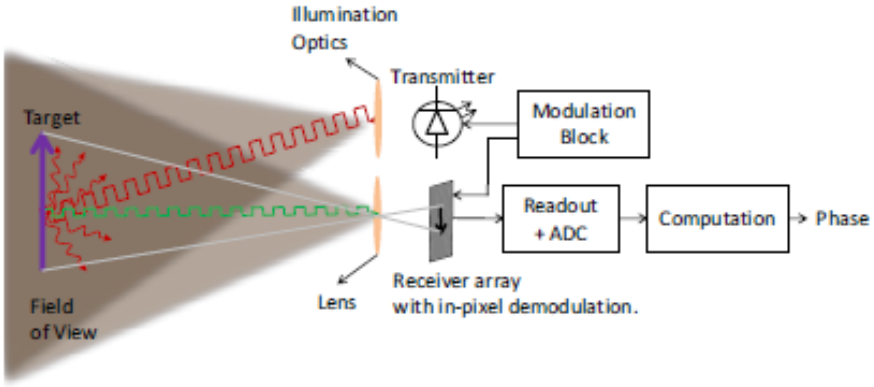


Obr. 2.8: Automobilový radarový snímač BOSCH [16]

Výhoda radaru je především ve vysokém dosahu, což se výborně hodí pro dopravní prostředky. Horizontální zorné pole těchto radarů bývá okolo 30°, což je pro silniční dopravu dostatečné. V robotice se tak radary začínají uplatňovat ve sféře autonomních vozidel.

2.1.7 TOF kamera

Dálkoměrná kamera využívající pro měření dobu letu paprsku. Jedná se o aktivní kameru, která vysílá směrem k překážce modulovaný infračervený signál. Kmitočet modulace je v řádu MHz (vzorec 2.1) a k vyslání signálu slouží sada výkonných infra LED. Kvůli zpracování číslicovými obvody je častěji používána binární modulace oproti spojité (harmonické). Pro výpočet vzdálenosti bodu se obvykle pracuje s vyhodnocováním fáze vyslaného a přijatého paprsku.



Obr. 2.9: TOF princip [23]

Zde je nutné, aby snímací úhel kamery byl celý pokryt vyzařovacím úhlem LED, a aby intenzita modulovaného světla byla vyšší než intenzita okolního osvětlení. Především nesmí dojít k saturaci snímacího prvku. I tak je pro správnou funkci nutné odfiltrovat vliv okolního osvětlení, a to jak fyzicky pomocí optických filtrů, tak číslicově při následných výpočtech [23].

Výstupem kamery jsou maticová data, podobně jako u černobílé kamery. Jednotlivé body ale místo intenzity odraženého světla udávají jeho vzdálenost. Při známých pozorovacích úhlech a rozlišení není problém z těchto dat vypočítat prostorové souřadnice každého bodu.

Nevýhodou snímače je neschopnost detekovat opticky čiré překážky a překážky s minimálním difúzním odrazem, jako jsou například zrcadlové či podobně vysoce lesklé plochy. Dále pak také omezená funkčnost při silném okolním osvětlení (záleží na výkonu vysílacího prvku).

2.1.8 Kinect for Windows

Kinect je dálkoměrné vyvinuté pro rozpoznávání pohybu. Jeho účelem je umožnit ovládání her pohyby vlastního těla. Původně byl Kinect vyráběn pro herní konzoli Xbox 360. Kinect for Windows je pak speciální verze se standardním USB konektorem, kterou vydal Microsoft pro uživatelský vývoj software. Je k němu k dispozici SDK, který je zdarma ke stažení. Bohužel co se týče podrobnějších technických informací, není Microsoft příliš sdílný a o použité technologii mnoho neprozrazuje. Technické informace tak byly získány pomocí „reverse engineering“ [9].

Starší verze původně pro Xbox 360 pracovala na principu triangulace. Scéna byla osvětlována strukturovaným infračerveným světlem o vlnové délce 830 nm a snímána infračervenou kamerou. Minimální vzdálenost kterou dokáže zpracovat je 0,7 m, ma-

ximální pak 6 m. Snímací úhly jsou 57° horizontálně a 43° vertikálně, při rozlišení 640x480 pixelů.

Obr. 2.10: Starší verze Kinect [9]

Novější typ pro Xbox One (obrázek 2.11) pracuje na úplně jiném principu a to TOF. Díky tomu je kompaktnější a údajně bylo dosaženo i výrazně lepších parametrů. Oficiální veřejně přístupné technické informace opět neexistují, výrobce uvádí vylepšení pouze kvalitativně. Větší pozorovací úhly, lepší rozlišení než předchozí model a připojení pomocí USB3.0.

Obr. 2.11: Kinect pro Xbox One [9]

Ve srovnání s předchozími maticovými snímači je Kinect díky masové produkci velmi levný (řádově levnější). Díky oficiálnímu SDK je vývoj software využívající toto zařízení značně usnadněn. V robotice se tento systém začíná experimentálně používat pro roboty pro vnitřní prostředí budov. Výhodou může být přítomnost běžné barevné kamery, což může v některých případech ušetřit nutnost instalace dalšího zařízení.

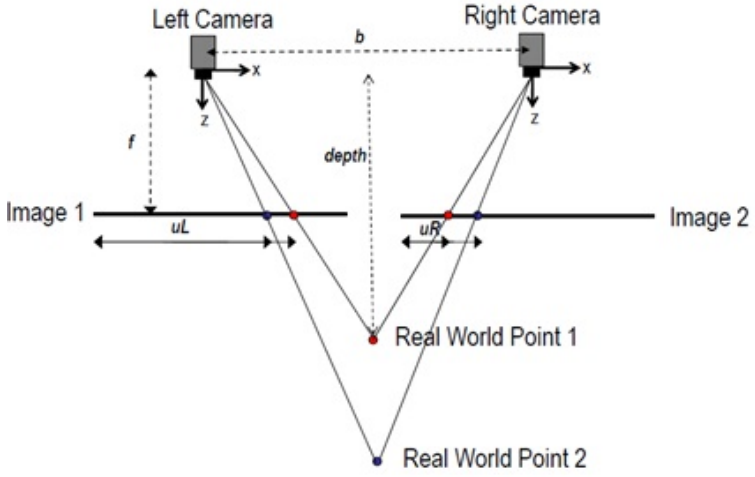
Nevýhodami obou verzí Kinectu jsou opět omezená funkčnost při vyšších intenzitách okolního osvětlení.

2.1.9 Stereovize

Stereovize se od předešlých snímačů značně odlišuje. Jednak tím, že se nejedná přímo o typ snímače, ale o metodiku získávání 3D dat pomocí dvojice kamer. Další

významný rozdíl je, že jde o čistě pasivní technologii kde nedochází k vysílání signálu směrem k překážce, ale pouze se snímá obraz scény.

Pro výpočet pozice bodu v prostoru stačí najít jeho pozici na snímcích z obou kamer. To se obvykle provádí pro opticky výrazné (významné) body s vysokým gradientem. Právě to je kamenem úrazu celého systému. Problém plyne z odlišného úhlu pohledu obou kamer, tudíž jejich obrazy nemusí obsahovat stejné významné body nebo nemusí být dostatečně opticky podobné. Problémy jsou i v celistvých plochách, kde se významné body nenacházejí, a tak není možné získat informaci o poloze v prostoru.



Obr. 2.12: Stereovize [5]

Největší problém této metody je poměrně velká výpočetní náročnost a zároveň také nevalná spolehlivost v reálných podmínkách. V robotice se tato metoda zatím využívá převážně experimentálně či společně s dalšími dálkoměrnými senzory.

2.2 Algoritmy

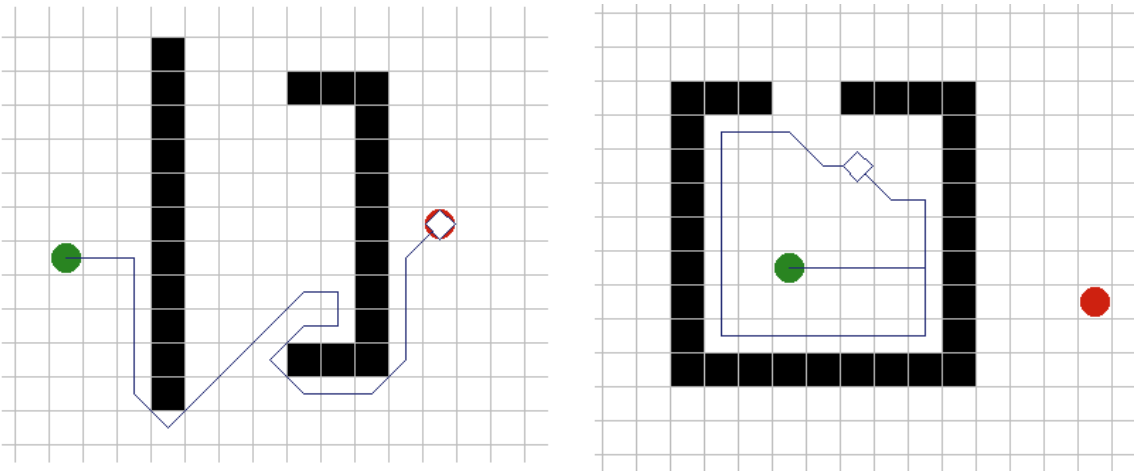
Výběr vhodného algoritmu má zcela zásadní význam. Algoritmů a hlavně jejich modifikací existuje nepřeberné množství. Liší se jak ve způsobu zpracování získaných informací, tak v dosažitelných výsledcích a spolehlivosti. V následujících odstavcích budou popsány ty nejznámější a nejpoužívanější.

Algoritmy lze rozdělit do dvou skupin. První skupina by se nechala nazvat jednoduché objíždění překážek. V této skupině jsou algoritmy, jenž si vytvářejí velmi jednoduchou nebo žádnou mapu okolí. Druhá skupina jsou algoritmy, které vyhledávají optimální trasu v mapě, která je úplně nebo jen částečně známá. Tyto algoritmy (grafové) jsou výrazně složitější, ale také mohou dosáhnout výrazně lepších výsledků.

2.2.1 Jednoduché metody objíždění překážek

Všechny tyto algoritmy pracují velmi podobným způsobem [2]. Postupují od startu směrem k cíli dokud nenarazí na překážku. Poté již není možné pokračovat správným směrem a dle strategie se zvolí náhradní směr.

Nejjednodušší je v tomto případě pohyb pevně zvoleným směrem, například obcházení zleva. Někdy se také nazývá těsné obcházení překážek (obrázek 2.13).



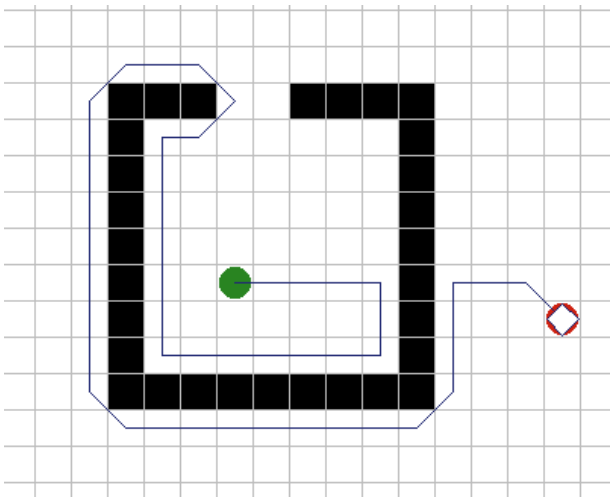
Obr. 2.13: Těsné obcházení překážek [2]

Další možností je při kontaktu z překážkou zvolit směr náhodně či pseudonáhodně. Tato strategie je velmi nenáročná, vhodná pouze pro obcházení malých překážek o velikosti řádově v jednotek kroků, jenž pro obejití nevyžadují více než jeden krok směrem zpět od překážky.

Vcelku použitelnou metodou je robustní obcházení překážek. Je podobná těsnému obcházení, ale obcházení je ukončeno až v okamžiku průchodu spojnicí mezi prvním uhnutím z trasy a cílovým bodem (obr. 2.14). Tato metoda již zvládá značně komplikované typy překážek. Co se týče délky trasy, mohou být výsledky značně neuspokojivé. Záleží zde především na očekávaném tvaru překážek.

2.2.2 Umělé neuronové sítě

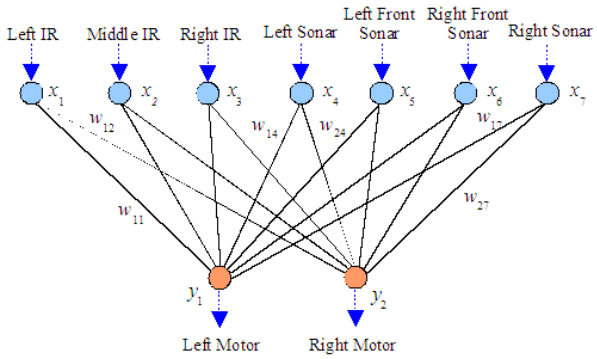
Umělé neuronové sítě mají tak široké uplatnění, že ani zde nemohou chybět. Algoritmy pracující s neuronovými sítěmi jsou poměrně časté, obvykle se řadí do první skupiny algoritmů (jednoduchých). Jejich nevýhodou je nutnost učení se na trénovacích datech. Po naučení však dokáží v celku spolehlivě pracovat v naučeném rozsahu



Obr. 2.14: Robustní obcházení překážek [2]

vstupních dat, mimo tento rozsah je pak chování systému v podstatě nepředvídatelné. Hlavní výhodou je, že není třeba přesná znalost systému robotu, ta se naučí z trénovacích dat.

V nejjednodušším případě můžeme požadovat pouze vyhnutí se kolizi. V tom případě může být UNS velmi jednoduchá, jednovrstvá s několika neurony (obr. 2.15). Postačí ji data z několika málo dálkoměrných senzorů a na výstupu ze sítě může být například údaj pro natočení kol podvozku [13]. Naučení sítě bude snadné a rychlé. Naučená síť si pak poradí s jednoduchým typem překážek (krabice), ale nemusí být schopna vyřešit situaci typu slepá ulice.



Obr. 2.15: Jednoduchá UNS pro vyhýbání se překážkám [13]

Pokud by bylo požadováno nalezení nejkratší možné trasy ve známém, nebo částečně známém prostředí, stává se problém mnohem složitější. Neuronová síť by v tomto případě měla mít i paměť, aby nedocházelo k zacyklení. Topologie takovéto

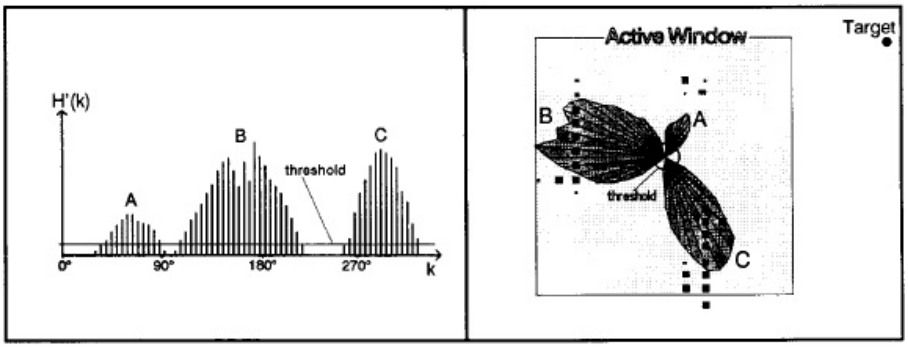
sítě by byla velmi složitá. Učení této sítě by pak bylo velice náročné a zdlouhavé. I dobře naučená síť bude pracovat správně jen v mezích naučených parametrů, mimo ně bude její reakce neznámá.

2.2.3 VFH

Vector field histogram je poměrně zastaralý algoritmus. Byl vytvořen pány Johnatan Borestein a Yoram Koren v roce 1991 [4]. Výhodou tohoto algoritmu je, že předpokládá na výstupu snímačů nejistotu a přítomnost šumu. Výpočetně i paměťově je velmi nenáročný (úměrně roku 1991).

Algoritmus vychází z metod VFF (Virtual Force Fields) oproti níž je výrazně méně náročná a řeší její nedostatky. Dokáže už vést skrz úzký průchod, například otevřenými dveřmi.

Princip algoritmu spočívá ve vytvoření polárního histogramu blízkého okolí (active window, obrázek 2.16). Vstupem pro výpočet polárního histogramu je mapa prostředí, obsahující pravděpodobnost výskytu překážky. Pro filtraci šumu je použita na výsledný histogram prahová funkce, jenž definitivně oddělí bezpečný směr pohybu od zakázaného.



Obr. 2.16: Princio funkece VFH algoritmu [4]

Dá se říci, že tato metoda stojí na mezi jednoduchých a složitých algoritmů. Implementace tohoto algoritmu je jednoduchá, pro dobrou funkčnost je zapotřebí snímat široké okolí, nejlépe celých 360°. Výsledný směr jízdy se vybere jako bezpečný směr, který je nejblíže přímému směru k cíli.

Nevýhodou je možná nestabilita při průjezdu úzkým koridorem. Algoritmus je schopen řešit pouze jednoduché typy překážek, v delší slepé ulici může dojít k zaseknutí či zacyklení.

2.2.4 A* algoritmus

Nejznámější algoritmus ze skupiny grafových algoritmů [2]. Pro optimální funkci je vyžadována znalost celé mapy. V případě, kdy existuje cesta od startu do cíle, algoritmus vždy nalezne nejkratší možnou variantu. Umožňuje volně ohodnocovat jednotlivá pole, například dle náročnosti terénu.

Metoda je značně univerzální a tak její výsledky i výpočetní náročnost záleží na heuristice hodnocení jednotlivých bodů. Tato metoda či její modifikace se používá i v počítačových hrách pro pohyb autonomních postav (NPC).

Algoritmus pracuje se dvěma seznamy bodů (či uzlů), prvním jsou otevřené body(open). Tento seznam obsahuje všechny body kterým již byla vypočtena jejich cena, ale ještě nepřišly na řadu. Každý otevřený bod obsahuje i pozici bodu z něhož byl otevřen. Pokud má otevřený bod ze všech nejnižší cenu (danou heuristikou), je přesunut do seznamu uzavřených bodů (closed). Jeho okolním bodům, které nejsou uzavřené, se vypočítá cena a přidají se do seznamu otevřených bodů. Je-li nový bod na nepřístupné pozici, je automaticky uzavřen a jeho cena se nastaví na nekonečně vysokou. Obdobně lze k vypočtené ceně připočítat cenu danou terénem, může být jak kladná tak záporná (zvýhodnění či penalizace). Hledání cesty končí dosažením cílového bodu, či vyčerpáním seznamu otevřených bodů. Výsledná optimální trasa se pak určí zpětně, z odkazů na pozici předchozího bodu, z něhož byl daný bod otevřen.

Existuje velké množství heuristik [2] pro výpočet ceny bodu. Výpočet ceny bodu (uzlu):

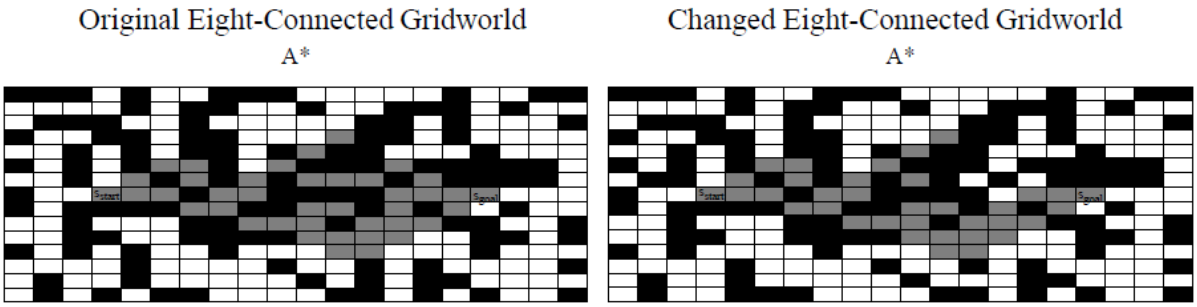
$$f(x) = g(x) + h(x) \tag{2.3}$$

g(x) je cena cesty od startu přes již uzavřené body, *h(x)* je odhad zbývající vzdálenosti do cíle, vypočítaný pomocí heuristiky. Aby byla heuristika přípustná, musí být její odhad vždy menší nebo stejný než skutečná vzdálenost do cíle. Přidáním násobku u *h(x)* můžeme získat nadhodnocenou heuristiku (nepřípustnou), jenž bude směřovat hledání více směrem k cíli. Tím může dojít ke zrychlení, ale v některých případech nemusí být cesta nalezena.

Běžné heuristiky:

- šachovnicová vzdálenost (maximová metrika, 8 směrů) ($h(x) = max(dx, dy)$)
- Manhattan vzdálenost (4 směry) ($h(x) = dx + dy$)
- Eukleidovská vzdálenost ($h(x) = \sqrt{dx^2 + dy^2}$)

Speciálními heuristikami jsou například Djisktrův algoritmus či Best-first. Djisktrův algoritmus odpovídá stavu, kdy $h(x) = 0$, je nevhodný z hlediska výpočetní náročnosti, jelikož většinou prohledává zbytečně velkou oblast. Best-first pak odpovídá $g(x) = 0$, což často způsobuje nalézání špatných cest [2].

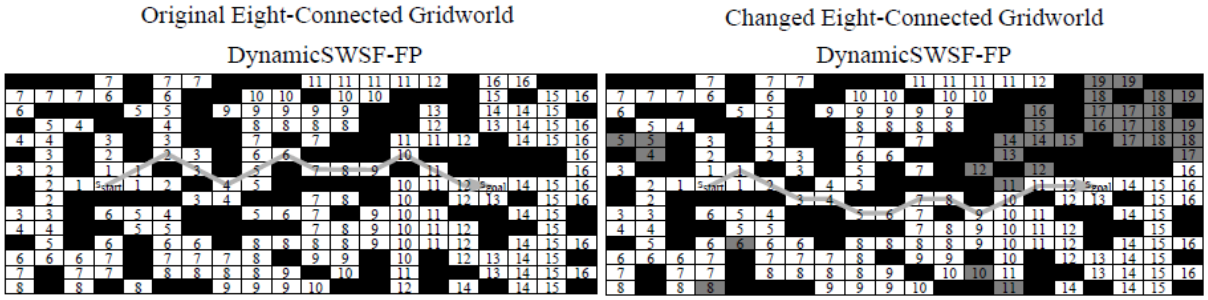


Obr. 2.17: Prohledávaná oblast při prvním hledání a při změně v mapě A* [11]

Nevýhoda algoritmu je poměrně velká výpočetní náročnost a počáteční zpoždění, způsobené nutností vypočítat celou trasu před zahájením pohybu. Pokud v mapě dojde ke změně, musí se přepočítat celá trasa.

2.2.5 IDynamicSWSF-FP algoritmus

DynamicSWSF-FP vypočítává a ukládá vzdálenost každého bodu od cílového. Což sice způsobuje vyšší zpoždění před startem, ale při změně v mapě se již nepřepočítává celá mapa, ale pouze body, jejichž hodnota se změnila.

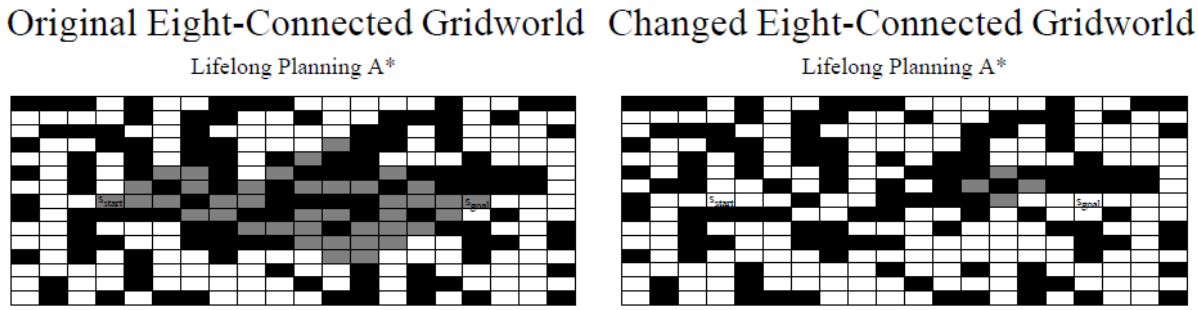


Obr. 2.18: Prohledávaná oblast při prvním hledání a při změně v mapě DynamicSWSF-FP [11]

Nevýhodné je, že přepočítává všechny body jejichž vzdálenost se změnila, což může být zbytečné. Hodí se pro hledání cest z různých výchozích pozic do jedné cílové.

2.2.6 Incremental A* algoritmus

Incremental A* [11], někdy také nazívaný Lifelong Planing A* (LPA*), vznikl jako modifikace A* a DynamicSWSF-FP za účelem urychlit výpočet při změně v mapě. Při prvním hledání pracuje naprosto stejně jako A*, při jakékoli změně mapy ale na rozdíl od DynamicSWSF-FP nepřepočítává všechny body, ale jen ty, jenž by mohly vést nejkratší cestou.



Obr. 2.19: Prohledávaná oblast při prvním hledání a při změně v mapě LPA* [11]

Při zablokování původní cesty v blízkosti startovní pozice je vyhledávání značně náročnější, protože počet polí nutných k přepočítání je výrazně vyšší. Pokud dojde k přesunu výchozí pozice, je nutné přepočítat celou trasu.

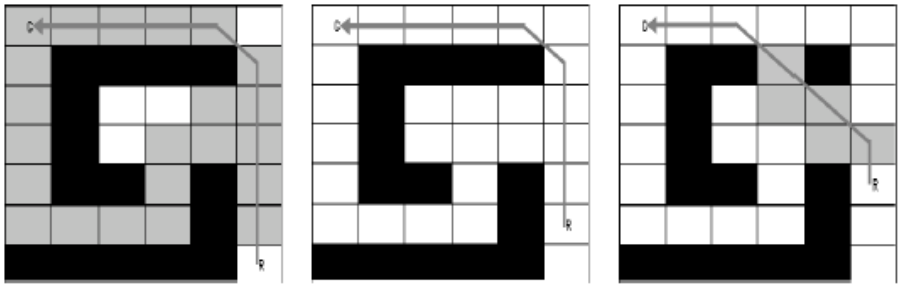
2.2.7 D* algoritmus

Nebo-li také Dynamic A*. Tento algoritmus vycházející z LPA*, opět předpokládá, že mapa k cíli je nepřesná nebo dokonce úplně neznámá.

Zaručuje, že najde nejkratší cestu z dostupných informací [2], nemusí tedy jít o skutečně nejkratší cestu. Prochází prostor v opačném pořadí než A*, tedy od cílového bodu směrem k aktuální pozici. Pokaždé když najde změnu oproti nepřesné nebo neúplné mapě, dojde k přepočítání trasy. K výpočtu ceny či délky trasy používá podobné heuristiky jako A* (je zde více možností). Hlavní výhodou oproti LPA* je, že se zde počítá s pohyblivým výchozím bodem, proto je tento algoritmus vhodný pro mobilní roboty.

D* Lite

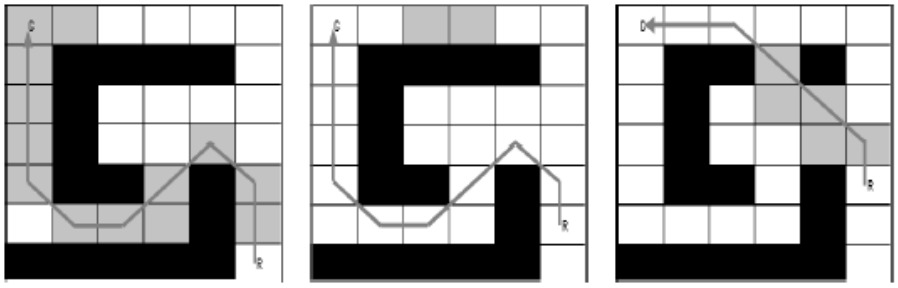
Podobně jako u A*, i zde existují další modifikace jako Focused D*, D* Lite či Field D*. D* Lite je modernější verzí Focused D*, je algoritmicky jednodušší, jinak má ale stejné vlastnosti. Co se týče výpočetní náročnosti je obvykle rychlejší, v nejhorším případě srovnatelný. Použití původního D* je tak již bezvýznamné a nevýhodné [15].



Obr. 2.20: Prohledávaná oblast D* Lite [15]

Anytime D*

Někdy také nazývaný Anytime Dynamic A*, je kombinace D* Lite s Anytime Replanning A* [15]. Je vhodný pro minimalizaci počátečního zpoždění, dokáže velmi rychle najít sub-optimální trasu, kterou postupně optimalizuje.



Obr. 2.21: Prohledávaná oblast Anytime Dynamic A* [15]

Field D*

Field D* se odlišuje především možností pracovat všemi směry, ne jen v obvyklých 4 nebo 8. Byl využit v praktickém nasazení na Mars Roveru [2].

2.2.8 Hierarchické plánování

Hierarchické hledání je další skupinou algoritmů. Mohou výrazným způsobem ušetřit výpočetní čas. Jejich princip je podobný lidskému uvažování, nejprve se cesta nalezne přibližně a poté se postupně zpřesňuje. Implementace těchto algoritmů je zpravidla složitější a výsledná trasa nemusí být optimální [2].

Výkonnost algoritmů je ovlivněna dalšími faktory, především rozdělením mapy na sektory, mezi nimiž se z počátku hledá cesta. Zvláště problematická je zde implementace neznámého prostředí.

3 POUŽITÝ ROBOT

Robot, pro který je systém bezkolizní navigace určen, je vývojová verze robotu Orpheus. Jedná se o průzkumného či záchranného robota, určeného i pro vojenské nasazení. Primárním účelem je průzkum člověku nebezpečných oblastí.

3.1 Popis konstrukce robotu

Konstrukce robotu je velmi jednoduchá a účelná. Podléhá především požadavkům na odolnost vůči nešetrnému zacházení i extrémním provozním podmínkám.

3.1.1 Podvozek

Základem robotu je čtyřkolový diferenciální podvozek (obrázek 3.1). Pro každé kolo je použit vlastní elektromotor, což umožňuje nezávislé řízení každého kola. Veškeré zatáčení se provádí v podstatě smykem vzniklým rozdílnou rychlostí kol levé a pravé strany. Tento způsob změny směru není příliš účinný, značně zvyšuje spotřebu energie a také opotřebení pneumatik (pouze na tvrdém povrchu). Výhodou tohoto podvozku je především robustnost a tuhost, způsobená absencí pohyblivých částí zatáčejícího mechanismu.

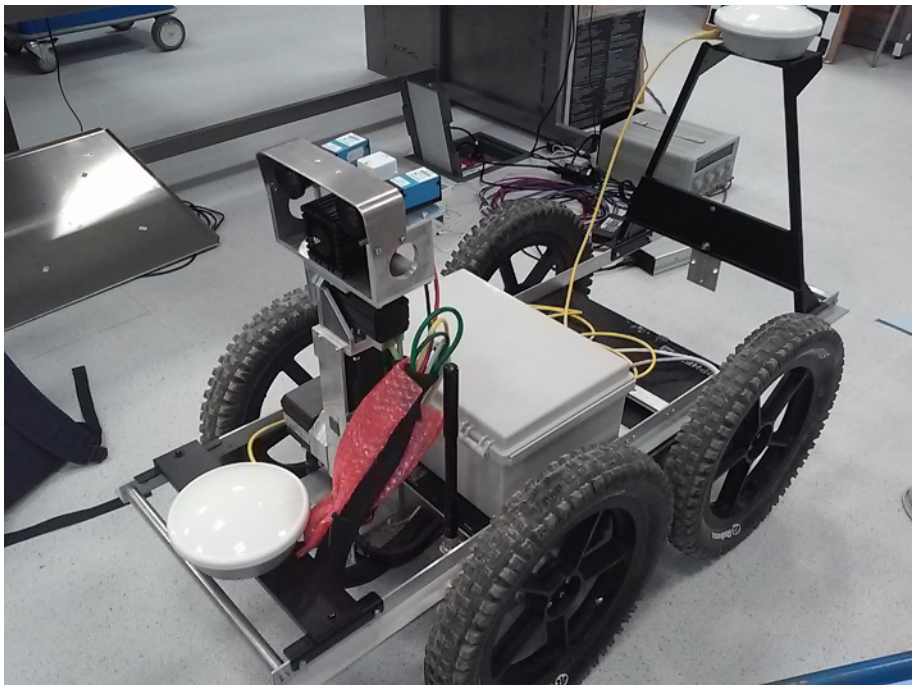
Další výhodou je také dobrá manévrovatelnost. Poloměr otáčení může být alespoň teoreticky nulový. V praxi je však tato možnost značně omezena výkonovými možnostmi pohonu. Při tomto typu zatáčení dochází k velkým proudovým odběrům a zatěžování motorů i akumulátorů.

3.1.2 Pohyblivé rameno

Na celém robotu je jen jedna pohyblivá část, tou je pohyblivé rameno pro umístění senzorů. Rameno má celkem 3 stupně volnosti. Celé rameno je sklopné, čímž je možné snížit celkovou výšku robotu, například kvůli nízkému podjezdu či pro skladnost při přepravě. Skladnost vývojové verze robotu je však omezena především montáží rámu pro GNSS antény.

Na vrcholu ramene je umístěna hlava, kterou je možné osadit řadou snímačů. Aktuální osazení je následující:

- stereo dvojice barevných kamer
- TOF kamera



Obr. 3.1: Fotografie vývojové verze robotu

3.2 Senzorové vybavení

Robot je vybaven celou škálou snímačů. Z hlediska zaměření práce není třeba popisovat vnitřní snímače, jenž slouží k měření interních veličin robotu (napětí baterií, odebírané proudy atp.). Popis tedy bude zaměřen pouze na snímače, umožňující získávat informace o okolním prostředí.

3.2.1 Výběr vhodného snímače pro detekci překážek

Vhodný snímač bylo nutné vybírat s ohledem na výslednou aplikaci robotu. Důležité jsou také konstrukční možnosti robotu, kam lze snímač umístit tak, aby byl schopen snímat potřebná data.

Vzhledem k faktu, že robot je již osazen pohyblivou hlavou osazenou snímači, bylo vhodné vybrat některý z nich. Pro detekci překážek zde byly instalovány dva vhodné snímače. První z nich je dvojce kamer pro stereovizi, druhá pak TOF kamera SR4000. Stereovize byla kvůli přílišné výpočetní náročnosti a nepříliš velké spolehlivosti zamítnuta. Nevýhoda zbývající TOF kamery je, že výrobce nedoporučuje její provoz ve venkovním prostředí na přímém slunečním svitu. Výrobce však již neuvádí, jaká konkrétní intenzita okolního osvětlení je pro správnou funkci kritická.

Pro návrh a testování algoritmu byla tedy zvolena již osazená TOF kamera. Výsledný algoritmus by neměl být pevně spjat se snímačem, z něhož data pochází,

měl by být pokud možno univerzální. Případné použití jiného snímače by tak nemělo být příliš komplikované. Týkalo by se pouze změny zpracování vstupních dat na mapu s vyznačenými překážkami.

Lepších výsledků by bylo možné dosáhnout použitím 3D laserového scanneru. Přineslo by to však další komplikace především v konstrukci robotu.

TOF kamera

Princip dálkoměrné TOF kamery byl popsán výše 2.1.7. Robot je vybaven kamerou Swiss Ranger 4000 od firmy MESA imaging. Jde o velmi kompaktní přístroj s vnějšími rozměry 65x65x76 mm [19]. Kamera je určena pro měření především ve vnitřních prostorách.

Obr. 3.2: TOF kamera SR4000 [19]

Technické parametry TOF kamery SR4000 [19]:

- vlnová délka 850 nm
- rozlišení 176x144
- pozorovací úhly (hxv) 43,6° x 34,6° nebo 69° x 56°
- pracovní teplota +10°C až +50°C
- modulační frekvence 29/30/31 MHz nebo 14.5/15/15.5 MHz
- dosah 0-5 m nebo 0-10 m
- přesnost +-1cm nebo +-1%
- rozhraní USB nebo Ethernet

Pozice kamery

Umístění kamery je pro detekci překážek velmi důležité. Optimální pozice je daná známými údaji o kameře a konstrukčními možnostmi robotu. Hlavní roli zde hrají úhly záběru kamery a její dosah.

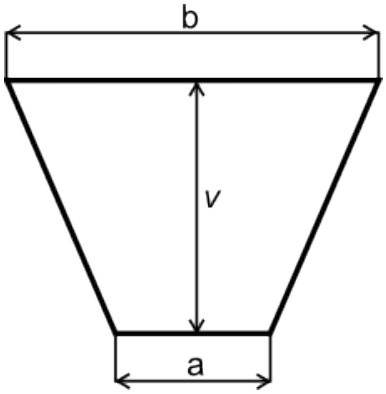
Při známé výšce umístění a úhlu natočení kamery je z těchto údajů možné, podle následujícího vzorce 3.1, určit předpokládané rozměry zorného pole. V případě, kdy je dosah kamery větší než je vzdálenost průsečíku horní hranice úhlu záběru a horizontální roviny, bude mít zorné pole tvar rovnoramenného lichoběžníku. V opačném případě bude mít vzdálenější strana částečně kruhový tvar o poloměru daném dosahem kamery.

$$v = h \cdot (tan(\frac{\pi}{2} - \alpha + \frac{\gamma}{2}) - tan(\frac{\pi}{2} - \alpha - \frac{\gamma}{2})) \tag{3.1}$$

$$a = 2 \cdot \frac{h}{cos(\frac{\pi}{2} - \alpha - \frac{\gamma}{2})} \cdot tan(\frac{\beta}{2}) \tag{3.2}$$

$$b = 2 \cdot \frac{h}{cos(\frac{\pi}{2} - \alpha + \frac{\gamma}{2})} \cdot tan(\frac{\beta}{2}) \tag{3.3}$$

Označení proměnných a, b a v odpovídá obrázku 3.3. Výška umístění kamery je označena h, úhel sklonu osy kamery směrem dolů je označen α. Horizontální úhel záběru je označen β, vertikální pak γ. Platí pro $\frac{\gamma}{2} < \alpha$.



Obr. 3.3: Zorné pole kamery

Je patrné, že výhodnější bude umístit kameru, co nejvýše konstrukce robotu dovolí. Získá se tím možnost vidět i za některé nižší překážky a získat tak o okolním prostoru více informací. Je tím značně zlepšena možnost vidět nejen vysoké překážky, ale i překážky typu díra, v níž by mohl robot zapadnout.

3.2.2 Lokalizace

Pro správný běh jakéhokoli mapovacího algoritmu je nezbytné znát polohu robotu i směr natočení. Bez těchto dat není možné do aktuální mapy správně zařadit nově získaná data. Na druhou stranu je někdy možné získat informaci o změně polohy,

porovnáním nových dat se staršími. Obvyklým přístupem je kombinace několika způsobů k zajištění vyšší přesnosti a spolehlivosti.

Snímače polohy můžeme rozdělit na relativní a absolutní. Relativní vychází z měření přímo na robotu, například měření otáček kol či měření úhlu natočení gyroskopem. Absolutní snímače měří pozici vůči nějakému pevnému bodu, patří sem satelitní navigace, různé typy navigací pomocí signálních majáků a také porovnávání zmapovaného okolí se známou mapou. Jednoznačnou nevýhodou relativních snímačů je jejich chyba rostoucí s časem či ujetou vzdáleností. Naopak výhodná je jejich jednoduchost a nezávislost na okolí. Přesný opak platí pro absolutní systémy (složité ale ustálená chyba).

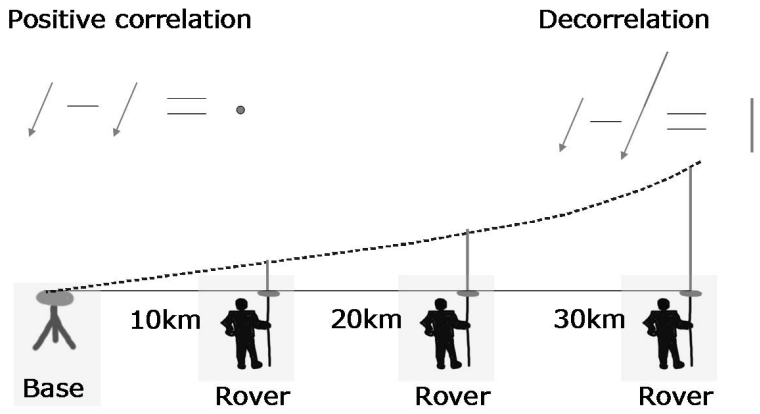
RTK GNSS

Na robotu je nainstalován RTK GNSS modul Trimble BD982, proto bude satelitní navigace primárním zdrojem polohových informací. RTK (Real Time Kinematis) je technologií diferenční GNSS navigace, někdy se nazývá real-time diferenční GNSS. Systém RTK je jakousi nadstavbou, umožňující zvýšit reálnou přesnost určení polohy na jednotky centimetrů až milimetrů. GNSS je souhrnné označení pro všechny současně používané satelitní polohové systémy, tedy americký GPS, ruský GLONASS a v budoucnu také evropský GALILEO a čínský Compass.

Vlastnosti GNSS přijímače Trimble BD982:

- Poziční anténa založená na 220 kanálovém Trimble MaxwellTM 6 čipu
- Vektorová anténa založená na druhém 220-channel Maxwell 6 čipu
- Nízkošumové měření fáze nosného GNSS signálu s <1 mm přesností v 1 Hz pásmu
- 1 USB port
- 1 CAN port
- 1 LAN Ethernet port:
- 4 × RS-232 porty
- 1 Hz, 2 Hz, 5 Hz, 10 Hz, 20, and 50 Hz poziční výstup
- Referenční výstup: CMR, CMR+TM, RTCM 2.1, 2.2, 2.3, 3.0, 3.1
- 1 × 1 Hz výstup

RTK systém se skládá minimálně ze dvou zařízení, z nichž jedno má pevnou, přesně známou pozici a nazývá se základna (base). Druhé zařízení je pohyblivé (rover) a jeho pozici chceme určit. Základna zajišťuje výpočet a vysílání korekčních hodnot v reálném čase. Na jednu základnu může být samozřejmě připojeno větší množství roverů. Maximální vzdálenost mezi stanicí a roverem může být 10-20 km. Tento dosah je dán především faktem, že korekční hodnoty jsou platné právě pro pozici základny a s rostoucí vzdáleností se jejich hodnota mění (obr. 3.4).



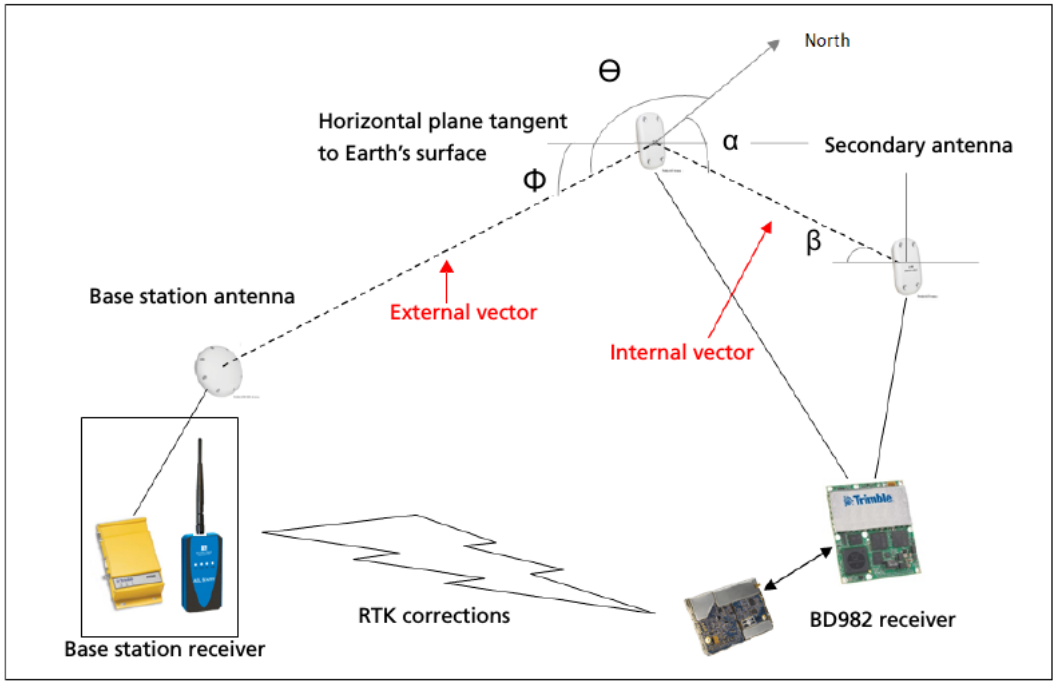
Obr. 3.4: Nárůst odchylky na vzdálenosti od základny [18]

Samotné určení přesné polohy spočívá v přesném měření fáze nosného kmitočtu signálů, přijímaných z dostupných satelitů. Tento proces je proti kódovému principu daleko náročnější na kvalitu přijímaného signálu. Pro úspěšnou inicializaci je potřeba signál z minimálně pěti satelitů jednoho systému (např. GPS), poté již stačí ze čtyř. Pro určení směru natočení robotu je vybaven GNSS přijímač hned dvojicí antén. Jedna se nazývá hlavní či poziční a druhá vektorová. Úhel natočení se následně vypočítá z rozdílu poloh obou těchto antén.

Odometrie

Metoda využívající odometrii k určení pozice se nazývá Dead reckoning. Odometrický modul by mohl sloužit jako krátkodobá náhrada polohových informací v případě ztráty satelitního signálu. Vzhledem k integrující se chybě by bylo použití na delší dobu (trasu) nevhodné.

V tomto případě by se opět jednalo o jakousi nadstavbu, jenž by v případě dostupnosti satelitních polohových dat pouze přeposílala získané souřadnice. V případě ztráty satelitního signálu by automaticky převzala kontrolu nad vysílanými polohovými daty, které by byly dopočítány pomocí odometrie. Funkce palubního počítače by tak krátkodobou ztrátou signálu nebyla nijak ovlivněna.



Obr. 3.5: Funkce RTK GNSS [24]

4 NÁVRH ALGORITMU

Existuje spousta algoritmů pro hledání cesty z jednoho bodu do druhého, zde je ale problematika odlišná. Předpokládaná činnost robotu bude asi následující. Robot bude řízen operátorem pomocí dálkového ovládání do cílové oblasti. Cestou se budou zaznamenávat GNSS souřadnice jimiž robot projel. Teprve po provedení průzkumu v oblasti se robot přepne do režimu automatického řízení pro návrat zpět. Tento návrat proběhne po zaznamenaných bodech trasy v opačném pořadí než byly pořízeny.

Nicméně na trase se v čase mezi jízdou do cílové oblasti a jízdou zpět mohly vyskytnout neočekávané překážky, které bude nutné objet. Po objetí se musí robot co nejdříve napojit zpět na plánovanou trasu. Cílem je maximálně se držet trasy pokud je to možné.

4.1 Požadavky

Požadavky na algoritmus jsou poněkud specifické. Úkolem algoritmu v případě, že na zvolené trajektorii nebudou detekovány žádné překážky, je nedělat vůbec nic a nechat jet robota přesně po zadaných bodech trasy. A to i v případě, kdy trasa bude zdánlivě značně neoptimální.

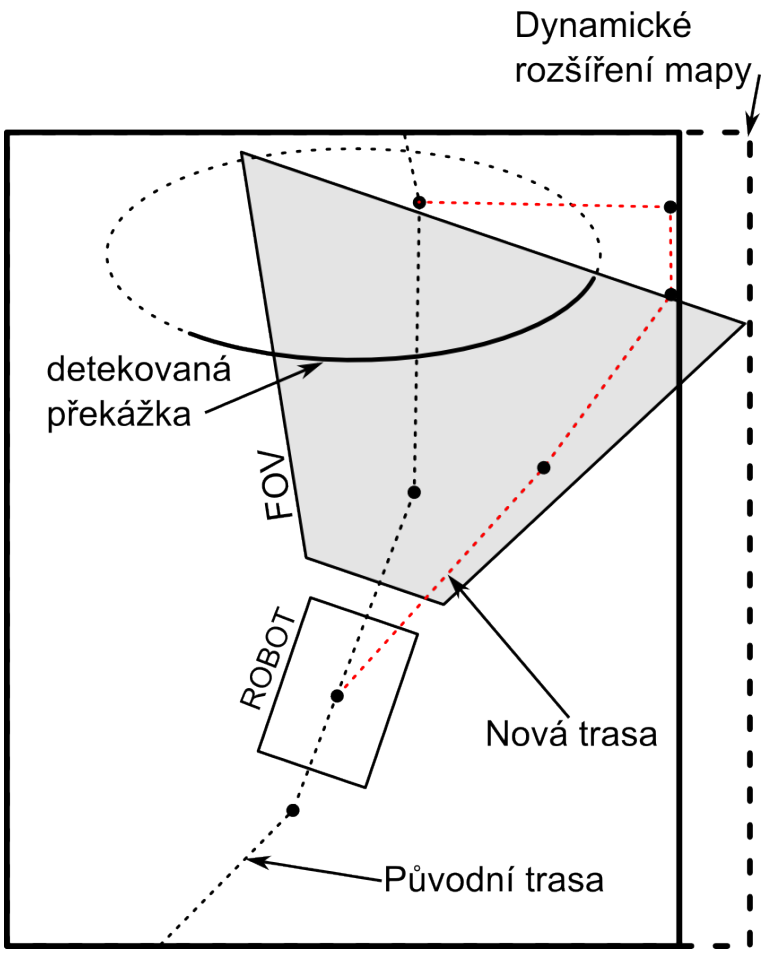
4.1.1 Mapování

Vzhledem k neznámému terénu je obvyklá volba reprezentace mapy čtvercová mřížka. Tento systém je implementačně jednodušší, na druhou stranu je také datově náročnější. Datová náročnost závisí především na velikosti celé mapy v poměru k velikosti jednotlivého čtverce. Budou-li jednotlivé čtverce velké, sníží se tím datová náročnost, ale i vypovídající přesnost samotné mapy. Je tedy vždy nutné zvolit jistý kompromis mezi datovou velikostí a přesností mapy.

Pro snížení objemu zpracovávaných dat je vhodné, aby velikost mapy byla dynamicky měněna v závislosti na požadavcích. Zjednodušeně řečeno je zbytečné vytvářet rozsáhlou mapu, pokud na známé části trasy neleží žádná překážka. Na druhou stranu pokud je detekována překážka, je zapotřebí začít vytvářet mapu na níž bude možné nalézt náhradní trasu. Tato mapa by měla být dynamicky rozšiřována, aby obsáhla všechen nově prozkoumaný prostor (na obr. 4.1).

4.1.2 Metodika obcházení překážek

Jelikož celý pohyb robotu je založen na navigaci mezi jednotlivými body trasy, je nutné vytvářet objízdnou trasu obdobným způsobem. Provede se nahrazení bodů



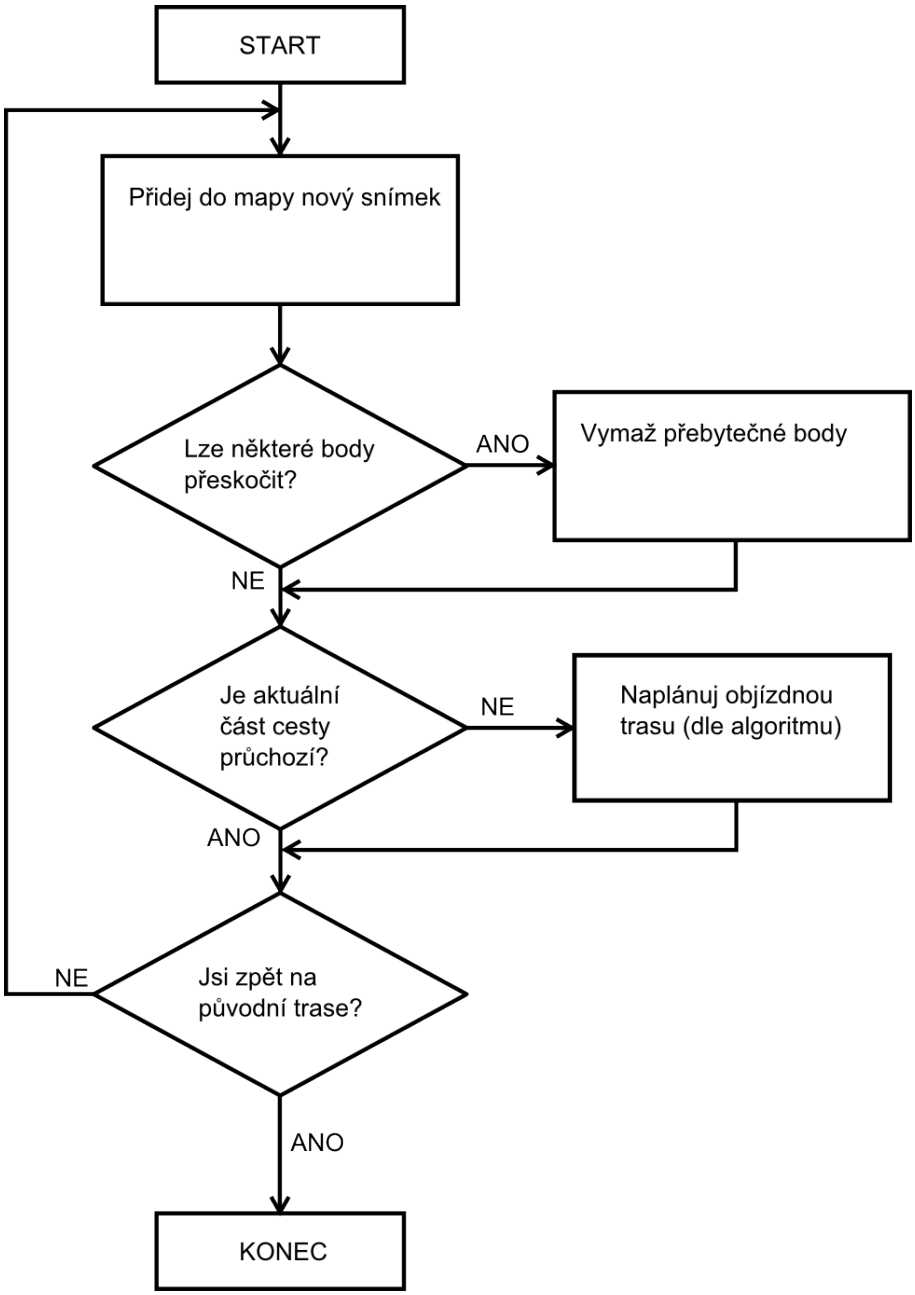
Obr. 4.1: Požadovaná metodika vyhýbání se překážkám

vedoucích ke kolizi novými body objízdné trasy (na obr. 4.1).

Metoda pro objíždění překážek se spustí až v případě, kdy dojde k detekci překážky na trase. Ukončení objíždění překážky je pak definováno návratem na původní trajektorii.

4.2 Popis plánovacího algoritmu

Celý algoritmus pro přeplánování trasy lze rozdělit do několika funkčních bloků. Jejich uspořádání je znázorněno na vývojovém diagramu 4.2.



Obr. 4.2: Vývojový algoritmus pro obcházení překážek

První částí je mapování okolí robotu a jeho převod na diskrétní mapu. To bude probíhat tak, že v určitých časových intervalech bude zaznamenán snímek popisující okolí robotu, například z dálkoměrné kamery. Tento snímek bude následně transformován na výškovou mapu. Na základě známé pozice robotu dojde k umístění snímku do vytvářené mapy.

Druhou částí je vynechání nadbytečných bodů. Tato část je důležitá z hlediska optimalizace trasy a může zajistit její značné zkrácení. V principu se jedná o přím-

kovou dostupnost některého z následujících bodů trasy. Je-li některý bod takto dostupný, pak není třeba projíždět body mezi ním a aktuální pozicí, ale lze zamířit přímo k danému bodu.

Třetí částí je kontrola blízké části trasy zda je bez nebezpečí kolize. Je-li cesta průchozí, nic se nemění. V opačném případě je ale nutné naplánovat objízdnou trasu dle metodiky popsané v odstavci 4.1.2.

4.2.1 Implementace algoritmu hledání cesty

Prvním problémem při implementaci jakéhokoli standardního algoritmu je zpracování trasy pomocí sady průjezdných bodů. Tento způsob není úplně obvyklý zejména proto, že vzdálenost jednotlivých bodů trasy nijak nesouvisí s použitou mapovou mřížkou. Tento problém lze vcelku snadno vyřešit správným zvolením výchozího a cílového bodu, mezi nimiž chceme nalézt trasu a následnou aplikací některého z běžných algoritmů pro nalezení optimální trasy. Získanou optimální trasu lze zpětně rozdělit na úseky, dle nějaké metriky, a tím získat požadované body trasy. Metrikou může být například konkrétní vzdálenost mezi jednotlivými body.

Další značně neobvyklou vlastností je nejen proměnná aktuální pozice, ale i cílová pozice. Zde se vlivem neznalosti prostředí může stát, že původně zamýšlený cílový bod se stane nedostupným a je tak nutno hledat cestu k následujícímu bodu. Samozřejmě i celá mapa je proměnná, včetně jejích rozměrů.

Výběr vhodného základního algoritmu

Z výše uvedeného je patrné, že mapa bude velice proměnlivá. Proto stojí za zvážení zda bude v takovémto prostředí výkon dynamických algoritmů pro hledání cesty znatelně lepší. Jakákoli změna prostředí způsobí u dynamicky pracujících algoritmů nutnost přepočítávat vyšší počet polí [22].

	searches per test case	moves per test case	expanded states per search	deleted states per search	runtime per search
A*	691	691	14489 (24.1)		6043
Di↑erential A*	691	691	14489 (24.1)		8385
GAA*	691	691	11246 (20.3)		4432
Basic MT-D* Lite	690	690	913 (5.2)		917
MT-D* Lite	690	690	535 (4.5)	550 (18.4)	570

$k = 1$

A*	697	697	14297 (23.6)		6006
Di↑erential A*	697	697	14297 (23.6)		8270
GAA*	697	697	11104 (19.9)		4401
Basic MT-D* Lite	697	697	920 (5.2)		920
MT-D* Lite	696	696	552 (4.4)	548 (18.1)	595

$k = 10$

A*	699	699	13740 (22.9)		5804
Di↑erential A*	699	699	13740 (22.9)		7956
GAA*	700	700	10429 (18.9)		4642
Basic MT-D* Lite	697	697	954 (5.3)		1054
MT-D* Lite	697	697	652 (4.6)	499 (16.1)	794

$k = 100$

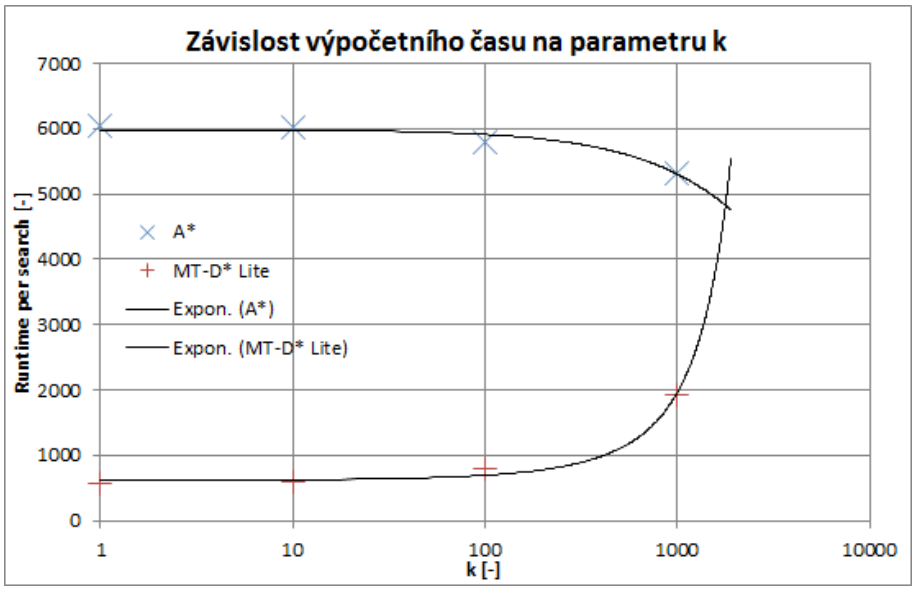
A*	707	707	12538 (21.1)		5311
Di↑erential A*	707	707	12538 (21.1)		7272
GAA*	709	709	9859 (81.8)		6607
Basic MT-D* Lite	695	694	1578 (6.9)		2129
MT-D* Lite	693	693	1426 (6.8)	352 (11.2)	1932

$k = 1000$

Obr. 4.3: Porovnání výkonnosti algoritmů v proměnném prostředí [22]

Z porovnávání je zřejmé, že pro málo proměnné prostředí dosahují dynamické algoritmy v dané úloze až desetinásobně rychlejšího nalezení cesty. S rostoucím množstvím změn v poli pak jejich výkonnost klesá až na pouhý 2,7 násobek statického algoritmu A*.

Nutno podotknout, že v porovnání (obr. 4.3) byla použita mapa o velikosti 1000x1000 bodů. Hodnota k pak určuje počet náhodných otevřených bodů, které budou uzavřeny a stejný počet uzavřených, jenž budou otevřeny v každém kroku. Z toho plyne, že maximální změna mezi jednotlivými hledáními (kroky) byla pouhých 0,2 %. Z tabulek lze odhadnout, že v případě výrazných změn mezi jednotlivými kroky, například v řádu jednotek procent, by se mohl stát dynamický algoritmus srovnatelným či dokonce nevýhodnějším (dle grafu 4.4). Delší výpočetní čas dynamických algoritmů na prohledání jednoho pole je způsoben složitostí samotného algoritmu.



Obr. 4.4: Extrapolace hodnot naměřených v [22]

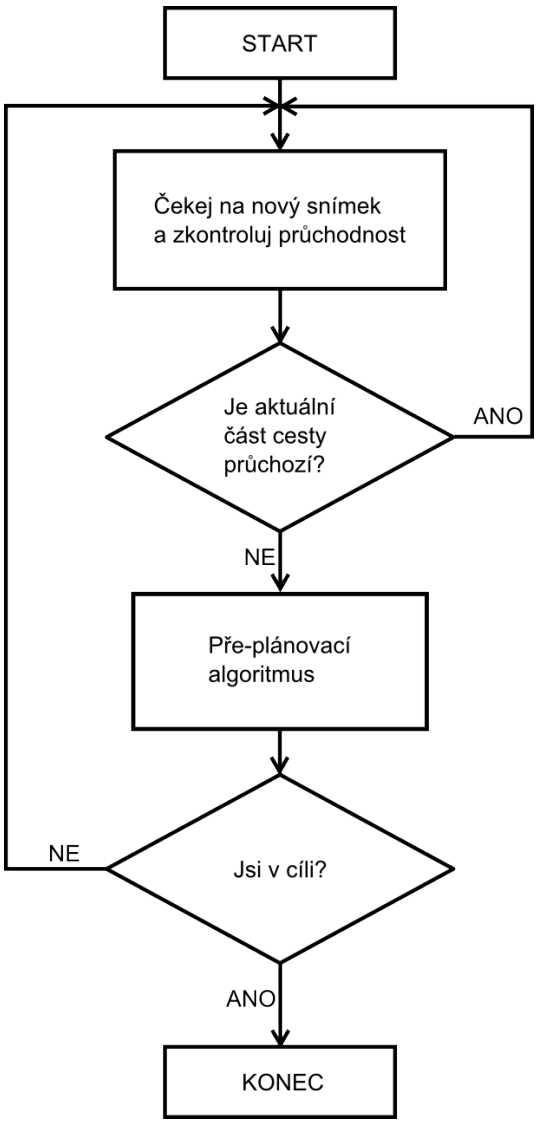
Situaci v grafu 4.4 nelze samozřejmě brát jako přesný příklad. Dosažené výsledky jsou sice přesné, ale jedná se o syntetické testy navržené pro snadné porovnání. Výsledky z reálného prostředí se proto mohou značně lišit. Zejména vlivem neznalosti celé mapy, jejíž velikost je navíc proměnná.

Vzhledem k výše uvedenému porovnání a porovnání složitosti implementace jednotlivých algoritmů bylo rozhodnuto využít jako základ vlastního řešení algoritmus A*. Jedná se sice o statický algoritmus neumožňující dodatečnou změnu trasy, ale jeho implementace je na rozdíl od dynamických algoritmů výrazně jednodušší.

4.3 Kompletní algoritmus

Kompletní algoritmus se skládá ze dvou základních bloků (obr. 4.5). Obsáhlejší blok je nazván pře-plánovací a byl popsán výše 4.2. Druhý blok je výrazně jednodušší. Jedná se o jednoduchou kontrolu průchodnosti cesty dle aktuálního snímku okolního prostředí. Následuje podmínka kdy je potřeba spustit plánovací blok.

V případě kdy je detekována překážka, je spuštěn pře-plánovací algoritmus. Ten se následně stará o celé objetí překážky. Teprve po dokončení objíždění a návratu na původní trasu dojde ke kontrole dosažení cílového bodu a případnému návratu na začátek smyčky.



Obr. 4.5: Vývojový diagram kompletního algoritmu

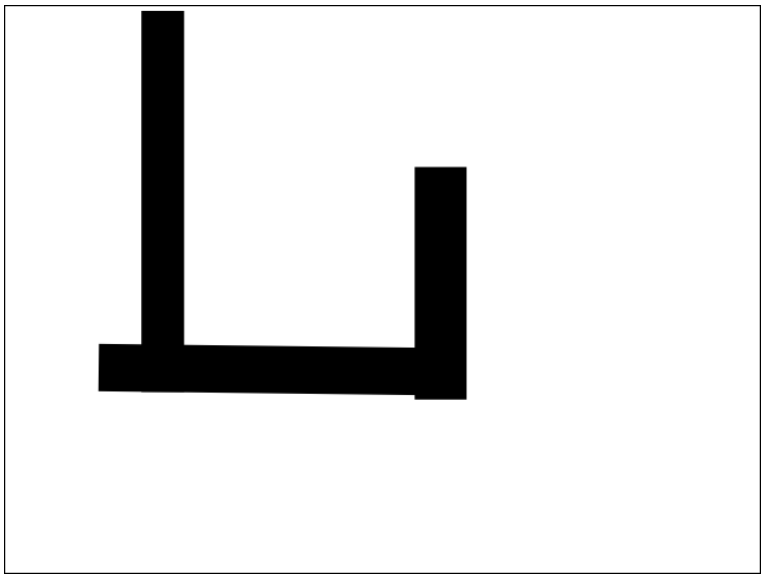
5 SIMULACE

Pro vývoj a testování metodiky detekce překážek a přeplánování trajektorie byla vytvořena simulace. Pro její realizaci bylo zvoleno prostředí programu MATLAB. Výhodou tohoto prostředí je velmi snadná implementace algoritmů. Nevýhodou je pomalejší zpracování dat.

5.1 Mapa

Pro jednoduchost je mapa tvořena černobílým či šedotónovým obrázkem. Překážky jsou vyznačeny černou barvou a pozadí je bílé. Rozlišení mapy je libovolné, ale se zvětšující se mapou také rostou požadavky na výpočetní čas a paměťová náročnost. Rozměrově je simulace navržena přibližně tak, že 1 pixel obrazu odpovídá 1cm, všechny rozměry se tak udávají v této jednotce. Rozměry robotu jsou tedy řádově v desítkách pixelů (50x60). Testovací mapy pak mají shodná rozlišení 640x480 pixelů.

Pro účely testování byl vytvořen soubor deseti různých map s překážkami navrženými pro otestování výkonnosti a spolehlivosti algoritmů.



Obr. 5.1: Příklad mapy pro testování

Složka mapy obsahuje 10 předpřipravených map, lze je volit zadáním jejich čísla v hlavním souboru *Simulation.m*.

5.2 Reprezentace robotu

Samotný robot je v simulaci reprezentován prostým obdélníkem. O jeho zobrazování se stará funkce *robPlace.m*, jenž obrys robotu vykreslí přímo do zadaného grafu. Pozice robotu se udává v řádkové matici 5.1.

Pozice robotu:

$$[x, y, \alpha] \tag{5.1}$$

Kde x udává vzdálenost od levého okraje, y od horního a α úhel natočení v radiánech. Vzdálenosti jsou v pixelech a vztahují se ke středu robotu, úhel 0° odpovídá směru svisle dolů.

5.3 Body trasy

Pro testování různých schopností algoritmu je vhodné mít možnost rychle umístit body požadované trajektorie. K tomuto se využívá integrované funkce *ginput*. Tato funkce umožňuje snadné získávání souřadnic přímo klikáním na mapu. Počet bodů trasy je možné volit proměnnou *n_wp* v hlavním souboru.

Pro kvantitativní porovnávání jednotlivých algoritmů by bylo vhodnější načítání pevně dané trasy, například z externího souboru. A zároveň provést s tím související měření výpočetního času. Tato možnost ale nebyla implementována.

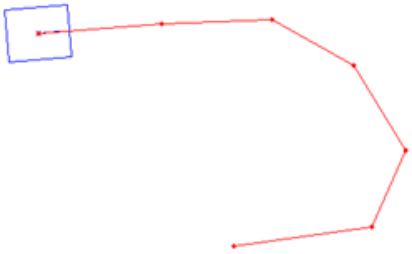
5.4 Pohyb robotu

Pro simulaci byl pohyb robotu maximálně zjednodušen. Skládá se pouze ze dvou operací, které nelze vykonávat současně. Buď robot stojí na místě a otáčí se směrem k následujícímu bodu trasy nebo je k tomuto bodu přímo natočen a po krocích se k němu přibližuje. Velikost kroku při natáčení i při posunu je možné upravit změnou parametru v souboru *stepXZ.m*. Velikost posunu uvádí vzdálenost v pixelech, o které se robot posune v každém kroku. Velikost úhlu pak udává maximální úhel, o který se v jenom kroku robot otočí.

Dosáhne-li požadovaného bodu, dojde k přepnutí na další. Pokud jsou již všechny body vyčerpány simulace končí.

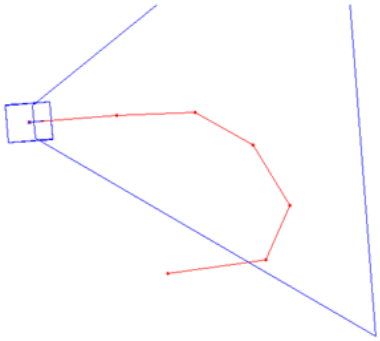
5.5 Detekce překážek

Správná detekce překážek je pro jejich objíždění naprosto klíčová. Pro maximální věrohodnost je třeba, aby robot "viděl"jen tu část mapy, kterou vzhledem ke své pozici může vidět. Zbytek mapy mu musí zůstat neznámý.



Obr. 5.2: Pohyb robotu po bodech trasy

Jelikož reálná TOF kamera či jakýkoli jiný snímač mají omezený dosah, je vhodné kameru natočit tak, aby její zorné pole nebylo tímto dosahem omezené. Potřebný úhel natočení je dán výškou umístění kamery, jejími pozorovacími úhly a maximálním dosahem.



Obr. 5.3: Zorné pole dálkoměrné kamery

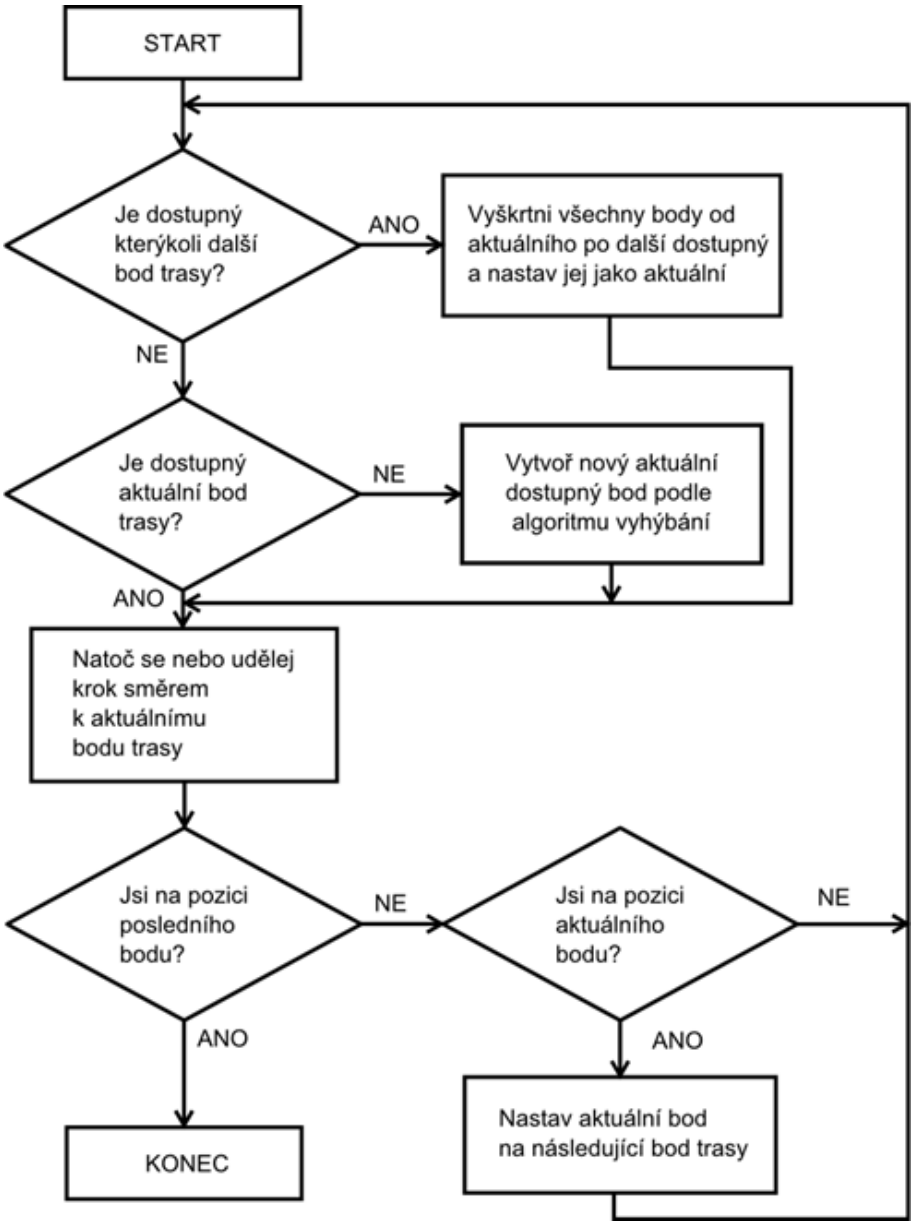
Pozorovací úhly kamery jsou také volitelné (v souboru *fovPlace.m*) a jsou nastaveny tak, aby odpovídaly skutečnosti, tedy 69° horizontálně a 56° vertikálně. Další potřebný údaj je výška umístění kamery na robotu, ta byla změřena na cca 65cm. Nastavitelný je také dopředný posun kamery vzhledem ke středu robotu, umožňuje tak nastavit umístění kamery v přední či zadní části robotu. Z těchto údajů a známé pozice robotu je již možné promítnout do mapy skutečné zorné pole. Vlivem perspektivní projekce se tedy jedná o rovnoramenný lichoběžník.

Pro detekci překážek se provede výřez mapy podle zorného pole a následně jeho zpětná perspektivní transformace. Po transformaci tak máme k dispozici obdélník či čtverec zahrnující celou plochu zorného pole. Rozlišení tohoto čtverce je opět volitelné (v souboru *obstacle_detection.m*). Vyšší rozlišení znamená značně vyšší

výpočetní čas v každém kroku. Na tomto obraze se provede v každém řádku pro každý pixel hledání levého minima, to znamená, že v každém bodě je minimální hodnota z bodů nalevo od něj. Tím se zajistí, že robot neuvidí překážky umístěné za sebou. Následně se provede hranová detekce ve směru pozorování a pak zpětná transformace. Tím se získá obraz viditelných překážek (jejich hran).

5.6 Přeplánování trajektorie

Celá metodika změn trajektorie se skládá ze tří částí a je zakreslena na vývojovém diagramu 5.4. První se stará o odsun blízkých bodů do dostatečné vzdálenosti od překážky. Jedná se pouze o body, které sice leží mimo překážku, ale leží stále příliš blízko, než aby jimi mohl robot bezpečně projet. Jako blízké body jsou označeny všechny následující body trajektorie, jejichž vzdálenost po trase je nižší než nastavený limit. Limit lze upravit v souboru *obstacle_avoidance.m*. Vybrané body se pak přesouvají ve směru gradientu vzdálenosti od překážky, dokud nejsou dostatečně daleko nebo dokud nejsou v lokálním minimu.



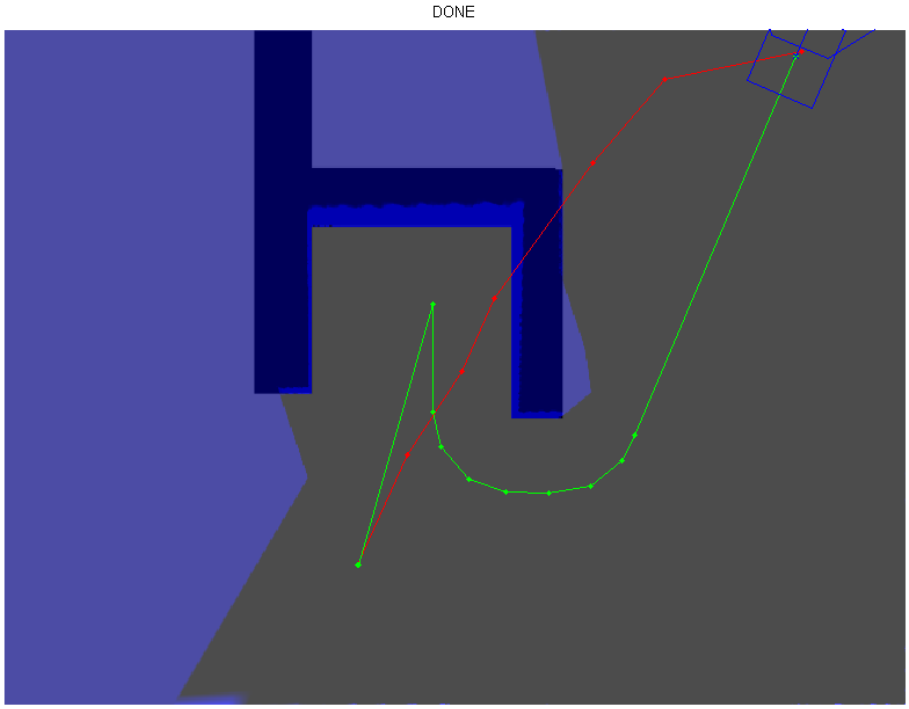
Obr. 5.4: Vývojový diagram simulace

Druhá část metodiky je vynechávání nadbytečných bodů. Existuje-li blízký bod, jenž se nachází v zorném poli robotu a je dostupný po přímce z aktuální pozice, pak se všechny body mezi ním a aktuální pozicí vyřadí. Aktuální pozice se pak přidá mezi projeté body a jako následující bod se nastaví právě zjištěný dostupný bod.

Poslední část je pak samotné obcházení překážek. K této části dochází pouze v případě, že následující bod trasy není dostupný, tedy leží mezi ním a aktuální pozicí překážky. Přeplánování trajektorie probíhá podle Dijkstrovy metody. Tato metoda také bývá nazývána záplavová. Tuto realizaci zajišťuje funkce *obstacle_avoidance.m*.

Na obrázku 5.5 je vidět výsledek přeplánování trasy. Černě jsou zde vyznačeny

překážky. Sytě modře jsou vyznačeny detekované překážky. Světle modrá zobrazuje neprozkoumané území, šedá pak již známé území bez překážek. Uživatelem zadaná trajektorie je vyznačena červenou čarou, zeleně je vyznačena přeplánovaná trajektorie.



Obr. 5.5: Přeplánovaná trajektorie

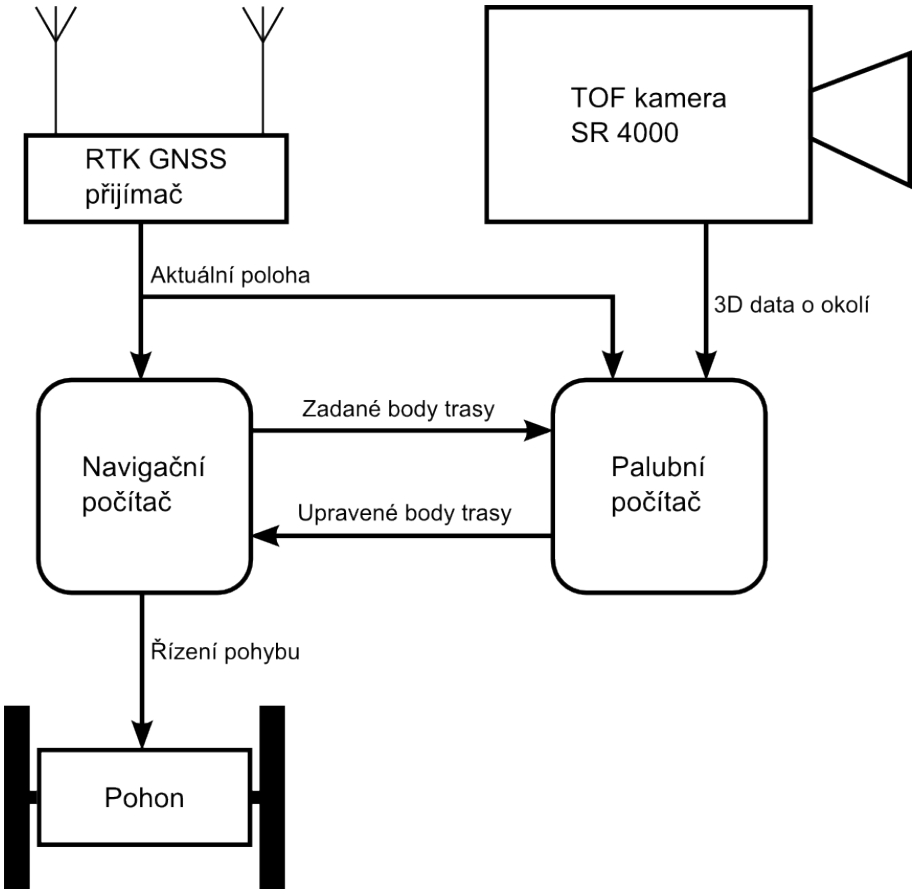
Bohužel výpočet Dijkstrovy vzdálenosti je poměrně zdlouhavý, zejména při větších vzdálenostech mezi aktuální pozicí a nedostupným bodem. Zaručuje nám však, že pokud je to možné, robot cestu do cíle najde a z dostupných informací to bude ta nejkratší varianta.

Zároveň přeplánování trasy pouze při nedostupnosti následujícího bodu se jeví jako nedostatečné, při kontrole delší trasy by často mohla být výsledná cesta výrazně kratší a s menším počtem zatáček.

6 REALIZACE

Realizace celého systému pro přeplánování trajektorie byla nejpracnější část práce. Z důvodu kompatibility s ostatními programy a na doporučení vedoucího práce byl zvolen programovací jazyk C#. Jelikož se jedná o jazyk vyvinutý a podporovaný firmou Microsoft, bylo jako vývojové prostředí zvoleno Visual Studio 2012, jenž je vyvíjené stejnou firmou.

Celý program je koncipován tak, aby sloužil jako nadstavba stávajících systémů robotu. Tedy jeho vyřazení by nemělo ovlivnit žádné další funkce. Program poběží v interním počítači robotu a s ostatními moduly bude komunikovat skrze jeho síťové rozhraní. Náčrt virtuálního propojení a datových toků mezi jednotlivými moduly je na obrázku 6.1.



Obr. 6.1: Zakreslení datových toků mezi moduly

Zapojení na obrázku je pouze ilustrační a zobrazuje směry toků informací. Všechny moduly budou navzájem propojeny ethernetovou sítí, skrze běžný ethernet switch uvnitř robotu. Jednotlivé komunikační protokoly a funkce palubního počítače budou popsány v následujících odstavcích. Popis navigačního počítače a způsobu řízení

pohonu by byl již nad rámec této práce.

6.1 Informace o poloze

Jak již bylo psáno (v odstavci 3.2.2) primárním zdrojem polohových dat bude RTK GNSS přijímač. Komunikace s GNSS přijímačem bude v zásadě jednostranná. GNSS přijímač bude vysílat data požadovaným způsobem a palubní počítač je bude zpracovávat.

6.1.1 NMEA-0183

Použitý RTK GNSS přijímač Trimble BD982 umožňuje zvolit si v jakém formátu a jak často má data vysílat. S vedoucím práce jsme se dohodli na použití formátu NMEA-0183. Jedná se o textový formát, jenž je jedním ze standardizovaných formátů pro přenos polohových informací. Výhodou tohoto formátu je, že je velmi snadno aplikovatelný na různá rozhraní. Drobnou nevýhodou je trochu náročnější zpracování dat jak pro vysílač, tak pro přijímač.

Všechny zprávy v tomto formátu začínají znakem $ [24]. Konec zprávy je pak realizován znakem pro ukončení řádku a návratem na začátek řádku(CR+LF). Jednotlivé datové hodnoty jsou odděleny čárkou (,) včetně prázdných polí. Na konci pole hodnot následuje znak * a po něm dva znaky kontrolního součtu.

Z celé sady zpráv, které tento protokol umožňuje, budou využívány pouze dvě. První z nich je zpráva GGA. Tato zpráva obsahuje především přesné zeměpisné souřadnice GNSS přijímače. Příklad zprávy může vypadat následovně:

```
$GPGGA,172814.0,3723.46587704,N,12202.26957864,W,2,6,1.2,18.893,M,
-25.669,M,2.0,0031*4F
```

Jednotlivé části oddělené čárkou mají následující význam:

Pole	Význam
0	ID Zprávy $GPGGA
1	UTC čas zjištění pozice
2	Zeměpisná šířka (Latitude)
3	Směr zeměpisné šířky: N: Severně, S: Jižně
4	Zeměpisná délka (Longitude)
5	Směr zeměpisné délky: E: Východně, W: Západně
6	GPS ukazatel kvality: 0-5
7	Počet používaných SV, rozsah od 00 do 12
8	HDOP
9	Orthometrická výška (zdroj MSL)
10	M: jednotkou pro orthometrickou výšku jsou metry
11	Výška nad geoidem
12	M: jednotkou pro výšku nad geoidem výšku jsou metry
13	Stáří diferenčního GPS záznamu, Typ 1 nebo Typ 9. Prázdné pole pokud není použito DGPS
14	ID referenční stanice, rozsah od 0000 do 1023. Prázdné pole pokud žádné ID referenční stanice není zvoleno a žádné korekce nebyly přijaty
15	Kontrolní součet dat, vždy začíná *

Druhý typ zprávy, který je třeba dekódovat je PTNL,AVR. Tato zpráva obsahuje údaje o přesném natočení přijímače, respektive vzájemné natočení jeho antén. Vzhledem k počtu dvou antén je jasné, že známé úhly natočení mohou být jen dva. První je úhel natočení (Yaw), který ukazuje horizontální odklon od severního směru. Druhý úhel je úhel sklonu, který popisuje vertikální odklon od horizontální roviny.

Příklad PTNL,AVR zprávy:

`$PTNL,AVR,181059.6,+149.4688,Yaw,+0.0134,Tilt„,60.191,3,2.5,6*00`

Pole	Význam
0	ID Zprávy $PTNL,AVR
1	UTC čas zjištění pozice
2	Úhel natočení ve stupních
3	Yaw
4	Úhel sklonu ve stupních
5	Tilt
6	Vyhrazeno
7	Vyhrazeno
8	Rozsah v metrech
9	GPS ukazatel kvality 0-4
10	PDOP
11	Počet satelitů použitý k řešení
12	Kontrolní součet dat, vždy začíná *

6.1.2 Zpracování polohových dat

Polohová data je nutné nejprve přijmout a následně dekódovat. Za tímto účelem byly navrženy třídy `com` a `GNSS_com`.

Třída com

Pro obecnou komunikaci byla vytvořena třída `com`, která zajišťuje záznam odeslaných a přijatých zpráv. Obsahuje také metody pro výpočet a kontrolu kontrolních součtů. Třída obsahuje pouze dvě proměnné, první je `logText`, což je textový řetězec obsahující záznam komunikace. Zbývající proměnná je pouze horní limit délky logu a slouží k zamezení hromadění přílišného množství dat.

Byly ještě přidány dvě metody pro oboustranný převod stupňů typu `double` na textový řetězec ve formátu DDMM.MMMMMM pro zeměpisnou šířku, respektive DDDMM.MMMMMM pro zeměpisnou délku.

Třída GNSS_com

Třídu `com` dědí další třída `GNSS_com`. Jejím prvořadým úkolem je zajistit komunikaci přímo s GNSS přijímačem. Proto obsahuje instanci třídy `UdpClient`. Neméně důležitá je také funkce dekódování přijatých zpráv a ukládání přijatých informací o poloze do vlastních proměnných.

Třída obsahuje následující proměnné, z nichž každá reprezentuje poslední známou informaci o dané veličině:

- `private double lastLatitude`

- `private double lastLongitude`
- `private double lastAltitude`
- `private double lastAzimuth`

Tyto proměnné jsou zvenčí přístupné pouze pomocí následujících metod:

- `public double getLatitude()`
- `public double getLongitude()`
- `public double getAltitude()`
- `public double getAzimuth()`
- `public PointD getLastPos()`

Metoda `getLastPos()` vrací strukturu typu `PointD`, což je obdoba standardní třídy `Point` z knihovny `System.Drawing`. Liší se pouze v použitém datovém typu jednotlivých složek, zde je to `double`.

Dále je přítomna public metoda connect, jenž zajistí otevření komunikačního kanálu na zadaném portu a spustí přijímání zpráv. Přijímání je tvořeno asynchronně spouštěnou metodou `GNSSDataReceived` a běží tak dlouho, dokud není ukončeno metodou `close()`. Každá přijatá zpráva je zpracována pomocí metody `parseGNSS()`, výsledkem zpracování jsou nové hodnoty aktuální pozice, které jsou ihned zapsány do patřičných proměnných.

Poslední metodou ve třídě je kontrola přítomnosti nových dat, `isUpdated()` vrátí hodnotu true pouze pokud od posledního volání byly hodnoty poslední polohy zaktualizovány.

6.2 Komunikace s navigačním počítačem

Navigační počítač je samostatně funkční součást. Ovládá pohonné systémy robotu a při dostupných polohových informacích dokáže dovést robot na vybranou pozici. Při zadání série bodů trasy je pak schopen tuto trasu následovat a dojet na její konec. Primárním účelem této funkce je možnost návratu z cílové pozice zpět do startovní, dle bodů zaznamenaných při jízdě od startu k cíli.

Této funkce je využito pro metodiku objíždění překážek. Účelem je změnit body trasy, aby nedošlo ke kolizi s překážkou. Navigační počítač předá sérii bodů trasy palubnímu počítači, jenž se postará o jejich zpracování. Poté palubní počítač pomocí příkazů upraví body trasy v navigačním počítači tak, aby došlo k zabránění střetu s překážkou.

6.2.1 Protokol komunikace

Pro komunikaci mezi počítači, palubním a navigačním, je navržen speciální čistě textový protokol. Formát zpráv částečně vychází z již použitého protokolu NMEA-

0183. Používá se zde stejný uvozující znak $, ukončující CR+LF i stejný kontrolní součet.

V následující sekci budou uvedeny jednotlivé typy zpráv se stručným popisem. Pro přehlednost budou uvedeny v předpokládaném chronologickém pořadí.

Formát zpráv

`$NALST,itemID,latitude,latitudeNS,longitude,longitudeWE,latitude_err,longitude_err*checksum(CR+LF)`

Po zapnutí navigačního počítače, či spuštění autonomního režimu dojde k odeslání celého seznamu bodů pomocí příkazů NALST do palubního počítače. Zpráva obsahuje index bodu trasy, jeho zeměpisnou šířku a délku a jejich toleranci v metrech.

`$NALST,END*checksum(CR+LF)`

Po celém seznamu pak následuje odeslání zprávy potvrzující konec listu.

`$NAWPT,itemID*checksum(CR+LF)`

Dále dojde k odeslání zprávy NAWPT obsahující index bodu, který byl právě dosažen. Pokud tato zpráva není přijata, předpokládá se, že nebyl dosažen ani první bod s indexem 0. K odeslání této zprávy navigačním počítačem dojde pokaždé, kdy je dosaženo dalšího z bodů trasy.

`$CAREM,itemID*checksum(CR+LF)`

Zpráva CAREM je odeslána palubním počítačem pro odstranění zvoleného bodu ze seznamu. Obsahuje pouze index bodu určeného ke smazání.

`$CAINS,itemID,latitude,latitudeNS,longitude,longitudeWE,latitude_err,longitude_err*checksum(CR+LF)`

Příkazem CAINS dojde k vložení nového bodu trasy na zadaný index. Původní bod na zadaném indexu se automaticky přesune na další pozici, taktéž i zbytek listu.

`$CAREM,END*checksum(CR+LF)`
`$CAINS,END*checksum(CR+LF)`

Po provedení všech požadovaných změn odešle palubní počítač zprávu o ukončení odebírání a přidávání bodů do listu.

`$CASPD,speed(000.0-100.0Õ%-100%)*checksum(CR+LF)`

CASPD je speciálním příkazem pro možnost regulace maximální rychlosti pohybu. Původně byl zamýšlen pro řízení maximální rychlosti dle náročnosti terénu či členitosti povrchu. Zatím tato funkce ale nebyla implementována.

6.2.2 Implementace protokolu

Pro implementaci vytvořeného protokolu byla vytvořena třída `NAV_PC_com`. Metody této třídy jsou pak externě volány tak, aby bylo dodrženo správné pořadí jednotlivých zpráv. Je nutné volat metodu třídy pro každé odstranění i přidání bodu do trasy. Po dokončení tohoto procesu je také třeba zavolat metodu pro odeslání ukončující zprávy.

Třída WP_LIST

Třída `WP_LIST` byla vytvořena pro snadnou správu jednotlivých bodů trasy. Je navržena, aby mohla být velmi snadno upravena na jiný počet hodnot v každém prvku (například zahrnout i výškovou souřadnici).

Třída interně pracuje se standardní třídou `List` objektů `double`. Všechny hodnoty obsažené v `WP_LIST` tak musí být také typu `double`.

Pro snazší implementaci obsahuje třída metody pro získání jednotlivých hodnot zeměpisné šířky i délky, ale také obou souřadnic najednou ve struktuře `PointD` (odstavec 6.1.2).

Za stejným účelem jsou přetíženy metody pro vkládání nových bodů. Je možné přidávat body se souřadnicemi uvedenými zvlášť, či dohromady ve struktuře `PointD`.

Třída NAV_PC

Tato třída je opět potomkem třídy `com`(odstavec 6.1.2). Komunikace je opět zajištěna protokolem UDP, proto tato třída obsahuje instanci třídy `UdpClient`. Jenž je inicializována pomocí metody `connect()`. Zde je navíc nutné, kvůli obousměrné komunikaci, uvést kromě portu i IP adresu navigačního počítače.

Třída dále obsahuje instanci třídy WP_LIST, určenou pro uschování bodů trasy přijatých od navigačního počítače.

Po otevření patřičného UDP komunikačního kanálu se spustí příjem dat z navigačního počítače. Přijatá data jsou dekódována pomocí metody parseNAV_PC a získané souřadnice nových bodů jsou ukládány do listu. Po přijetí zprávy o ukončení vysílání bodů je nastavena proměnná allWPreceived na hodnotu true. Při přijetí zprávy NAWPT je aktualizována hodnota proměnné lastReachedWP.

Další metody třídy NAV_PC:

`public int getLastReachedWP()`
`public int getNextWP()`

Tyto metody slouží k získání indexu posledního projetého či následujícího bodu trasy.

`public bool isWPreceived()`
Voláním této metody lze získat informaci o ukončení příjmu nových bodů. Je-li seznam bodů kompletní, je vrácena hodnota true.

`public void setSpeed(double speed)`
Pomocí této metody dojde k odeslání zprávy s nastavením maximální možné rychlosti. Údaj je relativní hodnota zadaná v procentech (0-100 %).

`public void addWP()` Tato metoda je velmi podobná jako u třídy WP_LIST a slouží k přidávání bodů do listu. Kromě toho zde dochází k odeslání nového bodu do navigačního počítače.

`public void removeWP(int ID)`
Slouží opět k odebrání z listu, zároveň dojde k odeslání zprávy k odstranění daného bodu i v navigačním počítači.

6.3 Komunikace s SR4000

Hardwarová vrstva komunikace mezi dálkoměrnou kamerou a palubním počítačem je realizována pomocí sítě Ethernet. Zařízení komunikuje pomocí protokolu TCP/IP. Pro usnadnění práce se zařízením je v manuálu ke kameře [20] uveden odkaz na stažení ukázkového programu v jazyce C# (`http://www.mesa-imaging.ch/temp/SwissRanger.zip`).

6.3.1 Instalace ovladače SR4000

Aby bylo možné se s kamerou spojit a komunikovat je zapotřebí nainstalovat výrobcem dodávané ovladače. Tyto ovladače jsou ke stažení na stránkách výrobce MESA IMAGING jak pro operační systémy Windows, tak pro Linux, oboje pro 32 i 64 bitové varianty.

Pro Windows se jedná o klasický instalační exe soubor. Instalace je podrobně popsána v manuálu [20]. Po nainstalování je kamera připravena k použití.

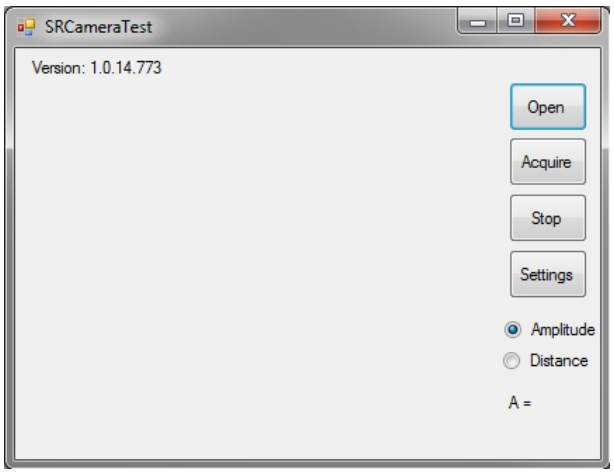
6.3.2 Ukázkový program pro SR4000

Pro vývoj software používajícího tuto kameru existuje několik vývojářských balíčků, které obsahují hotové funkce pro základní komunikaci. Součástí balíčků bývá i příklad použití ve formě jednoduchého demonstračního programu. SDK balíček pro C#

lze stáhnout pouze od jiných firem, například od MetriCam `http://www.metricam.net/metricam/`. K instalaci tohoto balíčku ale nakonec nedošlo, hlavně z důvodu minimalizace počtu knihoven nutných k běhu programu.

Pro test kamery a poté i tvorbu softwaru byl použit ukázkový příklad přímo od MESA IMAGING. Po rozbalení staženého balíčku bylo zjištěno, že se jedná o projekt vytvořený přímo pro Visual Studio. Tímto byla instalace usnadněna na maximální možnou úroveň. Stačilo pouze složku s rozbalenými soubory přesunout do požadovaného umístění a otevřít soubor SwissRangerInterface.sln ve Visual Studiu.

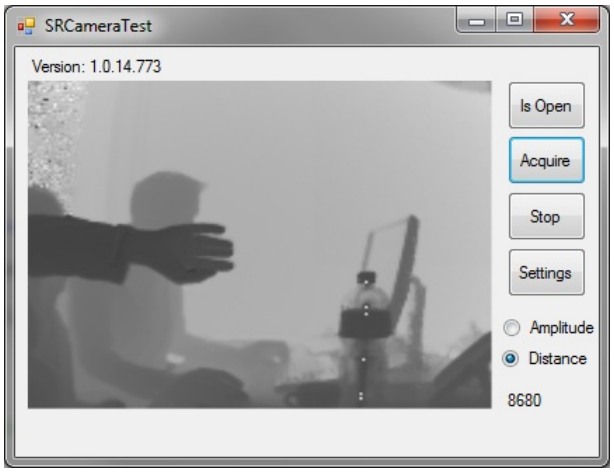
Po zkompilování a spuštění načteného programu se objeví okno s několika tlačítky (obr. 6.2). Po kliknutí na tlačítko Open se pouze změní jeho popisek na Is Open. Teprve po kliknutí na tlačítko Acquire se spustí ukázková video smyčka. Tlačítkem Stop lze přehrávání pozastavit. Tlačítko Settings sice i při běhu ukázkové smyčky zobrazí nabídku pro úpravu hodnot, ale jejich změna se nijak neprojeví.



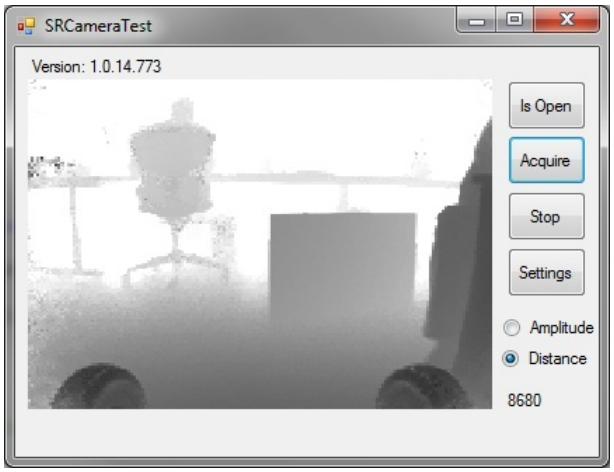
Obr. 6.2: Ukázkový program po spuštění

Dále ve vizualizaci nalezneme dvojici selekčních tlačítek *Amplitude* a *Distance*. Při zatažení tlačítka *Amplitude* se zobrazují amplitudová data, odpovídající intenzitě odraženého infračerveného záření. Při zatržení *Distance* se nám zobrazí data o vzdálenosti jednotlivých bodů (obr. 6.3).

Pro připojení ke skutečné kameře je potřeba v souboru Form1.cs zvolit patřičné rozhraní Ethernet či USB. V případě volby Ethernetu je také nutné zadat odpovídající IP adresu, která je zadaná v kameře. Lze také zvolit variantu průvodce připojením. Dále je třeba zvolit správnou maximální vzdálenost pro převod na šedotónový obraz vzdálenosti z rozsahu 0 až 65535. Po těchto úpravách (a nainstalování ovladače) je možné se připojit k fyzické kameře (obr. 6.4).



Obr. 6.3: Vzdálenostní data v ukázkovém programu



Obr. 6.4: Experimentální data získaná TOF kamerou

6.4 Zpracování prostorových dat

Část získávání dat je již vyřešena ve vzorovém příkladu. Jak již bylo napsáno výše, z kamery lze získat několik různých typů dat [20], jmenovitě jsou to tyto:

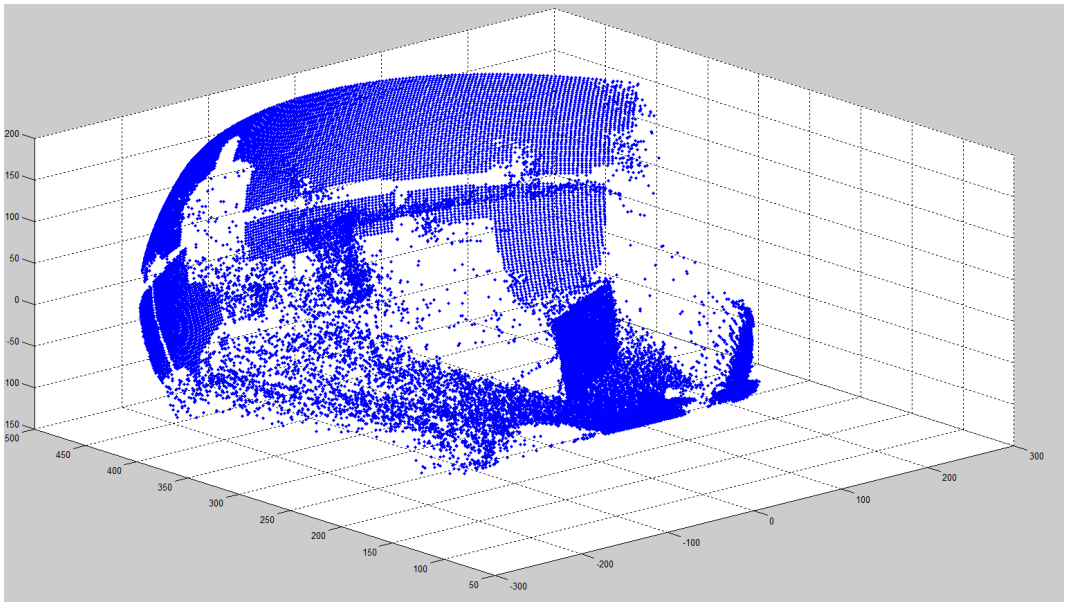
- vzdálenostní data (měření fáze)
- (x, y, z) souřadnicová data [m]
- amplitudová data (podobné šedotónovému obrazu)
- mapa důvěryhodnosti (odhad spolehlivosti)

Zpracování amplitudových a vzdálenostních dat zůstává stejné z ukázkového příkladu a nemá na výslednou navigaci žádný vliv. Přibyla pouze část pro zpracování prostorových dat, která jsou přímo vypočítávána pomocí nainstalovaného ovladače.

Tato data by již měla být zbavena všech známých zkreslení [20].

6.4.1 Výšková mapa

Pro realizaci detekce překážek je vhodné nejprve vytvořit výškovou mapu. Tato mapa se nejlépe vytvoří ze známých 3D souřadnic. Na obrázku 6.5 je zobrazen výstup v podobě zobrazených jednotlivých bodů v prostoru.



Obr. 6.5: Reprezentace bodů v prostoru

K vytvoření 2D výškové mapy je v podstatě potřeba pouze změnit úhel pohledu na snímaný prostor a výškovou souřadnici převést na hodnotu v daném bodě roviny. Přetvořením prostorové mapy na plošnou samozřejmě dojde ke ztrátě některých informací, ne však těch, které jsou důležité pro jízdu po povrchu.

Pro převod sady trojrozměrných dat do plošné mapy byla napsána funkce `XYZtoMap`. Pracuje principiálně tak, že postupně probírá jednotlivé body. Pro každý bod je vypočítána nová výška a pozice na nově vytvářené mapě. Pokud už v mapě na dané pozici je zapsaná výška z některého předchozího bodu, je nutné ji porovnat s novou hodnotou, kterou chceme zapsat. Na pozici se nakonec zapíše ta hodnota, jejíž absolutní hodnota je větší, tedy ta extrémnější.

K výpočtu polohy nového bodu je třeba zvolit nějaké pevné měřítko, podle něhož bude pozice přepočítávána. Toto měřítko se volí v globální proměnné `map_pixel` a udává velikost jednoho bodu v metrech. Tato hodnota tak přímo ovlivňuje minimální velikost detailů na mapě. Další důležitou hodnotou je požadované rozlišení

výsledné mapy vytvořené z každého snímku. Volba tohoto rozlišení nijak neovlivňuje velikost bodu, ale pouze velikost mapy. Při volbě příliš malého rozlišení tak bude ve výsledné mapě pouze část ze získaných dat. Naopak při příliš velkém bude mapa obsahovat velké množství prázdného prostoru bez informace.

Přepočet souřadnicového systému kamery

Prostorová data získaná pomocí kamery je nutné přepočítat vzhledem k jejímu umístění. Vliv zde má výška umístění nad povrchem a její sklon. To ovlivňuje jak velikost a tvar zorného pole, tak vztah mezi souřadnicovým systémem kamery a povrchu, který mapuje.

Prostorové souřadnice získané z kamery jsou vztažené ke středu její přední plochy. Pokud by kamera byla vodorovně s povrchem, po němž se robot pohybuje, stačilo by pouze snížit všechny Y souřadnice o výšku umístění kamery. Takováto poloha je značně nevýhodná z hlediska zorného pole promítnutého na plochu (viz. odstavec 3.2.1).

Proto bylo třeba přidat možnost nastavit libovolný úhel sklonu kamery a přidat odpovídající přepočet. Úhel je měřen v kladném směru od osy kamery k horizontální rovině (míří-li kamera směrem k zemi, úhel je kladný). Přepočet byl zahrnut do funkce `XYZtoMap` popsané v předchozím odstavci. Úhel sklonu kamery stejně jako výšku umístění lze zadat do připravených textových polí v okně aplikace. Při zadání správného úhlu sklonu a výšky kamery budou body ležící v rovině povrchu (po němž se robot pohybuje) přepočítány na výšku 0.

6.5 Tvorba mapy a kontrola trasy

Vytvořením výškové mapy z jednoho snímku máme k dispozici pouze malou část informací o okolí robotu. Proto je pro spolehlivé obcházení překážek nutné skládat mapu z postupně přibývajících dalších snímků. Každý snímek je umístěn do mapy dle aktuální pozice robotu ve skutečném prostoru. Aby však bylo možné nový snímek umístit, je nejprve nutné vypočítat jeho pozici v souřadnicovém systému výsledné mapy. Dále je také nutné, aby v případě, kdy se snímek již do mapy nevejde, došlo k jejímu patřičnému rozšíření.

Pro tvorbu mapy a realizaci s tím souvisejících funkcí byla vytvořena třída `mapping`. Tato třída obsahuje instanci třídy `Bitmap`, jenž obsahuje kompletní mapu. Součástí je také řada proměnných popisujících vlastnosti této mapy.

- `private int mapWidth`
- `private int mapHeight`
- `public double map_pixel`

- `private double origin_latitude`
- `private double origin_longitude`
- `private int origin_shiftX`
- `private int origin_shiftY`
- `private double kx`
- `private double ky`

Proměnné `mapWidth` a `mapHeight` odpovídají rozměrům celkové mapy. Velikost jednoho pixelu mapy v reálném prostředí je obsažena v `map_pixel`. Ostatní proměnné budou popsány v následujících odstavcích.

6.5.1 Přepočet souřadnic

Přepočet zeměpisných souřadnic (ϕ, λ) na lokální metrické pravoúhlé souřadnice (x, y) může být i značně komplikovaná záležitost. Záleží především na požadované přesnosti výsledné mapy.

V nejjednodušším případě můžeme zemi považovat za ideální kouli o poloměru odpovídajícím střednímu poloměru Země 6371 km. Dále je třeba si zvolit nějaký výchozí bod se známými zeměpisnými souřadnicemi. Tento bod pak budeme považovat za střed (se souřadnicemi 0,0) nového metrického plošného systému. Souřadnice tohoto bodu (ϕ_0, λ_0) se uloží do proměnných `origin_latitude` a `origin_longitude`.

Pro tento bod jsou vypočítány převodní konstanty. Pro převod mezi zeměpisnou šířkou a Y souřadnicí vyjde vždy stejná hodnota, dle vzorce 6.2 nezávisle na zeměpisné poloze a bude uložena v proměnné `ky`. Hodnota `kx` pro zeměpisné délky je závislá na zeměpisné šířce, dle vzorce 6.1. S polohou blížící se k zeměpisným pólům dochází ke snižování této konstanty.

$$kx = \frac{2 \cdot \pi \cdot R \cdot cos(\phi)}{360} \tag{6.1}$$

$$ky = \frac{2 \cdot \pi \cdot R}{360} \tag{6.2}$$

Při vytvoření nové mapy je aktuální pozice uložena jako počáteční a po převodu do metrické pravoúhlé soustavy se bude nacházet v jejím počátku (0, 0). Pozice počátku je ale pohyblivá v rámci souřadnic výsledné mapy, proto je uložena v proměnných `origin_shiftX` a `origin_shiftY`. Přepočet dalších souřadnic probíhá identicky pro zeměpisnou šířku i délku dle vzorců 6.3 a 6.4.

$$x = (\lambda - \lambda_0) \cdot kx \tag{6.3}$$

$$y = (\phi - \phi_0) \cdot ky \tag{6.4}$$

Složitější a přesnější metodou může být výpočet převodních konstant ze znalosti přesnějšího lokálního Zemského poloměru pomocí elipsoidu či geoidu spolu s připočtením nadmořské výšky. Rovníkový a pólový poloměr Země se liší dle použitého aproximačního elipsoidu přibližně o 21 km, což při středním poloměru Země 6371 km činí rozdíl pouhých 0,33 %. Stejně velká chyba se takto může projevit na nepřesnosti měřítka výsledné mapy. Měřítko mapy není z hlediska zaměření aplikace podstatné a jeho nepřesnost se projeví pouze v rámci zorného pole robotu, jenž je řádově v jednotkách metrů. Vzniklá chyba může být v řádu mm, maximálně několik cm, což je srovnatelná hodnota s přesností použitého RTK GNSS systému. Proto pro výpočet převodních konstant není podstatná vyšší přesnost a bylo použito jednoduššího způsobu.

O přepočet zeměpisných souřadnic na polohu v mapě se stará metoda `ToImageCoord()`, jejímž vstupním parametrem je struktura `PointD` obsahující polohu v zeměpisných souřadnicích. Přepočtená poloha je vrácena v obdobné, ale celočíselné struktuře.

6.5.2 Dynamické rozšiřování

Každá nová část mapy, jenž má být přidána do mapy je nejprve rotována pod odpovídajícím úhlem, aby odpovídala směru robotu vzhledem k zeměpisnému severu. K tomuto účelu je využito funkcí obsažených v knihovně `System.Drawing`, jenž umožňuje provádět základní transformace obrazu a je standardní součástí instalace Visual Studia.

Dále dojde k ověření, zda je možné část vložit přímo do stávající mapy. Není-li to možné, vyhodnotí se o kolik je potřeba mapu zvětšit a kterým směrem. Následně se vytvoří nová rozšířená mapa s požadovanými rozměry, do níž se vloží na odpovídající pozice původní mapa a nová část mapy.

Dynamické rozšiřování je součástí metody `addToMap()`, jenž zajišťuje přidání nové části mapy do celkové mapy. Jejími vstupními parametry jsou nová část mapy třídy `Bitmap`, zeměpisná šířka, délka a úhel natočení. Patřičné přepočty souřadnic se provádí automaticky na základě globálních proměnných třídy popsaných v předchozích odstavcích.

6.5.3 Kontrola průjezdnosti trasy

Kontrola zda je zadaná trajektorie průjezdná, je nutná především pro určení stavu, kdy je nutné trasu změnit, aby nedošlo ke kolizi. Tento úkol vykonává metoda `checkPath()`. Jejím vstupem je série bodů trasy v poli struktur `PointD`, práh výšky překážky a požadovaná šířka cesty. Výstupem je počet kolizních bodů překážek vyšších než prahová hodnota a bodů patřičně široké trasy vedené zadanými body.

Prahová hodnota je v rozsahu (0.0-1.0) a je vztažena k maximální výšce překážky. Tato metoda opět využívá knihovny `System.Drawing`, která mimo jiné umožňuje vykreslit sérii čar ze zadaných bodů, čehož je zde využito.

6.5.4 Přeplánování trasy

Přeplánování trasy je hlavním úkolem celého programu. Je implementováno pomocí metody `replanPath()`. Jako vstupní parametry zde jsou výchozí a cílový bod, oboje pomocí struktury `PointD`. Nezbytné jsou opět prahová hodnota překážky, šířka cesty a navíc rozestup bodů výsledné trajektorie.

Jak již bylo psáno v kapitole 4, je zde realizováno vyhledávání pomocí A* algoritmu. V případě, že není možné trasu najít, je vrácena hodnota `null`. Tento případ může nastat například, pokud výchozí či cílový bod leží na poloze překážky, nebo je překážkou obklíčen a nekolizní cesta tak neexistuje.

V případě, kdy překážka zasahuje až na kraj mapy je cesta nalezena i mimo aktuální mapu. Neznámé území je zde předpokládáno jako průjezdné.

V případě, že cestu lze najít, vrátí metoda pole struktur `PointD` se souřadnicemi bodů objízdné trasy. Body trasy mezi sebou mají rozestup podle zadaného vstupního parametru.

6.6 Implementace kompletního algoritmu

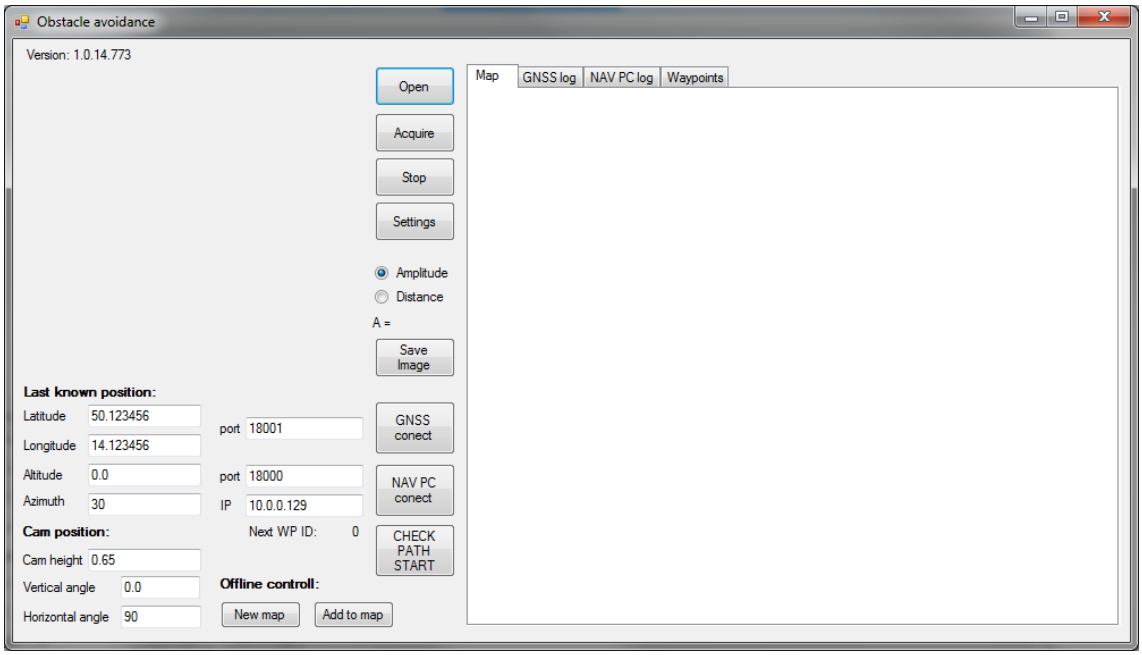
Kompletní implementace musí zajistit obsluhu všech výše popsaných tříd dle navrženého algoritmu, popsaného v kapitole 4. Cílem je tedy vytvoření uživatelské aplikace umožňující spustit režim automatického vyhýbání se překážkám.

6.6.1 Vizualizace

Aplikace je řešena spolu s grafickým uživatelským rozhraním, nebo-li GUI (obrázek 6.6).

Vychází opět z ukázkové aplikace k dálkoměrné kameře SR4000. Proti původnímu řešení je výsledné GUI rozšířeno o řadu prvků. V dolní levé části přibylo zobrazení aktuální pozice a nastavení pozice kamery. Dále tlačítka pro připojení k GNSS přijímači a navigačnímu počítači spolu s textovými poli pro nastavení parametrů připojení.

V pravé části okna pak přibilo zobrazení 4 karet, které slouží pro přepínání mezi zobrazením kompletní mapy, záznamu GNSS komunikace, záznamu komunikace s navigačním počítačem a seznamu bodů trasy.



Obr. 6.6: GUI navržené aplikace

6.6.2 Automatický režim

Aplikace byla vyvíjena pro přehlednou prezentaci funkce objíždění překážek.

Chronologický postup je následující. Nejprve je nutné získat informace o aktuální poloze. Za tímto účelem je inicializována instance třídy `GNSS_com` a zavolána její metoda `connect()` se vstupním parametrem obsahujícím číslo portu, na kterém jsou poziční data vysílána.

Dále je třeba spustit komunikaci s navigačním počítačem a získat seznam bodů trasy. K tomuto účelu je inicializována instance třídy `NAV_PC_com` a její metoda `connect()`. V parametrech je zde třeba kromě čísla portu uvést i IP adresu navigačního počítače.

Poté, dojde ke spuštění komunikace s dálkoměrnou kamerou. Jako v ukázkové aplikaci je zde použit časovač pro pravidelnou obnovu dálkoměrných dat.

Nyní již konečně přichází na řadu navržená smyčka algoritmu dle vývojového diagramu 4.5. Algoritmus se pak dále větví na smyčku, jenž detekuje překážku v naplánované cestě a smyčku zajišťující objetí překážky. Přepnutí z jednoho stavu do druhého je indikováno stavovou proměnnou `noObstacle`, jenž má hodnotu `true` pokud není na původní trase ležící před robotem žádná překážka.

6.7 Dosažené výsledky

Navržený software byl nejprve testován s umělými daty. Po odstranění nalezených chyb se pokračovalo k testům na robotu a poté i k testům ve venkovním prostředí.

Testy probíhaly s robotem, jenž byl propojen ethernetovým kabelem s přenosným počítačem, na kterém byl spuštěn realizovaný (testovaný) program. Kvůli vyšším datovým tokům především z dálkoměrné kamery nemohlo být použito bezdrátové připojení. V následujících odstavcích budou popsány výsledky dosažené při testech v reálném prostředí.

6.7.1 První test

První test ve venkovním prostředí proběhl za poměrně slunného dne, což potvrdilo nevhodnost dálkoměrné kamery SR4000 pro provoz za těchto podmínek. Dále bylo ověřeno, že přesnost RTK GNSS systému je pro určování polohy naprosto dostatečná, avšak ne příliš spolehlivá. Hlavní problém spočívá v nutnosti příjmu signálu z většího počtu GNSS satelitů. Již při slabém zaclonění jedné z přijímacích antén je obtížná inicializace polohy. I v případě správné inicializace neustále hrozí ztráta synchronizace. Například už při vzdálenosti přibližně 4 m od stěny vyšší budovy došlo ke ztrátě signálu.

Výsledkem testu bylo několik vytvořených map. Ty ale vlivem šumu vzniklého přesvícením modulovaného záření dálkoměrné kamery popisují spíše trasu pohybu než mapu překážek.

6.7.2 Druhý test

Pro druhý test byly vybrány příznivější světelné podmínky, i přes to byla detekce překážek značně problematická. Nastavení polohy TOF kamery totiž muselo být změněno z důvodu stínění jedné z antén GNSS přijímače. Výsledná poloha kamery již není tak výhodná pro detekci překážek jako původní pozice.

Bohužel se kvůli neodladěným chybám v programu nepodařilo otestovat funkčnost obcházení překážek. Byla tedy pouze otestována funkčnost obousměrné komunikace s navigačním počítačem.

6.7.3 Závěrečný test

Pro závěrečný test byly opět zvoleny vhodné světelné podmínky. Vyskytla se však neočekávaná komplikace s během programu v emulovaném prostředí na PC s Linuxovým operačním systémem. Program přestal pracovat v kritické části, kterou je vykreslení trajektorie dle zadaných bodů trasy. Jelikož detekce překážek interně

využívá právě tohoto principu, nebyl program schopen správné funkce a překážky detekovat. Pod operačním systémem Windows se program jeví jako plně funkční.

Tento test byl poslední možný před termínem odevzdání práce. Realizovaný program tak zůstává bez výsledků z testu v reálných podmínkách.

7 ZÁVĚR

V první řadě byla vyhotovena rešerše dostupné literatury, týkající se bezkolizní navigace mobilního robotu. Na jejím základě a konstrukčních možnostech robotu byl vybrán snímač pro detekci překážek. Tímto snímačem se stala dálkoměrná kamera SR4000, která již byla na robotu instalována. Její parametry jsou pro mobilní robotiku vhodné, až na problematickou funkčnost při intenzivnějším okolním osvětlením.

Tato nevýhoda byla při praktických testech významnější, než se původně předpokládalo. Za vhodných světelných podmínek však bylo možné testy provést. Pro použití v reálném provozu průzkumného robotu by bylo nutné tento snímač nahradit vhodným 3D laserovým scannerem.

Pro hledání náhradní cesty, v případě detekce překážky, byl na základě rešerše vybrán algoritmus A*. Výhody dynamických algoritmů byly shledány pro značně proměnné (neznámé) prostředí jako nepříliš efektivní.

Pro zachování modulárního systému bylo zvoleno, že vyhnutí se překážce bude provedeno úpravou bodů trasy. Navigační počítač tak bude schopen vykonávat svoji funkci nezávisle na připojení modulu pro bezkolizní navigaci. Dále byl navržen vlastní algoritmus pro přeplánování části kolizní trajektorie. Tento algoritmus zajišťuje přeplánování trasy pouze v případě, kdy je detekována překážka bránící průjezdu po zadané trase.

Pro ověření funkčnosti algoritmu byla vytvořena simulace v prostředí MATLAB. Tato simulace ověřila schopnost navrženého algoritmu zajistit změnu trajektorie, aby nedošlo ke kolizi. Simulace je výpočetně poměrně náročná a hledání optimální trasy mezi vzdálenými body je zdlouhavé. Díky použitému algoritmu pro hledání cesty je z dostupných dat vždy nalezena nejkratší možná varianta objízdné trasy. Algoritmus si dokáže poradit i s velmi složitými překážkami. V extrémním případě je schopen nalézt i cestu skrz bludiště.

Implementace navrženého algoritmu byla provedena napsáním aplikace v jazyce C#. Výsledná aplikace je provozována na standardním hardwaru odpovídajícím architektuře x86/x64 a komunikuje s navigačním počítačem pomocí síťového protokolu UDP.

Bylo provedeno několik testů v reálném prostředí, během kterých byla opravena většina nedostatků. Problém nastal při posledním testu, kdy bylo zjištěno, že vytvořená aplikace pracuje v emulované prostředí linuxového OS jinak, než přímo pod Windows. Tato chyba způsobila neschopnost aplikace detekovat překážky. Výsledky z reálného prostředí tak do termínu odevzdání práce zůstávají neznámé.

7.1 Možnosti vylepšení

Jak již bylo zmíněno výše, prvním návrhem na zlepšení je použití jiného snímače pro detekci překážek i za cenu nutných konstrukčních změn. Dálkoměrná kamera SR4000 se neosvědčila z hlediska práce ve venkovním prostředí.

Způsob klasifikace překážek podle jejich výšky nad rovinou robotu, by bylo vhodné nahradit sofistikovanější metodou. Především pro možnost použití robotu i ve výškově členitém terénu. Bylo by pak možné vyhodnocovat úhly stoupání či klesání svahů pro rozpoznání jejich sjízdnosti.

Pro reálné použití robotu k průzkumu by bylo vhodné upravit výslednou aplikaci do konzolové formy, s jednoduchým textovým vstupem a výstupem. Jeho použití v robotu pro reálné nasazení by pak bylo značně jednodušší.

LITERATURA

[1] 3D scanner. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-04-20]. Dostupné z: http://en.wikipedia.org/wiki/3D_scanner

[2] BAJER, L.: *Algoritmy pro pathfinding.* [cit. 2015-04-26]. Dostupné z: http://bajeluk.matfyz.cz/mff/algs-4-pathf.pdf

[3] BOSTELMAN, R., T. HONG a R. MADHAVAN. 2005. *Towards AGV Safety and Navigation - Obstacle Detection using a TOF Range Camera* In: 12th International Conference on Advanced Robotics: Seattle, WA, 17-20 July 2005. DOI: 0-7803-9177-2/05.

[4] BORENSTEIN, J., Y. KOREN a D. FOX. *The vector field histogram-fast obstacle avoidance for mobile robots. IEEE Transactions on Robotics and Automation.* vol. 7, issue 3, s. 278-288. DOI: 10.1109/70.88137. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=8813

[5] CLARKSON, S.: *Science Behind 3D Vision. Depth Biomechanics* [online]. [cit. 2015-04-25]. Dostupné z: http://www.depthbiomechanics.co.uk/?p=102

[6] FREEDMAN, B. *Depth mapping using prijected patterns* [patent]. US2010, 0118123A1. Uděleno 13.5.2010. Dostupné z: http://www.freepatentsonline.com/20100118123.pdf

[7] GP2Y0A21YK. SHARP. [online]. [cit. 2015-04-20]. Dostupné z: https://www.sparkfun.com/datasheets/Components/GP2Y0A21YK.pdf

[8] GREWAL M., WEILL, L., ANDREWS, A. 2000. *Global Positioning Systems, Inertial Navigation, and Integration*[online]. 1st ed. New York, NY [u.a.]: Wiley [cit. 2015-04-17]. ISBN 04-712-0071-9.

[9] Hardware info. OpenKinect. [online]. [cit. 2015-04-21]. Dostupné z: http://openkinect.org/wiki/Hardware_info

[10] HRLV-MaxSonar-EZ Series. MAXBOTIX. [online]. 2005 [cit. 2015-04-20]. Dostupné z: http://www.maxbotix.com/documents/HRLV-MaxSonar-EZ_Datasheet.pdf

[11] KOENIG, S. and LIKHACHEV, M.: *Incremental A*.* Atlanta. GA 30312-0280. Dostupné z: http://idm-lab.org/bib/abstracts/papers/nips2001.pdf. Georgia Institute of Technology.

[12] Laser scanners LMS5xx. SICK. [online]. [cit. 2015-04-20]. Dostupné z: https://www.mysick.com/PDF/Create.aspx?ProductID=45446&Culture=en-US

[13] Learning Obstacle Avoidance by Example. Pi Robot. [online]. [cit. 2015-04-27]. Dostupné z: http://www.pirobot.org/blog/0007/

[14] Lidar-Lite. PulsedLight. [online]. [cit. 2015-04-20]. Dostupné z: http://kb.pulsedlight3d.com/helpdesk/attachments/5007466446

[15] LIKHACHEV, M., et al.:: *Anytime Dynamic A*: An Anytime, Replanning Algorithm.* Pittsburgh. GA 30312-0280. Dostupné z: http://www.cs.cmu.edu/~ggordon/likhachev-etal.anytime-dstar.pdf. Carnegie Mellon University.

[16] LRR3: 3rd generation Long-Range Radar Sensor. BOSCH. [online]. [cit. 2015-04-20]. Dostupné z: http://spectrum.ieee.org/transportation/advanced-cars/longdistance-car-radar

[17] LUO, Y. and SCHMITZ, T. Infrared proximity sensing: Building blocks, mechanical considerations, design trade-offs. INTERSIL CORPORATION. EE-Times [online]. 2009 [cit. 2015-04-20]. Dostupné z: http://www.eetimes.com/document.asp?doc_id=1272536

[18] RIETDORF, A., DAUB, C. and LOEF P.:*Precise Positioning in Real-Time using Navigation Satellites and Telecommunication.* 3rd WORKSHOP ON POSITIONING, NAVIGATION AND COMMUNICATION 2006. Dostupné z: http://wpnc.net/fileadmin/WPNC06/Proceedings/34_Precise_Positioning_in_Real-Time_using_Navigation_Satellites_and.pdf

[19] SR4000 Data Sheet. MESA IMAGING. [online]. [cit. 2015-04-21]. Dostupné z: http://downloads.mesa-imaging.ch/dlm.php?fname=pdf/SR4000_Data_Sheet.pdf

[20] SR4000/SR4500 User Manual. MESA IMAGING. [online]. [cit. 2015-04-29]. Dostupné z: http://downloads.mesa-imaging.ch/dlm.php?fname=customer/Customer_CD/SR4000_SR4500_Manual.pdf

[21] STACHNISS, C. 2009. *Robotic mapping and exploration*[online]. Online-Ausg. Berlin: Springer [cit. 2015-04-17]. ISBN 978-364-2010-972.

[22] SUN, X., Yeoh, W. and Koenig S.:*Moving Target D* Lite.* CA 90089-0781. AAMAS 2010. Dostupné z: http://idm-lab.org/bib/abstracts/papers/aamas10a.pdf University of Southern California.

[23] Time-of-Flight Camera - An Introduction. Texas Instruments. [online]. [cit. 2015-04-27]. Dostupné z: http://www.ti.com/lit/ml/sbau219d/sbau219d.pdf

[24] Trimble BD982 GNSS Receiver Module. Trimble Navigation Limited. Version 4.30, Revision B, April 2011. Dostupné z: http://www.geoteam.dk/files/manager/pacificcrest/bd982_manual.pdf

[25] Ultrasonic Sensing. ROCKWELL AUTOMATION. [online] [cit. 2015-04-18]. Dostupné z: http://www.ab.com/en/epub/catalogs/12772/6543185/12041221/12041229/print.html

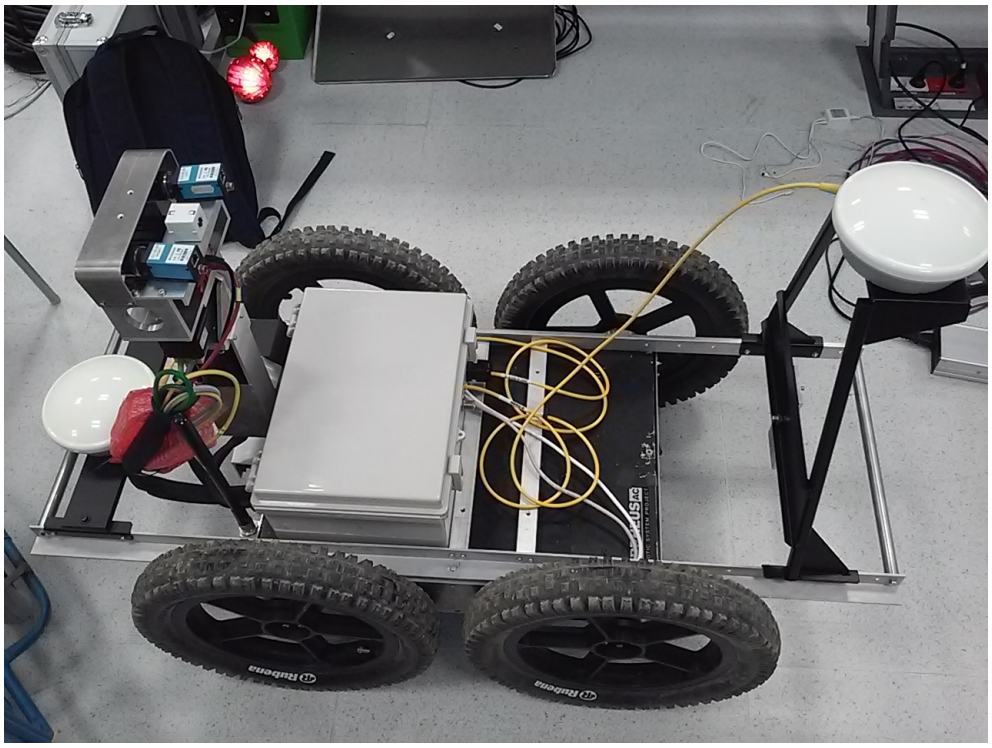
[26] ZUG, S. et al. Are laser scanners replaceable by Kinect sensors in robotic applications? In: *2012 IEEE International Symposium on Robotic and Sensors Environments Proceedings.* IEEE, s. 144-149. DOI: 10.1109/ROSE.2012.6402619. ISBN 978-1-4673-2706-0. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6402619

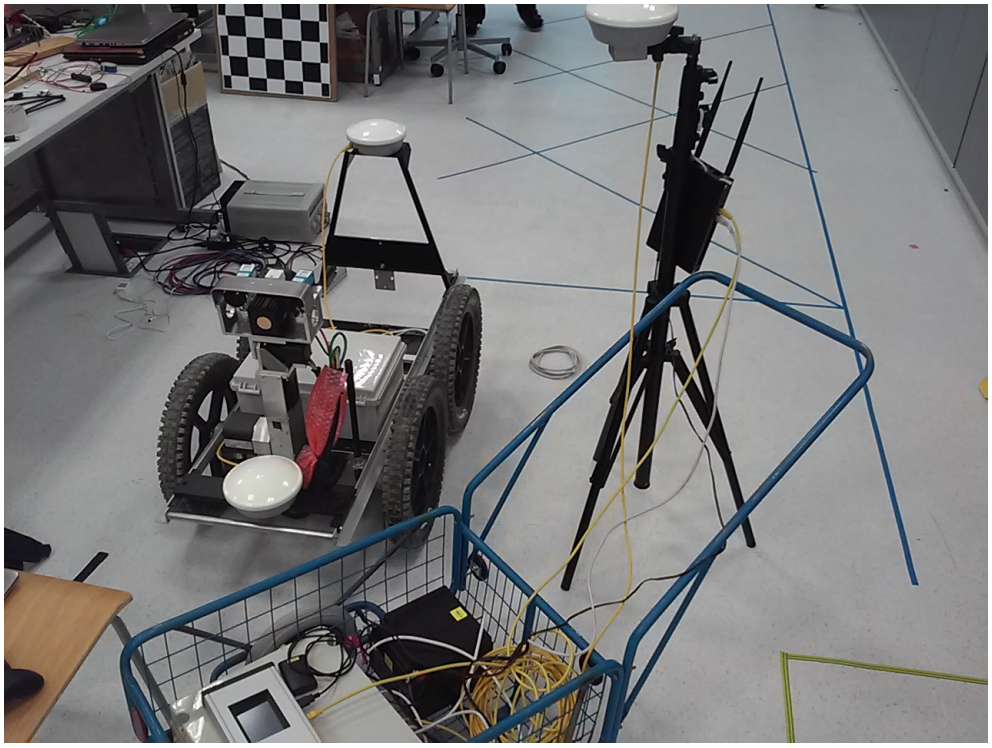
SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

FOV	úhel pohledu či záběru – Field of view
GNSS	globální satelitní navigační systém – Global Navigation Satelite System
LIDAR	způsob měření vzdálenosti využívající dobu letu světelného paprsku – Light Detection And Ranging
NPC	nehratelná počítačová postava – Non player character
RTK	systém diferenciální GNSS navigace - Real time kinematics
SDK	balíček pro vývoj specializovaného software – Software development kit
TOF	doba letu – Time of flight
UNS	umělá neuronová síť
UTC	čas na nultém poledníku – Coordinated universal time

SEZNAM PŘÍLOH

A ROBOT ORPHEUS - VÝVOJOVÁ VERZE





B OBSAH PŘILOŽENÉHO CD

- Elektronická verze diplomová práce (xstrit06.pdf)
- Zdrojové soubory textu práce (složka Diplomová práce)
- Soubory MATLAB simulace (složka Matlab simulace)
- Zdrojové soubory vytvořeného programu (složka ObstacleAvoidance)
- Fotografie robotu (složka Robot foto)
- Data z testů (složka Test data)