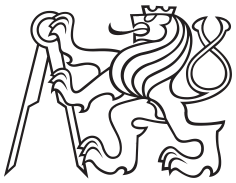
Bakalárska práca



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Lokálna vonkajšia navigácia mobilného robota pomocou mapovania priechodnosti terénu

Michal Kasarda

Školiteľ: Ing. Jan Chudoba
Január 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kasarda** Jméno: **Michal** Osobní číslo: **492050**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra kybernetiky**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Lokální venkovní navigace mobilního robotu s mapováním průchodnosti terénu

Název bakalářské práce anglicky:

Local Outdoor Navigation of a Mobile Robot by Terrain Traversability Mapping

Pokyny pro vypracování:

Cílem práce je návrh a implementace metody pro navigaci mobilního robotu v nerovném terénu. Vstupem metody je cílová pozice, které chceme dosáhnout. Metoda má zmapovat okolní prostředí robotu, vyhodnotit průjezdnost prostředí, zvolit vhodnou cestu s ohledem na průjezdnost a generovat řídicí příkazy které robota provedou po naplánované cestě.
• Seznamte se s metodami pro mapování prostředí a senzory používanými pro tento účel. Vypracujte rešerši.
• Navrhněte metodu navigace do zadaného cílového bodu. Vhodné senzorické vybavení konzultujte s vedoucím práce.
• Implementujte navrženou metodu a otestujte její funkčnost v simulátoru a/nebo (podle reálných možností) na laboratorním robotu.
• Proveďte vyhodnocení testů, diskutujte možnosti a případné omezení vytvořené metody.

Seznam doporučené literatury:

[1] Shan, T., Wang, J., Englot, B. & Doherty, K.. (2018). Bayesian Generalized Kernel Inference for Terrain Traversability Mapping. Proceedings of The 2nd Conference on Robot Learning, in Proceedings of Machine Learning Research 87:829-838
[2] Gerald Cook. Mobile Robots: Navigation, Control and Remote Sensing. Wiley-IEEE Press, 2011
[3] P. E. Hart, N. J. Nilsson and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," in IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107, July 1968
[4] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," in IEEE Robotics & Automation Magazine, vol. 13, no. 2, pp. 99-110, June 2006

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jan Chudoba inteligentní a mobilní robotika CIIRC

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **14.09.2022** Termín odevzdání bakalářské práce: **10.01.2023**

Platnost zadání bakalářské práce: **19.02.2024**

Ing. Jan Chudoba	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

Poďakovanie

Chcel by som sa poďakovať Ing. Janovi Chudobovi za trzpezlivosť, rady a odborné vedenie tejto bakalárskej práce. Taktiež dakujem svojej rodine za všetkú poskytnútu pomoc.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval samostatne a že som uviedol všetky použité informačné zdroje, v súlade s pravidlami Metodického pokynu o dodržovaní etických princípov pri príprave záverečných prac.

V Prahe, 5. januára 2023

.....

Michal Kasarda

Abstrakt

Táto práca sa zaoberá autonómnou navigáciou robota v neznámom prostredí. V rámci práce je vypracovaný prehľad techník a senzorov používaných pri simultánnej lokalizácii a mapovaní (SLAM). Následne je predstavený algoritmus iterative closest point (ICP), na ktorom je založená metóda ICP SLAM. Pri mapovaní je potrebné vyhodnotiť priechodnosť prostredia s ohľadom na kinematické a dynamické vlastnosti robota. V práci sú predstavené metódy analýzy priechodnosti prostredia. Zároveň je navrhnutý spôsob analýzy priechodnosti, založený na geometrických vlastnostiach prostredia. Na navigáciu robota je vybraná ICP SLAM metóda a implementovaná analýza priechodnosti spolu s hľadaním cesty v mape a regulátorom na sledovanie nájdenej cesty. Implementácia je otestovaná v rámci simulátoru Gazebo a testov v reálnom mestskom prostredí.

Kľúčové slová: simultánna lokalizácia a mapovanie (SLAM), iterative closet point (ICP), analýza priechodnosti prostredia

Školiteľ: Ing. Jan Chudoba
CIIRC B-323,
Jugoslávských partyzánů 3,
16000 Praha 6

Abstract

This thesis deals with autonomous navigation of mobile robot in unknown enviroment. This works contains overview of methods and techniques used for simultaneous localization and mapping (SLAM). The iterative closest point (ICP) alghorithm is introduced, on which the ICP SLAM method is based. It is necesery to evaluate traversability of enviroment while mapping. This evaluation is based on kinematic and dynamic properties of robot used. This work contains overview of traversability analysis methods. A method for traversability evaluation is proposed, based on geometric features of enviroment. For navigation, ICP SLAM method is chosen and a method for traversability analysis, pathfinding and regulator for path following is implemented. The implementation is then tested in Gazebo simulator and in real city enviroment.

Keywords: simultaneous localisation and mapping (SLAM), iterative closet point (ICP), traversability analysis of enviroment

Title translation: Local Outdoor Navigation of a Mobile Robot by Terrain Traversability Mapping

Obsah

Obrázky

Tabuľky

Kapitola 1
Úvod

Autonómni roboti sú dnes používaní v širokej škále aplikácií, cez roboty používane v domácnostiach [1], v armáde [2] alebo ako autonómne vozidlá [3]. Títo roboti musia byť schopní navigácie v neznámom prostredí, k čomu patrí schopnosť lokalizácie a mapovania. Problémom simultánnej lokalizácie a mapovania (SLAM) sa vedecká komunita zaoberá posledných 30 rokov. Za tento čas boli navrhnuté a úspešne otestované rôzne metódy SLAM. Roboti sa pohybujú rôznymi spôsobmi, sú stavané v rôznych veľkostiach a špecifikované na rôzne úlohy. V zmapovanom prostredí je potrebné pre každého robota vymedziť priestor, v ktorom je pohyb pre daného robota bezpečný. Touto problematikou sa zaoberá analýza priechodnosti prostredia. V tejto práci sa bližšie pozrieme na vyhodnotenie priechodnosti terénu pre robota HUSKY A200, pomocou LIDAR senzora.

1.1 Ciele práce

Hlavným cieľom tejto práce je vytvoriť systém, ktorý bude schopný autonómnej navigácie robota HUSKY A200 vo vonkajšom teréne. Prácu je možné rozdeliť do nasledujúcich podúloh:

1. Naštudovanie a výber vhodných senzorov, metód SLAM a metód analýzy priechodnosti prostredia.
2. Návrh a implementácia softvéru, ktorý zabezpečí analýzu priechodnosti terénu, hľadanie cesty a sledovanie danej cesty.

Kapitola 2

Simultánna lokalizácia a mapovanie

2.1 Formulácia a štruktúra úlohy SLAM

Simultánnu lokalizáciu a mapovanie (SLAM) je možno formálne popísať z pravdepodobnostnej perspektívy [4]. Znakom t je označený čas a znak x_t popisuje pozíciu robota. Trajektória robota je daná ako

$$X_T = \{x_0, x_1, x_2, ...x_T\}, \tag{2.1}$$

kde poloha robota v čase t je označená ako x_t. Znakom T je označený čas, v ktorom metóda končí. Počiatočná poloha x_0 je známa, ostatné pozície sa nedajú presne pozorovať.

Odometria poskytuje relatívne informácie o pohybe medzi po sebe nasledujúcimi polohami. Informácia o pohybe vykonanom medzi časom $t-1$ a časom t je označená ako u_t. Sekvencia

$$U_T = \{u_0, u_1, u_2, ...u_T\} \tag{2.2}$$

reprezentuje relatívne pohyby vykonané robotom. Pre merania bez nepresností by U_t bolo dostačujúce na získanie X_t pomocou počiatočnej polohy x_0. Avšak, odometrické merania sú nepresné a časom akumulujú chybu od skutočnej polohy.

Robot je schopný pomocou senzorov získavať informácie o prostredí, v ktorom sa nachádza. Mapa m reprezentuje prostredie skutočne bez šumu a chýb. Senzorické merania poskytujú informáciu medzi rysmi mapy m a polohou robota x_t. Ak predpokladáme, že robot vykoná jedno meranie v každom časovom okamihu, sekvencia meraní je daná ako

$$Z_T = \{z_0, z_1, z_2, ...z_T\}. \tag{2.3}$$

Problém SLAM rieši získanie mapy m a získanie sekvencie polôh robota X_T z odometrie a senzorických meraní. Problém SLAM vieme rozdeliť na dve hlavné formy, full SLAM a online SLAM.

Full SLAM odhaduje podmienenú pravdepodobnosť všetkých polôh v trajektórií X_T spolu s mapou m:

$$p(X_T, m|Z_T, U_T). \tag{2.4}$$

Online SLAM odhaduje podmienenú pravdepodobnosť polohy x_t spolu s mapou m:

$$p(x_t, m|Z_T, U_T). \tag{2.5}$$

Algoritmy online SLAM metód sú zvyčajne inkrementálne a sú schopné spracovávať merania osobitne. Naopak full SLAM metódy spracovávajú všetky merania naraz, keďže odhad prebieha naprieč všetkými polohami v X_T. Online SLAM metódy sú oproti full SLAM metódam výpočtovo menej náročné, avšak full SLAM metódy majú väčšiu šancu zotaviť sa z chybných meraní [4].

2.2 Lokalizácia

Lokalizáciou je myslený proces odhadu aktuálnych súradníc robota na základe informácií z dostupných senzorov. Úloha lokalizácie sa dá rozdeliť na problémy sledovania polohy a globálnej lokalizácie. Sledovanie polohy znamená určovať polohu robota vzhľadom na nejakú počiatočnú pozíciu. V tomto prístupe sú dôležité relatívne súradnice vzhľadom k počiatočnej polohe, preto nie je nutné poznať model prostredia. V prípade globálnej lokalizácie sa určuje poloha vzhľadom k predom známemu modelu prostredia. Obecne sa súradnice robota skladajú z polohy a orientácie. Poloha je popísaná v troch karteziánskych súradniciach (x,y,z) a orientácia v troch smerových uhloch označovaných ako „roll/pitch/yaw", t.j. rotácie voči osiam súradnicového systému x, y, z. Lokalizáciu je možné zjednodušiť, pokiaľ sa robot pohybuje po rovine. V takom prípade je poloha popísaná v dvoch karteziánskych súradniciach (x,y) a orientácia pomocou uhla θ, ktorý reprezentuje rotáciu voči ose kolmej na roviny x, y [5].

2.3 Mapovanie

Mapovanie je proces zostavenia modelu okolitého prostredia. Mapovanie, z pohľadu robotiky, prevádza fyzické rysy sveta na vhodnú dátovu reprezentáciu. Na základe mapy je robot schopný vytvárať rozhodnutia z ohľadom na terajší stav okolitého prostredia. Pri tvorbe mapy [4] popisuje tri základne problémy.

1. Mapy musia byť dostatočne kompaktné nato, aby s nimi bolo možné pracovať v ostatných častiach systému ako napríklad hľadanie trasy.
2. Mapy musia byť prispôsobené riešenému problému a charakteru mapovaného prostredia.
3. Mapa musí brať v úvahu nepresnosti, spojené so senzormi a odhadom stavu v ktorom sa robot nachádza.

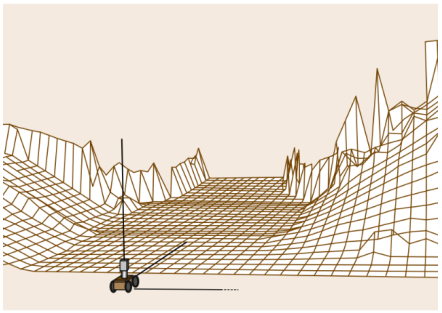
Obecne neexistuje univerzálna mapa, aplikovateľná k riešeniu každého problému. V nadchádzajúcom prehľade sú popísané niektoré z druhov máp, vhodných k riešeniu problému SLAM vo vonkajšom teréne [4].

2.3.1 Mriežka obsadenosti

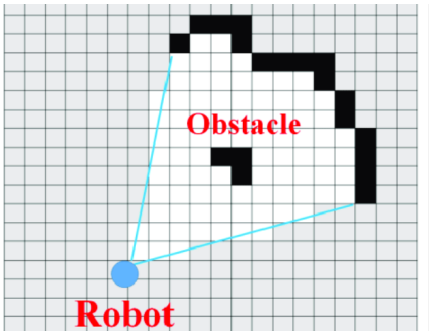
Mriežka obsadenosti obr. 2.2 rozdeľuje priestor na bunky, kde každá bunka obsahuje pravdepodobnostný odhad jej stavu. Bunky sú definované predvolenou dĺžkou hrany a majú presne určenú polohu. Bunka môže byť prázdna, obsadená, poprípade nie definovaná, v prípade, že nebola doposiaľ zmapovaná. [6],[7]. Výpočet hodnoty obsadenosti sa skladá z dvoch krokov. Najprv je vytvorený pravdepodobnostný model senzoru, ktorý umožňuje určiť hodnotu obsadenosti pre bunky, obsiahnuté v jednotlivých senzorických meraniach. V druhom kroku sa hodnoty obsadenosti zlučujú do konečnej mriežky obsadenosti, na základe Bayesovej vety alebo iných pravdepodobnostných prístupov. Nevýhodou mriežky obsadenosti je kvadratický, poprípade kubický rast pamäťovej náročnosti. Výhodou je zahrnutie nepresnosti senzorov do mapy a definovanie prázdneho priestoru, čo je vhodné pre algoritmy na plánovanie ciest [5].

2.3.2 2.5D mapa

Táto forma mapy sa skladá z 2D mriežky buniek, ktoré obsahujú pridanú informáciu. Jedným s najbežnejších použití je priradenie nadmorskej výšky do každej bunky, čím dostávame takzvanú výškovú mapu. Výšková mapa obr. 2.1 je oproti bodovej mape menej náročnejšou dátovou reprezentáciou a je obzvlášť vhodná pre aplikácie, kde sa robot pohybuje vo vonkajšom teréne, ktorý neobsahuje rôzne previsy, tunely a mosty. Ako pridanú informáciu je taktiež možné použiť informáciu o priechodnosti danej bunky, v takomto prípade hovoríme o mape priechodnosti [4].



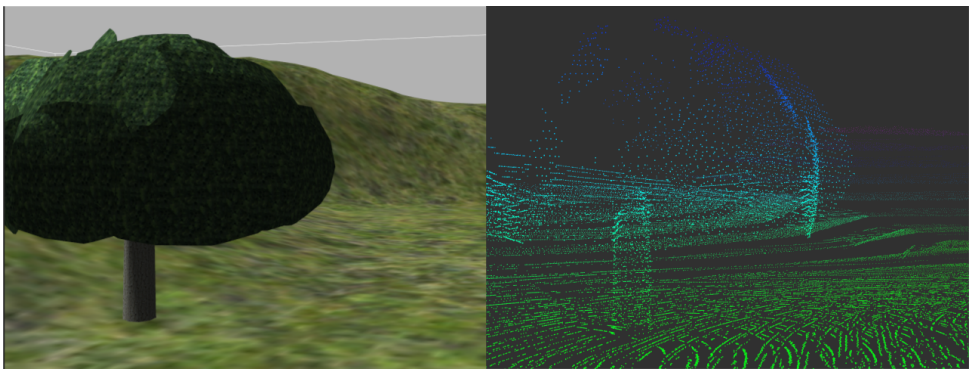
Obrázok 2.1: Výšková mapa [4]



Obrázok 2.2: Mriežka obsadenosti v 2D [8]

2.3.3 Bodová mapa

Bodovú mapu obr. 2.3 tvorí množina bodov, ktorá reprezentuje diskretizovanú geometriu okolitého prostredia. Bodová mapa je vhodná na lokalizáciu a mapovanie pomocou scan-matching metód [5]. Bodová mapa je schopná uchovať senzorické merania v ich originálnom rozložení. Jej nevýhodou je pomerne rýchlo narastajúci počet celkových bodov [4]. Výpočtovú náročnosť operácií, aplikovaných na bodovú mapu, je možné zmenšiť použitím vhodnej reprezentácie bodovej mapy. Jednou z takýchto štruktúr je K-D strom, popísaný v kapitole 4.4.



Obrázok 2.3: Bodová mapa simulačného prostredia

2.4 SLAM metódy

Väčšina SLAM metód spadá do nasledujúcich kategórií:

- SLAM s použitím rozšíreného kalmanového filtra (EKF SLAM)
- SLAM s použitím časticového filtra
- grafový SLAM

V tejto kapitole je popísaný základný princíp daných metód spolu s metódou scan-matching SLAM, ktorá je použitá v praktickej časti tejto práce.

2.4.1 EKF SLAM

EKF SLAM používa stavový vektor na odhad pozícii robota a pozícií rysov prostredia, spolu s kovariančnou maticou, ktorá reprezentuje chyby a neistotu týchto odhadov. Kovariančná matica taktiež obsahuje korelácie medzi odhadmi stavov robota a rysov prostredia. Robot pri pohybe prostredím získava senzorické merania, podľa ktorých sa aktualizuje stavový vektor a kovariančná matica, použitím rozšíreného kalmanového filtra. Pozorovaním nových rysov prostredia sú do stavového vektoru pridávané nové stavy. Tieto metódy pracujú s predpokladom toho, že polohu robota a polohy rysov prostredia môžeme reprezentovať normálnym rozdelením [4], [9].

2.4.2 SLAM s použitím časticového filtra

SLAM s použitím časticového filtra je metóda, ktorá používa častice na reprezentáciu polohy robota a mapy prostredia. Každá častica obsahuje vlastnú polohu robota, mapu prostredia a takzvaný importance faktor. Importance faktor vyjadruje pravdepodobnosť toho, že sa robot v danej polohe nachádza. Metóda je inicializovaná časticami s rovnakým importance faktorom a polohou, najčastejšie v počiatku súradnicového systému mapy prostredia. S pohybom robota sú získavané odometrické a senzorické merania. Nasleduje rekurzívne opakovanie takzvanej predikcie a korekcie. Pri predikcii je poloha častíc posunutá na základe odometrického merania. V korekcií je pre každú časticu vypočítaný importance factor, na základe zhody mapy prostredia a senzorického merania. Na záver sú častice s malým importance factorom zahodené a nahradené novými časticami. Na tomto princípe funguje prelomový FastSLAM algoritmus [4].

2.4.3 Grafový SLAM

Grafový SLAM reprezentuje polohy robota, rysy prostredia a vzťahy medzi nimi myšlienkou grafu. Graf je zložený z uzlov a hrán. Uzly môžu popisovať buď polohu robota v čase, alebo pozorované rysy. Hrany reprezentujú obmedzenia medzi uzlami. Medzi dvoma uzlami polohy je obmedzením odhad vzdialenosti medzi danými polohami, najčastejšie formou odometrie. Obmedzením medzi uzlom polohy a uzlom pozorovaného rysu je vzdialenosť pozorovaného rysu od danej polohy [4].
Takýto graf si je možné predstaviť ako systém hmotných bodov spojených pružinami [10]. Výpočet SLAM riešenia je analógiou výpočtu stavu tohoto systému s minimálnou energiou. Na tento výpočet sú aplikovateľné optimalizačné metódy, čo nám umožňuje riešiť SLAM problém ako optimalizačnú úlohu. Vzhľadom nato, že optimalizácia prebieha naprieč celým grafom, grafový SLAM patrí full SLAM metódam [4].

2.4.4 Scan-matching SLAM

Scan-matching metódy hľadajú transformáciu, ktorá minimalizuje vzdialenosť medzi meraním a referenciou. Základným predpokladom tejto metódy je to, že dané merania pochádzajú z rovnakého prostredia. Merania môžu byť reprezentované ako body, alebo ako špecifické rysy prostredia, napríklad detekované objekty. Scan-matching metódy je možné deliť na dva druhy, a to scan-to-scan metódy a scan-to-map metódy. Výpočet transformácie medzi po sebe idúcimi meraniami riešia scan-to-scan metódy. V prípade hľadania transformácie senzorického merania voči mape, hovoríme o scan-to-map metóde. Polohu robota je možné určiť na základe scan-to-map postupu, ktorého výsledkom je transformácia medzi senzorickým meraním a mapou. Senzorické meranie je možné podľa nájdenej transformácie zakomponovať do mapy[11], [12]. Na tomto princípe funguje metóda ICP SLAM 4.

Kapitola 3

Senzory

V tejto kapitole sú stručne popísane princípy fungovania bežných senzorov, používaných v SLAM metódach. Konkrétne senzory, použité v praktickej časti, sú podrobnejšie popísané v kapitole 6.2.

3.1 Odometria

Odometria je základný senzorický systém pre určovanie polohy robota. Na určenie polohy sú používané informácie z akčných členov robota alebo zo senzorov, ktoré robot obsahuje. V prípade robota, ktorý má za akčný člen koleso, je možné na odometriu použiť senzorické merania z inkrementálnych rotačných senzorov [5]. Pre všeobecný odhad odometrie je taktiež možno použiť inerciálnu meraciu jednotku (IMU).

3.1.1 Inerciálna meracia jednotka - IMU

Inerciálna meracia jednotka je zložená z trojice gyroskopov a trojice akcelerometrov. Tieto senzory sú montované do troch navzájom kolmých osí. V takomto rozložení sú schopné merať natočenie a zrýchlenie v 3D priestore. Akcelerometry slúžia na meranie zrýchlenia a gyroskopy na meranie uhlovej rýchlosti rotácie. Na základe nameraného rýchlostného vektora a natočenia voči osiam, je možné získať relatívnu polohu, orientáciu a zrýchlenie senzoru. Inerciálne senzory majú rýchlu odozvu. Sú však zaťažené chybou odhahu rýchlosti a polohy, ktorá bez korekcie časom kumulatívne narastá [5].

3.2 Globálny družicový polohový systém - GNSS

Globálny družicový polohový systém (GNSS) je jedným s najpoužívanejších systémov na lokalizáciu polohy. Systém je založený na prijímaní informácií od družíc obiehajúcich okolo zeme. Družice kontinuálne vysielajú signály, ktoré obsahujú informáciu o ich polohe a trajektórii v danom čase. Z prijatých informácii od viacerých družíc je následne možné dopočítať aktuálnu polohu prijímaču. Daná poloha je vyjadrená v horizontálnej rovine zeme [4]. Základná presnosť určenia polohy obsahuje chybu v jednotkách metrov. Chyba závisí

hlavne na podmienkach príjmu signálov z družíc a taktiež na počte družíc, od ktorých bol signál prijatý. Presnosť GNSS lokalizácie je možné vylepšiť príjmom a spracovaním diferenčných korekčných signálov, ktoré vysielajú pozemné alebo družicové vysielače. Nevýhodou GNSS je absencia signálov z družíc v budovách a pod zemou. V takýchto prípadoch nie je možné GNSS použiť [5].

3.3 Sonar

Sonar je jednoduchý senzor, ktorý používa akustické pulzy na detekovanie vzdialenosti k objektu. Keďže rýchlosť šírenia zvuku je zvyčajne známa, vzdialenosť detekovaného objektu je priamo úmerná času, v ktorom zvuková vlna letí od senzoru k prekážke a naspäť. Medzi hlavnú nevýhodu patrí presnosť meraní, obzvlášť vo vonkajšom prostredí. Presnosť meraní závisí na správnom určení rýchlosti vzduchu, ktorú ovplyvňuje teplota prostredia a vzdušné prúdenie. Kvalitu taktiež ovplyvňuje materiál prekážok. Materiál môže zvukovú vlnu pohltiť, poprípade odraziť. Pri úplnom pohltení zvukovej vlny prekážka nie je detekovaná. Pri odrazení zvukovej vlny môže dôjsť k určeniu nesprávnej vzdialenosti prekážky od senzoru. Výhodou sonaru je jeho relatívne nízka cena a veľkosť prostredia do ktorého vyžaruje zvukové vlny[5].

3.4 Kamery

Bežné monokulárne kamery je možné použiť na získanie obrazu prostredia. Výhodou týchto kamier je ich dostupnosť a cena. Nevýhodou je absencia hĺbkovej informácie pixelov. Hĺbkovú informáciu je možné získať na základe závislosti detekovaných rysov v kamerových záberoch. Ak poznáme polohu kamery v každom zábere, je možné danú kameru v prostredí lokalizovať a dané prostredie zmapovať. Týmto problémom sa zaoberajú Monocular SLAM metódy. Tieto metódy sú podobné structure-from-motion metódam z oblasti počítačového videnia [13].

RGB-D senzor je kombináciou monokulárnej kamery a infračerveného merača vzdialenosti. To umožňuje získavať obraz prostredia spolu s informáciou o hĺbke pixelov. Oproti monokulárnym kamerám je výhodou pridaná informácia o hĺbke pixelov v obraze. RGB-D senzory používajú metódy RGB-D SLAM [14].

Stereo kamery sú inšpirované ľudským zrakom. Tieto kamery obsahujú dvojicu objektívov, ktorá sníma tú istú časť prostredia. Na základe obrazu z kamier a známej vzdialenosti a orientácií objektívov, je možné vypočítať hĺbkovú informáciu snímaných objektov [14].

3.5 LIDAR

Light Detection And Ranging (LIDAR) je metóda merania vzdialenosti medzi senzorom a prostredím pomocou laserového lúču. Vzdialenosť je možné vypočítať na základe doby od vyžiarenia laserového lúču po čas, kedy je odrazený lúč opäť zaznamenaný senzorom alebo na základe fázového posunu prijatého laserového lúču. Vzorec na výpočet vzdialenosti D je definovaný v nasledujúcej rovnici [15].

$$D = c \cdot \frac{\Delta T}{2}, \tag{3.1}$$

kde c označuje rýchlosť svetla a ΔT je čas od vyžiarenia po príjem laserového lúču.

Laserový lúč môže byť statický, alebo rozpestrený po prostredí pomocou rotujúcich zrkadiel. Pre účely mobilnej robotiky sa používa lúč druhého typu. V jednoduchších aplikáciách, kde je predpokladom pohyb po rovine, je možné použiť lacnejšie senzory. Tieto senzory poskytujú 2D horizontálny rez prostredím, ktorý je reprezentovaný ako množina bodov v polárnych súradniciach s počiatkom v polohe senzoru. Naopak výstupom 3D LIDAR senzorov je množina bodov v sférických súradniciach, taktiež s počiatkom v polohe senzoru [5].

Výhodou týchto senzorov je rýchla odozva v jednotkách hertzov a presnosť merania vzdialenosti s chybou v jednotkách centimetrov. Niektoré LIDAR senzory dokážu snímať státisíce bodov za sekundu. Kvôli týmto vlastnostiam sú LIDAR senzory používané v oblasti mobilnej robotiky, keďže umožňujú vytvárať model prostredia robota v reálnom čase [5].

Kapitola 4
ICP SLAM

4.1 ICP algoritmus

Cieľom iterative-closest-point (ICP) algoritmu je nájdenie transformácie medzi `meraním` a `referenciou`, ktorá dané bodové mraky (z anglického point cloud) najlepšie zosadí [16]. Bodové mraky popisujú neusporiadanú množinu bodov v priestore. Body majú svoje súradnice v súradnicovom systéme daného bodového mraku. Formálne [16] popisuje tento algoritmus nasledovne.

Nech tvar ${}^{\mathbb{A}}\mathcal{P}$ reprezentuje `meranie` v súradnicovom systéme $\mathbb{A}$ a tvar ${}^{\mathbb{B}}\mathcal{Q}$ reprezentuje `referenciu` v súradnicovom systéme $\mathbb{B}$. Cieľom ICP je odhad transformácie ${}^{\mathbb{B}}_{\mathbb{A}}\mathcal{T}$ minimalizáciou chybovej funkcie $error(\mathcal{P}', \mathcal{Q})$:

$$ {}^{\mathbb{B}}_{\mathbb{A}}\widehat{\mathcal{T}} = \underset{\mathcal{T}}{\operatorname{argmin}}(error(\mathcal{T}({}^{\mathbb{A}}\mathcal{P}), {}^{\mathbb{B}}\mathcal{Q})), \tag{4.1} $$

kde $\mathcal{T}({}^{\mathbb{A}}\mathcal{P})$ je aplikovanie transformácie $\mathcal{T}$ na `meranie` ${}^{\mathbb{A}}\mathcal{P}$.

Chybová funkcia je počítaná medzi pármi bodov, ktoré boli asociované medzi bodovými mrakmi. Základná asociácia je vykonaná nájdením najbližšieho bodu `merania` pre každý bod `referencie`. V ideálnom prípade združenie nastáva pre pár bodov, ktoré majú po zarovnaní `merania` a `referencie` najmenšiu vzdialenosť.

Nech $\mathcal{M} = match(\mathcal{P}, \mathcal{Q}) = \{(p, q) : p \in \mathcal{P}, q \in \mathcal{Q}\}$ je množina asociovaných párov bodov medzi $\mathcal{P}$ a $\mathcal{Q}$. Potom môže byť chybová funkcia vyjadrená ako

$$ error(\mathcal{P}, \mathcal{Q}) = \sum_{(p,q)\in\mathcal{M}} d(p, q), \tag{4.2} $$

kde $d(p, q)$ označuje funkciu popisujúcu vzdialenosť bodov p,q.

Množina $\mathcal{M}$ môže obsahovať odľahlé páry. Tieto páry sú identifikované a vymazané zo skupiny $\mathcal{M}$. Dodatočne sú zavedené váhy $\mathcal{W} = outlier(\mathcal{M}) = \{w(p, q) : \forall(p, q) \in \mathcal{M}\}$, ktoré môžu byť pridelene párom zo skupiny $\mathcal{M}$, aby zvýšili alebo znížili ich dopad v chybovej funkcii

$$ error(\mathcal{P}, \mathcal{Q}) = \sum_{(p,q)\in\mathcal{M}} w(p, q)d(p, q). \tag{4.3} $$

V prípade ideálnej asociácie bodov do množiny $\mathcal{M}$, minimalizácia chybovej funkcie produkuje najlepší odhad transformácie ${}^{\mathbb{B}}_{\mathbb{A}}\mathcal{T}$ 4.1. Obecne problém

asociácie bodov nie je možné presne vyriešiť. Hlavnou myšlienkou ICP algoritmu je to, že aj s nepresnou asociáciou bodov do skupiny $\mathcal{M}$, minimalizácia chybovej funkcie produkuje lepší odhad počiatočnej transformácie ${}^{\mathbb{B}}_{\mathbb{A}}\mathcal{T}$. To znamená, že nepresnou asociáciou bodov v iterácií $i-1$, je ICP algoritmus schopný vyprodukovať presnejšiu asociáciu bodov v iterácií i. Cieľom je teda vybudovať postupnosť transformácií ${}^{i}_{i-1}\mathcal{T}$, ktoré sú aplikované na `meranie` $\mathcal{P}$. V každej iterácií sú body asociované do skupiny $\mathcal{M}$, na základe predošlého odhadu transformácie ${}^{i}_{i-1}\mathcal{T}$. Potom, na základe aktualizovanej asociácie bodov, je vypočítaná nová transformácia ${}^{i+1}_{i}\mathcal{T}$ minimalizáciou chybovej funkcie

$$ {}^{i+1}_{i}\mathcal{T} = \underset{\mathcal{T}}{\operatorname{argmin}}(error(\mathcal{T}({}^{i}\mathcal{P}'), \mathcal{Q})). \tag{4.4} $$

Po ukončení behu algoritmu je výsledná transformácia zložená z transformácií získaných v iteráciách algoritmu

$$ {}^{\mathbb{B}}_{\mathbb{A}}\widehat{\mathcal{T}} = {}^{i}_{i-1}\mathcal{T} \circ ... \circ {}^{3}_{2}\mathcal{T} \circ {}^{2}_{1}\mathcal{T} \circ \mathcal{T}_{init}, \tag{4.5} $$

kde $\mathcal{T}_{init}$ je počiatočný odhad medzi `meraním` a `referenciou` a i označuje počet iterácií algoritmu. ICP algoritmus môže byť zastavený na základe konvergenčných kritérií alebo určením maximálneho počtu iterácií. Konvergenčným kritériom môže byť napríklad minimálna veľkosť vzdialenosti a rotácie transformačnej matice ${}^{i}_{i-1}\mathcal{T}$, teda matice získanej v jednej iterácií algoritmu. Vylepšiť robustnosť ICP algoritmu je možné predfiltrovaním bodových mrakov `merania` a `referencie`. Na výber je taktiež viacero chybových a outlier funkcií. Filtre a funkcie použité na konfiguráciu ICP SLAM sú popísané v 4.4. Tento postup je zhrnutý formou pseudokódu v Algoritme 1.

Algorithm 1 ICP

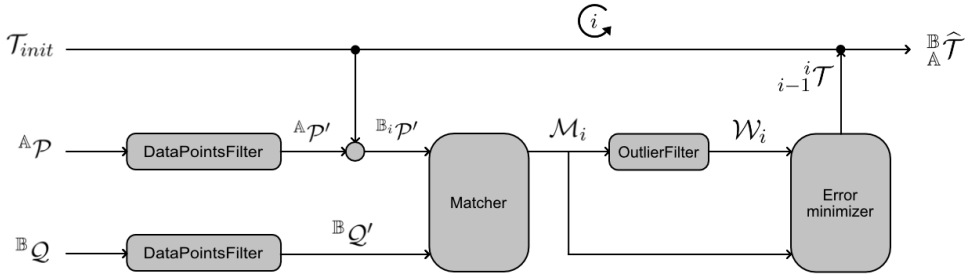
Require: ${}^{\mathbb{A}}\mathcal{P}$ ▷ `meranie`
Require: ${}^{\mathbb{B}}\mathcal{Q}$ ▷ `referencia`
Require: $\mathcal{T}_{init}$ ▷ počiatočná transformácia
  ${}^{\mathbb{A}}\mathcal{P}' \leftarrow datafilter({}^{\mathbb{A}}\mathcal{P})$ ▷ Aplikovanie filtrov bodových mrakov
  ${}^{\mathbb{B}}\mathcal{Q}' \leftarrow datafilter({}^{\mathbb{B}}\mathcal{Q})$ ▷ Aplikovanie filtrov bodových mrakov
  ${}^{i}_{i-1}\mathcal{T} \leftarrow \mathcal{T}_{init}$
  while not convergence **do**
    ${}^{\mathbb{B}_i}\mathcal{P}' \leftarrow {}^{i}_{i-1}\mathcal{T}({}^{i-1}\mathcal{P}')$ ▷ Aktualizácia `merania`
    $\mathcal{M}_i \leftarrow match({}^{i}\mathcal{P}', \mathcal{Q}')$ ▷ Asociácia bodov
    $\mathcal{W}_i \leftarrow outlier(\mathcal{M}_i)$ ▷ Filtrovanie outlierov
    ${}^{i+1}_{i}\mathcal{T} \leftarrow \underset{\mathcal{T}}{\operatorname{argmin}}(error(\mathcal{T}({}^{i}\mathcal{P}'), \mathcal{Q}))$
  end while
  ${}^{\mathbb{B}}_{\mathbb{A}}\widehat{\mathcal{T}} \leftarrow {}^{i}_{i-1}\mathcal{T} \circ ... \circ {}^{3}_{2}\mathcal{T} \circ {}^{2}_{1}\mathcal{T} \circ \mathcal{T}_{init}$

4.2 Knižnica Libpointmatcher

`Libpointmatcher`[1] je voľne dostupná knižnica, slúžiaca na registráciu mrakov bodov [17]. ICP algoritmus je rozdelený na modulárne časti, ktoré sú optimalizované, parametrizovateľné a jednoducho konfigurovateľné pomocou YAML[2] konfiguračných súborov. `Libpointmatcher` ponúka nasledujúce typy modulov:

- `DataPointsFilter` je modul, ktorý slúži na filtráciu mrakov bodov. Niektoré z týchto filtrov taktiež pridávajú informácie k bodom, napríklad normálu bodu voči povrchu.
- Modul `Matcher` slúži na asociáciu bodov z merania k referencii.
- Modul `OutlierFilter` obsahuje funkcie na vymazávanie, alebo váženie spojitostí bodov v `meraní` s ich asociovanými bodmi v `referencii`.
- `ErrorMinimizer` je modul, ktorý sa používa na výpočet transformačnej matice ${}^{\mathbb{B}}_{\mathbb{A}}\widehat{\mathcal{T}}$.
- Modul `TransformationChecker` zastavuje ICP algoritmus na základe zvoleného kritéria.

Blokové schéma ICP algoritmu z knižnice `libpointmatcher` je zobrazené na obrázku 4.1. Na začiatku algoritmu sú na `meranie` a `referenciu` aplikované filtre bodových mrakov. Po filtrácií začína cyklus ICP. Na `meranie` sa aplikuje počiatočná transformácia, poprípade nájdená transformácia z predošlého cyklu. V ďalšom kroku prebieha asociácia `merania` voči `referencii`. Outlier filtre buď vymažú, alebo zváži spojitosti medzi bodmi z `merania` a `referencie`. Následne chybová funkcia vykoná optimalizačný krok a vypočíta novú transformačnú maticu. V poslednom kroku sa skontrolujú podmienky ukončenia ICP algoritmu a podľa nich algoritmus končí, alebo sa celý cyklus opakuje znova.



Obrázok 4.1: Blokové schéma ICP algoritmu.

[1] https://github.com/ethz-asl/libpointmatcher

[2] YAML je pre človeka jednoducho čitateľný formát, slúžiaci na serializáciu dát.

4.3 ICP SLAM

Základný princíp metódy ICP SLAM bol popísaný v podkapitole 2.4.4. V tejto podkapitole je popísaná implementácia v rámci balíčku `norlab_icp_mapper`[3]. Tento balíček importuje ICP algoritmus z knižnice `libpointmatcher`. `Norlab_icp_mapper` poskytuje možnosť SLAM v reálnom čase s nastaviteľnými parametrami. Tento balíček je možné zakomponovať v robot operating system (ROS), v rámci balíčka `norlab_icp_mapper_ros`[4], ktorý slúži na interpretovanie ROS správ z a do balíčka `norlab_icp_mapper`.

Algoritmus ICP SLAM má na vstupe odhad transformácie a bodový mrak reprezentujúci `meranie`. V prípade tejto práce sú odhadom transformácie odometrické dáta a bodový mrak z LIDAR senzora. Výstupom je naopak `mapa` a aktuálna poloha robota v danej `mape`. Pre presný popis je dôležité zadefinovať si nasledujúce súradnicové systémy a transformácie medzi nimi:

1. Popis súradnicových systémov:
   - Súradnicový systém `map` reprezentuje súradnice, v ktorých je uložená výsledná `mapa`.
   - V súradnicovom systéme `odom` sú prijímané odometrické merania. V prípade tejto práce je počiatočná poloha počiatku súradnicových systémov `map` a `odom` rovnaká. Časom ICP SLAM opravuje chybu odometrie a vzájomná poloha týchto systémov sa mení.
   - Súradnicový systém `base_link` má počiatok zakotvený niekde na robotickej platforme. Ten slúži na transformovanie bodov zo senzora do súradnicového systému `odom`. Senzor má svoj vlastný súradnicový systém `lidar`.
2. Popis transformácií medzi súradnicovými systémami:
   - Transformácia ${}^{odom}_{map}\mathcal{T}$ popisuje vzájomnú polohu systémov `map` a `odom`.
   - Algoritmus prijíma odmetrické dáta ako transformáciu ${}^{base\_link}_{odom}\mathcal{T}$ popisujúcu vzájomnú polohu systémov `odom` a `base_link`.
   - Statická transformácia ${}^{lidar}_{base\_link}\mathcal{T}$ medzi systémom `base_link` a `lidar` popisuje vzájomnú polohu senzoru voči robotickej platforme.

Na začiatku algoritmu sú na `meranie` ${}^{lidar}\mathcal{P}$ aplikované filtre bodových mrakov. `Referencia` je v tomto prípade `mapa` ${}^{map}\mathcal{M}$ prostredia. `Mapa` sa v prvej iterácií inicializuje ako bodový mrak `merania` a výsledkom prvej lokalizácie je odometrické meranie. V prípade inicializovanej `mapy` je `meranie` transformované zo systému `lidar` do systému `map` pomocou transformácie

$$ {}^{lidar}_{map}\mathcal{T} = {}^{lidar}_{base\_link}\mathcal{T} \circ {}^{base\_link}_{odom}\mathcal{T} \circ {}^{odom}_{map}\mathcal{T}. \tag{4.6} $$

[3] https://github.com/norlab-ulaval/norlab_icp_mapper
[4] https://github.com/norlab-ulaval/norlab_icp_mapper_ros

Počiatočný odhad medzi `meraním` a `mapou` je zakomponovaný v transformovaní bodového mraku merania do systému `map`. Toto `meranie` je spolu s `mapou` následne použité v ICP algoritme za účelom získania korekčnej transformácie $\mathcal{T}_c$ z rovnice 4.1. Kombináciou korekčnej transformácie $\mathcal{T}_c$ a transformácie ${}^{lidar}_{map}\mathcal{T}$ získame lokalizovanú polohu senzora v systéme `map` formou transformácie

$${}^{lidar}_{map}\widehat{\mathcal{T}} = \mathcal{T}_c \circ {}^{lidar}_{map}\mathcal{T}. \tag{4.7}$$

Podľa tejto transformácie je `meranie` pridané do `mapy` a na `mapu` sú aplikované filtre. Následne je upravená transformácia

$${}^{odom}_{map}\mathcal{T} = {}^{lidar}_{map}\widehat{\mathcal{T}} \circ {}^{lidar}_{base\_link}\mathcal{T}^{-1} \circ {}^{base\_link}_{odom}\mathcal{T}^{-1}. \tag{4.8}$$

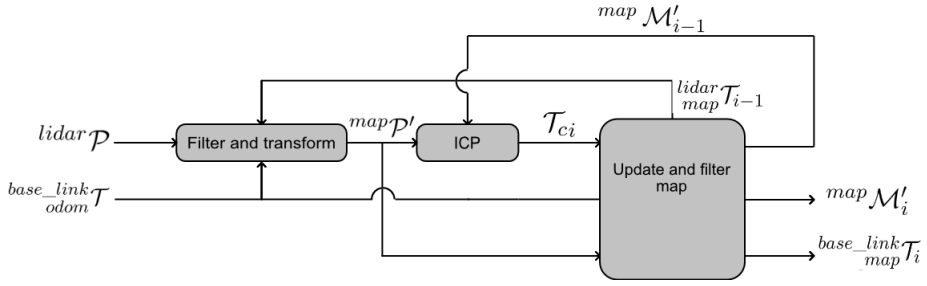
Reálna poloha robota v priestore je popísaná transformáciou

$${}^{base\_link}_{map}\mathcal{T} = {}^{base\_link}_{odom}\mathcal{T} \circ {}^{odom}_{map}\mathcal{T}. \tag{4.9}$$

Úlohou transformácie ${}^{odom}_{map}\mathcal{T}$ je korekcia chyby v odometrickom meraní. Proces algoritmu je priblížený na obrázkoch 6.11 a popísaný formou pseudokódu v algoritme 2. Zreťazenie operácií je vizualizované v diagrame 4.2.

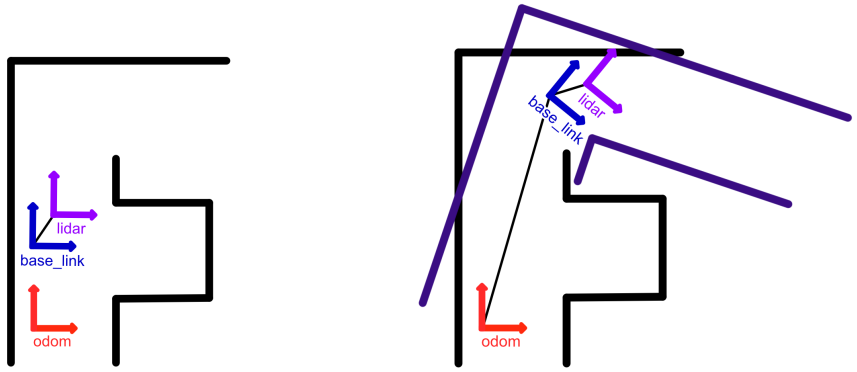
Algorithm 2 Iterácia ICP SLAM v reálnom čase

Require: ${}^{lidar}\mathcal{P}$ ▷ `meranie` zo senzora
Require: ${}^{base\_link}_{odom}\mathcal{T}$ ▷ odometria
${}^{lidar}\mathcal{P}' \leftarrow datafilter({}^{lidar}\mathcal{P})$ ▷ Filtrovanie.
${}^{lidar}_{odom}\mathcal{T} \leftarrow {}^{lidar}_{base\_link}\mathcal{T} \circ {}^{base\_link}_{odom}\mathcal{T}$
${}^{map}\mathcal{P}' \leftarrow {}^{lidar}\mathcal{P}' \circ {}^{lidar}_{odom}\mathcal{T} \circ {}^{odom}_{map}\mathcal{T}_{i-1}$ ▷ Transformácia merania.
$\mathcal{T}_{ci} \leftarrow icp({}^{map}\mathcal{P}', {}^{map}\mathcal{M}'_{i-1})$
${}^{map}\mathcal{M}_i \leftarrow updateMap(\mathcal{T}_{ci}, {}^{map}\mathcal{P}', {}^{map}\mathcal{M}'_{i-1})$ ▷ Aktualizácia mapy
${}^{map}\mathcal{M}'_i \leftarrow datafilter({}^{map}\mathcal{M}_i)$
${}^{lidar}_{map}\mathcal{T}_i \leftarrow \mathcal{T}_{ci} \circ {}^{lidar}_{map}\mathcal{T}_{i-1}$
${}^{odom}_{map}\mathcal{T}_i \leftarrow {}^{lidar}_{map}\mathcal{T}_i \circ {}^{lidar}_{odom}\mathcal{T}^{-1}$
${}^{base\_link}_{map}\mathcal{T}_i \leftarrow {}^{base\_link}_{odom}\mathcal{T} \circ {}^{odom}_{map}\mathcal{T}$ ▷ Lokalizácia polohy
return $({}^{map}\mathcal{M}'_i, {}^{base\_link}_{map}\mathcal{T}_i)$



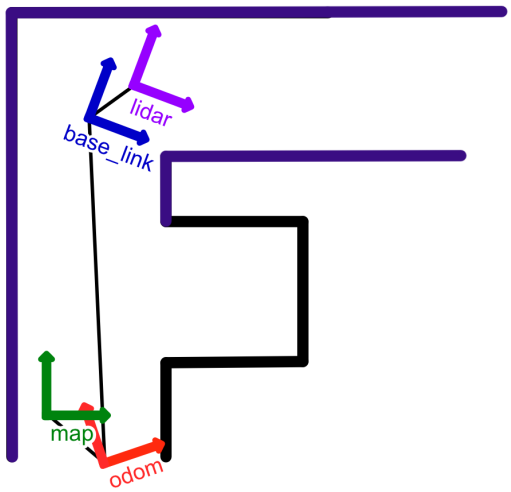
Obrázok 4.2: Blokové schéma ICP SLAM.

Balíček `norlab_icp_mapper` filtruje bodový mrak `mapy` v súradnicovom systéme senzora. Kvôli tomu je možné zachovať si lokálnu `mapu` prostredia v okolí robota, ktorý sa pohybuje.



(a) : Robot v prostredí.

(b) : Robot po vykonaní pohybu.



(c) : Zarovnanie merania a mapy pomocou ICP.

Obrázok 4.4: Vyzualizácia krokov metódy ICP SLAM - na obrázku 4.3a sa robot nachádza v prostredí. Čierne čiary popisujú kontúry zmapovaného prostredia. Súradnicové systémy `map` a `odom` majú v tomto momente rovnaký počiatok. Na obrázku 4.3b robot vykoná pohyb. Fialové čiary popisujú `meranie` zo senzora a čiernou čiarou je popísané odometrické meranie, medzi súradnicovým systémom `odom` a `base_link`. Na obrázku 4.3c je `meranie` zarovnané do `mapy` a robot je v `mape` lokalizovaný. Keďže bolo odometrické meranie nepresné, súradnicové systémy `map` a `odom` už nemajú rovnaký počiatok. Posun súradnicového systému `odom` od súradnicového systému `map` kompenzuje chybu odometrického merania.

4.4 Filtre a funkcie modulov ICP

V tejto kapitole budú popísané filtre bodových mrakov, chybové funkcie a outlier funkcie, ktoré boli použité v konfigurácií ICP SLAM v rámci praktickej časti. Filtrovanie bodových mrakov je dôležité, keďže zníženie počtu bodov v bodových mrakoch výrazne napomáha rýchlosti ICP algoritmu [18].

K-D Tree štruktúra

K-D strom je dátová štruktúra slúžiaca na ukladanie viacrozmerných bodových mrakov v priestore. Vstupom je bodový mrak s k rozmermi. K-D strom je zložený z uzlov, ktoré je možné rozdeliť na ľavý a pravý podstrom. Tieto podstromy môžu byť rekurzívne delené rovnakým spôsobom. Každý uzol v K-D strome popisuje k-rozmerný obdĺžnik. Pre K-D strom a jeho podstromy, platia nasledujúce podmienky [19]:

- Ak ľavý podstrom nie je prázdny, tak hodnota všetkých uzlov v ľavom podstrome nie je vačšia ako hodnota rodičovského uzla.
- Ak pravý podstrom nie je prázdny, tak hodnota všetkých uzlov v pravom podstrome je vačšia ako hodnota rodičovského uzla.
- Ľavý a pravý podstrom je taktiež K-D strom.

K-D strom umožňuje rýchle hľadanie najbližších susedných bodov, preto je táto dátová štruktúra kľúčová v implementácií ICP a filtrov bodových mrakov.

4.4.1 DataPointsFilter

Väčšina filtrov slúži na odstránenie nežiadúcich bodov z bodového mraku. Niektoré filtre pridávajú bodom dodatočnú informáciu. Dotatočnou informáciou môžu byť vlastosti bodu v bodovom mraku nazývané deskriptory. Príkladom deskriptora môže byť odhadovaná hustota bodového mraku v okolí určitého bodu.

FixStepSamplingDataPointsFilter

Tento filter ponecháva vo výstupnom bodovom mraku každý n_f-tý bod, kde n_f je nastaviteľný parameter. Tento filter je vhodný na filtráciu bodového mraku zo senzorov, ktorých meranie má určitý geometrický tvar.

BoundingBoxDataPointsFilter

V priestore je parametrami $b = [x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}]$ definovaný kváder. V závislosti na konfigurácií sú body vymazané, ak sa nachádzajú v danom kvádri alebo mimo neho. Tento filter je používaný najmä na odstránenie bodov, ktoré vznikli nasnímaním konštrukcie robota. Parametrami sú

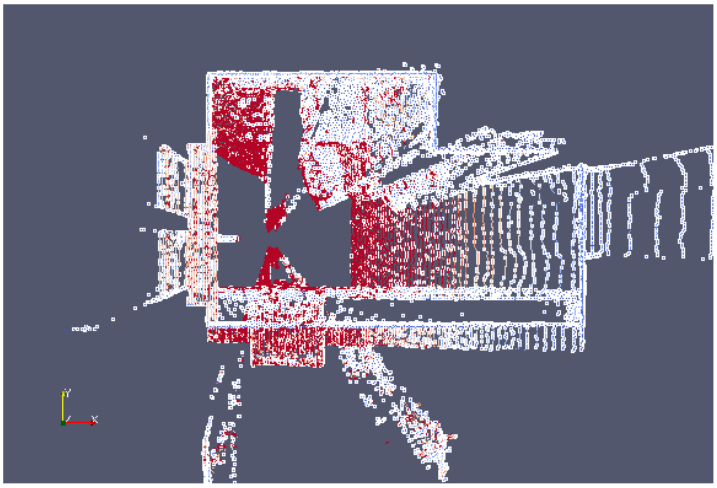
tri dvojice súradníc pre každú os v priestore, kde každá dvojica vyčleňuje minimálnu a maximálnu hodnotu súradnice v danej ose. Posledný parameter konfiguruje vymazávanie bodov z kvádra, alebo z jeho vonkajšku.

RandomSamplingDataPointsFilter

Tento filter priradí každému bodu parametrom p_r definovanú pravdepodobnosť vymazania bodu z bodového mraku. Následne sú tieto body s danou pravdepodobnosťou vymazané z bodového mraku.

MaxDensityDataPointsFilter

Na funkciu tohto filtra je potrebné, aby body obsahovali desktriptor hustoty. Vstupným parametrom ρ je maximálna povolená hodnota danej hustoty. Ak je hustota v okolí bodu väčšia ako tento parameter, bod je z bodového mraku odstránený. Oblasti bodového mraku z vysokou hustotou obsahujú často prebytočnú informáciu o prostredí. Funkcia filtra je zobrazená na obrázku 4.6.



Obrázok 4.5: Aplikovanie filtra MaxDensityDataPointsFilter - pri filtrovaní bola stanovená maximálna hustotou na hodnotu 50000 bodov na m^3. Červenou farbou je označený pôvodný bodový mrak a bielou farbou je označený bodový mrak po filtrovaní [20].

SurfaceNormalDataPointsFilter

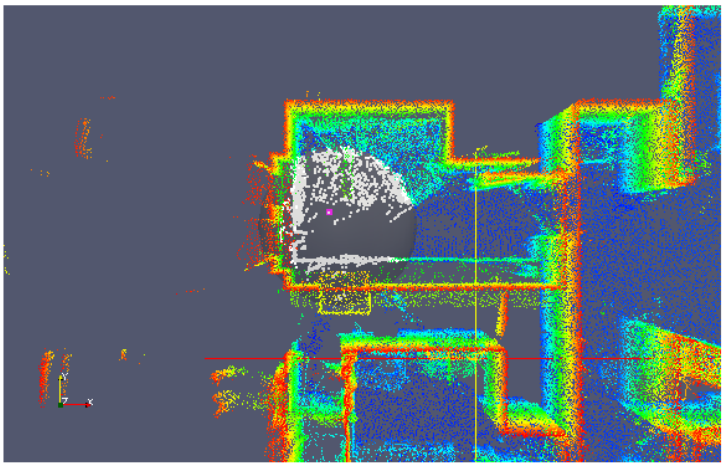
Hlavnou funkciou tohto filtra je výpočet normály voči povrchu. Parametrom *knn* je možné určiť počet susedných bodov použitých k danému výpočtu. Parametrom *keep_densities* je taktiež možné aktivovať výpočet hustoty v okolí každého bodu v bodovom mraku.

OrientNormalsDataPointsFilter

Susedné normály voči povrchu nemusia mať nutne rovnakú orientáciu. Parametrom *towardCenter* môžeme určiť obmedzujúcu podmienku, podľa ktorej sú normály preorientované voči počiatku súradnicového systému alebo opačne. Týmto je možné dosiahnuť konzistentné normály na rovnakom povrchu.

DistanceLimitDataPointsFilter

Parametrom *dist* je možne určiť maximálnu vzdialenosť bodov od počiatku v bodovom mraku. Taktiež je možné špecifikovať filtrovanie v osiach súradnicového systému. Ak má bod vzdialenosť od počiatku väčšiu ako maximálnu povolenú, je z bodového mraku odstránený. Filter je taktiež možné parametrizovať na odstránenie bodov, ktoré maximálnu povolenú vzdialenosť neprekračujú. Funkcia filtra je zobrazená na obrázku [20].



Obrázok 4.6: Aplikovanie filtra DistanceLimitDataPointsFilter - z bodového mraku boli odstránené všetky body vzdialené od počiatku na 1 m. Bielou farbou je označený bodový mrak po filtrovaní [20].

CutAtDescriptorThresholdDataPointsFilter

V tomto filtri je možné odstrániť body z bodového mraku, na základe hodnoty desktriptorov. V parametroch sa špecifikuje meno desktriptora *descriptor* a maximálna $d_{t_{max}}$ alebo minimálna $d_{t_{min}}$ hodnota daného deskriptora. Ak je hranica prekročená, bod je z bodového mraku odstránený.

4.4.2 Matcher

Tento modul obsahuje funkciu KDTreeMatcher. Táto funkcia spracováva bodový mrak do dátovej štruktúry KD-tree. Parametrami sa špecifikuje počet n_m najbližších susedov a ich maximálna vzdialenosť d_{max} od bodu, z ktorého sú daní susedia hľadaní. Funkcia poskytuje zoznam najbližších susedov pre každý bod z bodového mraku.

4.4.3 Outlier filtre

Outlier filtre sú funkcie, ktoré na základe vzdialeností spárovaných bodov z množiny $\mathcal{M}$ produkujú váhy, ktoré zvyšujú alebo znižujú ich dopad v chybovej funkcii 4.3. Keďže ICP používa metódu najmenších štvorcov, vplyv jednotlivých párov v chybovej funkcií narastá z ich vzdialenosťou. Skupina outlierov s dostatočne veľkou vzdialenosťou môže prevážiť vplyv správne asociovaných bodov v ICP algoritme [21].

RobustOutlierFilter

V oblasti štatistiky boli vytvorené metódy na minimalizovanie vplyvu outlierov na ICP algoritmus. M-estimator je estimátor založený na metóde maximálnej vierohodnosti. M-estimátor je bežne používaný na filtrovanie outlierov pri používaní metódy najmenších štvorcov [22]. Jedným z týchto estimátorov je Chauchyho M-estimator. Cauchyho estimátor pridáva váhu w každému páru (p, q) z množiny $\mathcal{M}$ podľa nasledujúceho vzorca

$$w = \frac{1}{1 + \frac{error(p,q)^2}{k^2}}, \tag{4.10}$$

kde k je parameter, ktorý slúži na ladenie tejto funkcie.

SurfraceNormalOutlierFilter

Tento filter odstraňuje vplyv bodov z bodového mraku `merania`, pri ktorých uhol medzi normálovým vektorom daného bodu a normálovým vektorom spárovaného bodu z `referencie` prekračuje parametrom ϕ_{sn} stanovenú hodnotu. Keďže outlier filtre stanovujú váhy daných dvojích bodov (p,q), pod odstranením vplyvu bodov je myslené stanovenie hodnoty váhy na nulu.

4.4.4 ErrorMinimizer

Funkcie v module `ErrorMinimizer` stanovujú tvar chybovej funkcie $error(\mathcal{P}, \mathcal{Q})$. Cieľom minimalizácie chybovej funkcie je vyriešenie rovnice 4.4.

Chybová funkcia point-to-plane

Veľkosť chyby point-to-plane chybovej funkcii naprieč celou množinou $\mathcal{M}$ je definovaný ako

$$error(\mathcal{P}, \mathcal{Q}) = \sum_{k=1}^{K} w(p_k, q_k) \|(p_k - q_k) \cdot n_k\|, \tag{4.11}$$

kde K popisuje počet asociovaných dvojíc v množine $\mathcal{M}$, n_k je normálový vektor v bode q_k a $\|.\|$ označuje L2 normu [23]. Chybovú funkciu je možné normalizovať pomocou estimátorov veľkosti. Jednou z možností je medián absolútnych odchyliek vzdialeností dvojíc v množine $\mathcal{M}$ [16].

4.4.5 TransformationChecker

Ako už bolo spomenuté, tento modul obsahuje funkcie na zastavenie ICP na základe zvoleného kritéria. Funkcia `CounterTransformationChecker` zastaví ICP algoritmus po prekročení parametrom i_{max} nastaviteľného maximálneho počtu iterácií. `DifferentialTransformationChecker` zastaví algoritmus, ak rotačný a translačný komponent transformačnej matice ${}^{i+1}_{i}\mathcal{T}$ nedosiahne parametrami $\varepsilon_{\theta_{min}}$, $\varepsilon_{t_{min}}$ stanovene minimá. Parametrami vieme nastaviť požadovanú presnosť zosadenia `merania` a `referencie`. `BoundTransformationChecker` naopak zastaví ICP v prípade, že rotačný alebo translačný komponent transformačnej matice ${}^{i+1}_{i}\mathcal{T}$ presiahne parametrami $\varepsilon_{\theta_{max}}$, $\varepsilon_{t_{max}}$ stanovené maximum.

4.5 Filtrácia dynamických prekážok v mape

Predpokladom práce bolo statické vonkajšie prostredie. Avšak skúšobne jazdy boli vykonané v mestských parkoch, kde sa pohybovali ľudia, autá a iné dynamické objekty. Bez filtrovania by sa do `mapy` permanentne dostával šum od pohybujúcich sa objektov. Balíček `norlab_icp_mapper` obsahuje funkcionalitu, ktorá každému bodu v `mape` pridá desktriptor *probDynamic*, ktorý popisuje pravdepodobnosť toho, že daný bod je dynamický. Výpočet tejto pravdepodobnosti je implementovaný podľa metódy predstavenej v článku [24]. Na výpočet tejto pravdepodobnosti je použitý princíp ray-tracingu. Ak je z perspektívy senzora bod `merania` identifikovaný za bodom `mapy`, pravdepodobnosť toho, že bod je dynamický, sa zvýši. Následne je možné odstrániť body, pre ktoré táto pravdepodobnosť prekročila stanovenú hranicu. Pre popísanie výpočtu je potrebné definovať si nasledujúce premenné:

- p: bod z `merania` v súradnicovom systéme senzora,
- q: bod z `referencie` asociovaný k bodu p v súradnicovom systéme senzora,
- Dyn: binárna premenná, ktorá označuje dynamický alebo statický stav bodu q,
- $Odyn$: binárna premenná, ktorá označuje dynamický alebo statický stav bodu q v minulosti,
- U: binárna premenná, označujúca potrebu zmeny stavu bodu,
- θ: uhol medzi bodmi p, q definovaný ako $\arccos(\frac{p \cdot q}{\|p\| \cdot \|q\|})$,
- ϕ: uhol dopadu na q definovaný ako $\arccos(n \cdot \frac{q}{\|q\|})$, kde n je normála voči ploche v bode q,
- δ: vzdialenosť medzi bodmi p q definovaná ako $\|p - q\|$.

Môžeme zapísať:

$$P(Dyn|\theta,\phi,\delta) \propto, \sum_{UOdyn} \left| \begin{matrix} P(Odyn)P(U|\theta,\phi) \\ \times P(Dyn|Odyn)P(\delta|Dyn) \end{matrix} \right. \tag{4.12}$$

Kde:

- $P(Odyn)$,: je buď počiatočná pravdepodobnosť alebo výsledok predošlého výpočtu,
- $$P(U|\theta,\phi) = \begin{cases} \frac{\theta}{\theta_{\max}}(1-\frac{2\phi}{\pi}) & \text{ak } \|p\| \geq \|q\| \\ 0 & \text{inak} \end{cases} \tag{4.13}$$

  je pravdepodobnosť zmeny stavu založená na uhloch ϕ a θ,
- $$P(Dyn|Odyn) = \begin{bmatrix} \alpha & 1-\beta \\ 1-\alpha & \beta \end{bmatrix} \tag{4.14}$$

  je matica, ktorá umožňuje zmenu stavu bodu,
- $$P(\delta|Dyn) = \begin{cases} \max(\mathrm{O}, \min(1, \epsilon_d + \|p\|\epsilon_a - \delta)) \\ \text{ak je } Dyn \text{ nepravdivé} \\ 1 - P(\delta|Dyn = \text{nepravdivé}) \\ \text{ak je } Dyn \text{ pravdivé} \end{cases} \tag{4.15}$$

  je pravdepodobnostné rozdelenie pozorovania.

Tento výpočet je možné parametrizovať. Parameter $\theta_{\max}$ určuje uhol, podľa ktorého je vymedzený priestor. Body `referencie` z tohto priestoru sú následne asociované k bodu p. Tento uhol je založený na rozptyle laserového lúču. Hĺbkový šum merania bodu p je možné zohľadniť parametrom e_d a parametrom e_a je popísaný proporcionálny hĺbkový šum merania bodu p. Parameter α popisuje pravdepodobnosť toho, že statický bod ostane statickým. Naopak parameter β popisuje pravdepodobnosť toho, že dynamický bod ostane dynamickým. Posledným parametrom γ_{max} je označená hranica pravdepodobnosti, po ktorej je bod permanentne považovaný za dynamický. Tento model pri výpočte zohľadňuje viacero meraní.

Kapitola 5

Mapovanie priechodnosti terénu

Aby sa mohla konkrétna robotická platforma pohybovať v prostredí, je nutné analyzovať priechodnosť prostredia s ohľadom na konkrétnu robotickú platformu. Do úvahy je potrebné zakomponovať dynamické a kinematické vlastnosti danej platformy. Najbežnejším spôsobom je implementácia 2.5D mapy priechodnosti, kde každá bunka 2D mriežky obsahuje informáciu o priechodnosti danej bunky, poprípade cenu alebo pravdepodobnosť priechodnosti danej bunky. Predtým ako môže byť priechodnosť terénu vyhodnotená, musí byť daný terén interpretovaný pomocou senzorov. Interpretácia terénu môže byť rôzna z ohľadom na zameranie riešeného problému [25].

5.1 Proprioceptívná analýza priechodnosti

Analýza priechodnosti založená na princípe proprioceptívnych senzorov, je užitočná v učení modelu, ktorý obsahuje zachytenú náročnosť pri pohybe terénom. Náročnosť je analyzovaná na základe senzorických meraní z proprioceptívnych senzorov ako napríklad vibrácie, nárazy, zmeny rýchlosti robota a podobne [26]. Problémom tejto analýzy je to, že robot musí vykonať jazdu terénom, aby mohol posúdiť jeho priechodnosť. To vyžaduje mimoriadne robustné roboty, ktoré sú schopné odolať opotrebovaniu a poškodeniu, ktoré je spôsobene zlyhaniami pri jazde ako napríklad nárazy robota. Z tohto dôvodu táto analýza nie je veľmi bežná a uprednostňuje sa analýza pomocou extroceptívnych senzorov. Extroceptívnu analýzu je možné skombinovať na predčasný odhad priechodnosti, ktorý môže byť následne pri prejazde terénom vylepšený proprioceptívnou analýzou.

5.2 Extroceptívna analýza priechodnosti

Extroceptívna analýza používa extroceptívne senzory na mapovanie priechodnosti terénu. Spôsoby interpretácie senzorických meraní je možné rozdeliť na postupy založené na interpretácií geometrických charakteristikách terénu alebo postupy ktoré analyzujú priechodnosť na základe rysov a vzhľadu prostredia (v angličtine Appearance-based) [27].

5.2.1 Geometricka analýza priechodnosti

Geometrická analýza priechodnosti posudzuje priechodnosť terénu na základe extrahovaných geometrických vlastností z prostredia. Podľa autorov článku [27] základné myšlienky geometrickej analýzy dobre popisujú nasledujúce práce. Autori v článku [28] vytvorili 2D mriežkové mapy, kde každá bunka obsahovala informáciu o maxime, minime, výške, rozptylu a sklonu prostredia v tejto oblasti. Tieto vlastnosti boli vypočítane z bodového mraku rozdeleného do mriežkovej mapy. Hodnoty týchto vlastnosti boli následné porovnávane so stanovenými limitami a ak boli limity prekročené, dané bunky boli vyhodnotené ako nepriechodné. V článku [29] bola priechodnosť vyhodnocovaná na základe váhovej funckie, ktorá brala do úvahy výšku, sklon, drsnosť a presnosť mapy.

V prostredí je taktiež možné identifikovať základne tvary ako hrany, plochy a zhluky bodov. Na základe detekcií týchto tvarov je následne využiť algoritmy, ktoré odhadnú priechodnosť prostredia na základe polohy daných tvarov a charakteristík robota. V práci [30] autori klasifikujú bodový mrak z LIDAR senzora na triedu rozptylu, ktorou sú reprezentované objekty ako tráva a vegetácia, na lineárnu triedu, ktorá popisuje objekty ako káble, alebo konáre stromov a na triedu plochy, ktorou sú popísané povrchy ako zem, kamene a podobne [27].

Častou komplikáciou je analýza priechodnosti v oblastiach, v ktorých nie sme schopný zachytiť senzorické merania. Príkladom týchto oblastí sú rôzne svahy, medzery a klesajúce schody. V práci [31] bol tento problém riešený detekciou prekážok rozdelených do nasledujúcich skupín: Pozitívne prekážky boli detekované v prípade, že rozptyl bodov v danej ploche prekročil stanovené limity. Ostré hrany boli detekované na základe porovnávaní výšky medzi susednými bunkami. Následne plochy dostatočnej veľkosti, ktoré neobsahovali žiadne body, boli vyhodnocované ako potenciálne negatívne prekážky. Filtrovacím procesom bolo rozhodnuté o tom, či sa jedná o negatívnu prekážku alebo o tienenie pozitívnej prekážky [27].

Ďalším spôsobom analýzy je porovnávanie 3D modelu prostredia s 3D modelom robota, ktorý sa má po danom prostredí pohybovať. V článku [32] autori rozdelili bodovú mapu na uzly a hrany. Robota modelovali ako kváder a jeho kolesá ako válce. Následne detekovali všetky uzly v mape, v ktorých by mohol byť robot položený bez kolízie. Tieto uzly boli vyhodnotené ako prejazdné a hľadanie cesty spočívalo v jazde po hranách, ktoré spájali prejazdné uzly.

5.2.2 Vzhľadová analýza priechodnosti

Vzhľadová analýza priechodnosti spadá pod oblasť počítačového videnia. Roboti ktorý používajú túto analýzu zvyčajne klasifikujú zvolené triedy terénu. Autori v článkoch [33] a [34] analyzovali priechodnosť prostredia meraním drsnosti, sklonu, tvrdosti a medzier v danom prostredí. Tieto vlastnosti boli extrahované z obrázkov. Sklon prostredia odhadovala natrénovaná neuronová sieť, ktorá bola naučená reagovať na pixely okolo horizontov. Medzery slúžili na detekciu útesov a roklín, na základe vzdialenosti medzi detekovanými hranami. Tvrdosť bola použitá na klasifikovanie rôznych materiálov (piesok, štrk a podobne) a podieľala sa na analýze priechodnosti ako vlastnosť, ktorá popisuje potenciálne šmyky kolies. Na základe týchto vlastností bola nasledovne vyhodnotená priechodnosť daného prostredia [27].

5.3 Hybrídna analýza priechodnosti

Hybridná analýza spočíva v kombinácií metód z proprioceptívnej a extroceptívnej analýzy. Autori v článku [35] kombinujú merania z kamery a LIDAR senzora. Z obrázkov z kamery je vytvorená prvá mapa, pomocou extrahovania rysov prostredia. Prostredie je klasifikované na hladké prostredie, drsné prostredie, prekážku a pozadie. Následne je analyzovaný bodový mrak mapy z LIDAR senzora. Z bodového mraku je vytvorená druhá mapa s informáciou o sklone, výške a drsnosti prostredia. Tieto mapy sú následne zlúčené do finálnej mapy priechodnosti, pomocou použitia Bayesového filtra.

V článku [36] autori skombinovali vzhľadovú a proprioceptívnu analýzu. Najprv sú pomocou RGB-D kamery extrahované geometrické vlastnosti prostredia (sklon, drsnosť) a dané prostredie je taktiež klasifikované do tried (vegetácia, vodné plochy). Na základe týchto vlastnosti je vytvorená mapa priechodnosti. Následne pomocou sondy, obsahujúcej senzor sily, je odhadovaná poddajnosť prekážok v prostredí. Keďže sondovanie je časovo náročné, prebieha len v predom identifikovaných oblastiach. Týmto spôsobom je možné odhaliť šum spôsobený vegetáciou a taktiež overiť hĺbku vodných povrchov.

5.4 Navrhovaná analýza priechodnosti

Analýzu priechodnosti prostredia v tejto práci budem vykonávať na základe bodového mraku mapy z ICP SLAM, pomocou geometrickej analýzy. Tento bodový mrak bude najprv predfiltrovaný a rozdelený do buniek 2D mriežky. V prvom kroku budú identifikované potenciálne priechodné bunky. Pre každú bunku, ktorá obsahuje nejaké body $p = [x, y, z] \in \mathcal{C}$, bude vypočítaná priemerná výška bodov h a ich rozptyl σ^2 od výšky h podľa nasledujúcich vzorcov:

$$h = \frac{1}{K} \cdot \sum_{k=1}^{K} z_{p_k}, \tag{5.1}$$

$$\sigma^2 = \frac{1}{K} \cdot \sum_{k=1}^{K} (z_{p_k} - h)^2, \tag{5.2}$$

kde K je počet bodov v množine $\mathcal{C}$ a z_{p_k} reprezentuje výšku bodu k z množiny $\mathcal{C}$. Bunky, ktorých rozptyl σ^2 presiahol stanovený limit rozptylu σ^2_{max}, sú klasifikované ako nepriechodné. Pre bunky, ktorých rozptyl σ^2 je v rámci stanoveného limitu σ^2_{max}, bude vykonaný druhý krok. V tomto kroku bude porovnávaný výškový rozdiel h_Δ pre každú susednú bunku vypočítaný ako

$$h_\Delta = |h_c - h_n|, \tag{5.3}$$

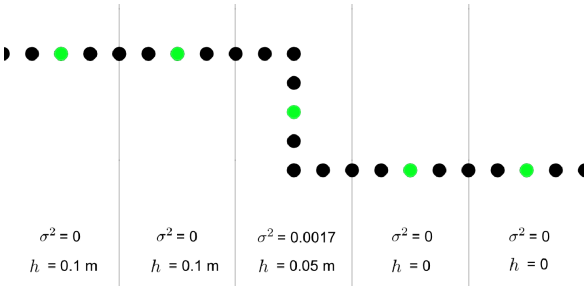
kde h_c je priemerná výška práve vyhodnocovanej bunky a h_n je výška susednej bunky. V prípade, že hodnota h_Δ je väčšia ako stanovený limit h_{max}, bunka je vyhodnotená ako nepriechodná. Výškový limit h_{max} je určený nasledujúcim vzťahom

$$h_{max} = l \cdot tan(\alpha) \tag{5.4}$$

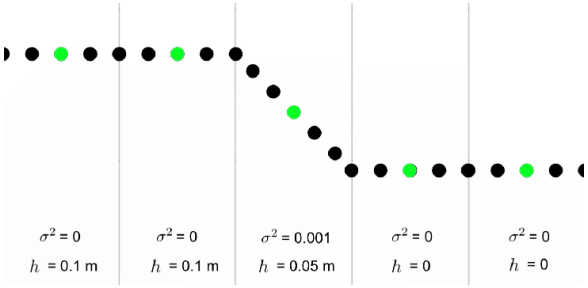
kde l je dĺžka hrany bunky a α je maximálny uhol, pod ktorým môže robot jazdiť. Pseudokód popisujúci vyhodnotenie buniek v mape priechodnosti je popísaný v algoritme 3.

Algorithm 3 Analýza priechodnosti jednotlivých buniek

```
Require: map                                   ▷ mriežka obsahujúca bunky
Require: h_max                                 ▷ výškový limit
Require: σ²                                    ▷ limit rozptylu
  for each cell in map do
    cell_h = (1/K) · Σ_{k=1}^{K} z_{p_k}
    cell_σ² = (1/K) · Σ_{k=1}^{K} (z_{p_k} − h)²
    if cell_σ² > σ²_max then
      assume cell is traversable
      for each neighbour around cell do
        h_Δ = |h_c − h_n|                      ▷ Výpočet výškového rozdielu
        if h_Δ > h_max then
          cell not traversable
          break
        end if
      end for
    else
      cell not traversable
    end if
  end for
```



(a) : Bodový mrak reprezentujúci obrubník.



(b) : Bodový mrak reprezentujúci šikmú plochu.

Obrázok 5.1: Teoretická analýza priechodnosti - na obrázkoch je možné pozorovať rozdielne hodnoty priemernej výšky a rozptylu, pre dva druhy prekážok. V tomto prípade je na základe rozptylu možné rozhodnúť o priechodnosti prekážok rozdielne, keďže rozptyl prekážky na obrázku 5.1 je takmer o polovicu menší oproti prekážke na obrázku 5.1a. Sivé vertikálne čiary predstavujú hranice jednotlivých buniek.

Porovnávaním výškových rozdielov susedných buniek sú v podstate vypočítané gradienty medzi bunkami. Výšky jednotlivých buniek sú vypočítané z bodov v oblasti určenej veľkosťou jednotlivých buniek v mriežke. Týmto procesom sú z bodovej mapy extrahované informácie, ktoré sú priemerom hodnôt výšky bodov v určitej oblasti. To znamená, že dve bunky s rovnakou priemernou výškou, môžu mať v realite úplne iný geometrický tvar. Zavedením rozptylu je naopak extrahovaná informácia o podobnosti meraní v spomínanej oblasti. V ideálnom prípade by bol rozptyl bodov vypočítaný voči ploche tvorenej bodmi v určitom okolí bunky. V tejto práci som sa tento výpočet rozhodol zjednodušiť. Rozptyl je počítaný voči horizontálnej ploche, keďže s robotom neplánujem jazdiť pod veľkým uhlom. Kombináciou gradientov a rozptylov buniek je možné vytvoriť lepší odhad priechodnosti terénu. V prípade, že gradienty medzi susednými bunkami sú v rámci limitu h_{max} ale rozptyl jednotlivých buniek je pomerne veľký, je možné usúdiť, že pri výpočte priemernej výšky jednotlivých buniek došlo k vyhladeniu terénu z bodovej mapy. Tento prípad môže nastať na ostrých prekážkach, ako sú napríklad obrubníky, v prípade, kedy hrana obrubníku prechádza stredom bunky. Tento prípad je teoreticky vizualizovaný na obrázku 5.1. Naopak, na základe príliš veľkých gradientov medzi susednými bunkami, ktorých rozptyl je pod limitom σ^2, je taktiež možné rozhodnúť o nepriechodnosti bunky. Príkladom by bol obrubník ktorého hrana by sa nachádzala na hrane dvoch susedných buniek.

Nevýhodou výpočtu priemernej výšky buniek je strata podrobnejšej informácie o geometrií terénu v danej oblasti. Výhodou výpočtu priemernej výšky je potláčanie šumu v bodovom mraku mapy. Naopak, výpočet rozptylu je na tento šum citlivý. Predpokladom pre použitie rozptylu na analýzu priechodnosti je dostatočné súvislá bodová mapa, ktorá reprezentuje plochy v teréne z dostatočnou presnosťou.

Ako senzor taktiež použijem polohu robota v bodovej mape. Pokiaľ sa bude bunka bez bodov nachádzať v určitej blízkosti robota, bude vyhodnotená ako nepriechodná. Myšlienkou je takto vyradiť rôzne medzery a diery v prostredí, ktoré LIDAR senzor nedokázal zachytiť pri jazde k danej prekážke. Priechodnosť týchto buniek sa opäť vyhodnotí v prípade pridania bodov do danej bunky.

Metóda je navrhovaná pre robota HUSKY A200. Tento robot sa po prostredí pohybuje pomocou kolies. K tejto vlastnosti sú prispôsobené hranice rozptylu σ^2 a výškového limitu h_{max}. S určitým ladením hraníc by bolo možné použiť túto metódu pre pásové roboty. Obecne sú pásové roboty schopné prekonať náročnejší terén. Metóda nebude vhodná pre roboty, ktoré sa prostredím pohybujú pomocou ramien. Klasifikácia nepriechodných buniek, konfigurovaná pre robota HUSKY A200, by pre pásové a chodiace roboty produkovala príliš veľa falošných negatívov, teda nepriechodných buniek ktoré sú v realite pre tieto roboty prejazdné.

Kapitola 6
Implementácia

6.1 Robot operating system

Robot operating system (ROS) je súbor nástrojov a knižníc, používaných vo vývoji robotiky. ROS je voľne dostupný a vznikol na Standfordskej univerzite v roku 2007. Pre rôzne robotické aplikácie je potreba zabezpečiť funkciu senzorov a komunikáciu medzi senzormi, aktuátormi a rôzným softvérom, ktorý je na danú aplikáciu vyvinutý. Vo výskume teda často nastávalo nezávisle implementovanie algoritmov s rovnakou funkciou, napríklad algoritmus na komunikáciu z určitým senzorom. Hlavnou myšlienkou ROS je poskytnutie prostredia, v ktorom je možné dané algoritmy zdieľať formou balíčkov. Tým pádom je možné venovať viac času na budovanie inteligentných robotických aplikácií.

6.1.1 Architektúra ROS

Implementácia algoritmov v ROS sa snaží o čo najväčšiu modularitu. Základným stavebným prvkom je takzvaný uzol. Uzol je program, ktorý bol navrhnutý na určitú úlohu, napríklad na komunikácia so senzorom. ROS ponúka veľa možností komunikácie medzi uzlami, no pre túto prácu sú dôležité komunikačné kanály a správy, posielané po daných kanáloch. Každý kanál má určený formát správy, ktorý môže byť komunikovaný v rámci daného kanálu. Uzly sa vedia na kanál pripojiť pomocou tried publisher a subscriber. Trieda publisher zabezpečuje posielanie informácií daným kanálom a trieda subscriber naopak odoberá informácie, posielané publsiherom z daného kanála. Pomocou komunikačných kanálov je teda možné jednoducho spojazdniť komunikáciu medzi uzlami.

6.1.2 Balíčky

V rámci ROS sú implementované algoritmy zorganizované do balíčkov. Obsahom balíčkov môže byť implementácia uzlov, knižníc a podobne. Výhodou tohto systému je jednoduchá distribúcia základných algoritmov. Pomocou balíčkov je jednoduché zdieľať prácu v rámci vedeckých projektov.

6.1.3 Simulátor Gazebo

V rámci ROS je možná simulácia robotov pomocou simulátora Gazebo. Gazebo je schopné simulovať interakciu objektov na základe fyzikálnych zákonov. V tomto simulátore je taktiež možné simulovať rôzne senzory s nastaviteľnými parametrami, ako je napríklad šum. Taktiež je možné vytvoriť rôzne vonkajšie alebo vnútorne prostredia, v ktorých je možné testovať algoritmy spojené s robotmi.

6.2 Konfigurácia robota HUSKY A200

V tejto práci bol použitý robot HUSKY A200 v kombinácií s LIDAR senzorom Velodyne VLP-16 a GNSS senzorom Trimble BX982. Robot a senzory boli simulované v simulátore Gazebo a uvedené do prevádzky v reálnom prostredí.

6.2.1 HUSKY A200

Husky A200 je modifikovateľné bezpilotné pozemné vozidlo (z anglického UGV). Toto vozidlo bolo dizajnované na použitie v náročnom vonkajšom teréne. V základnej konfigurácií, zobrazenej na obrázku 6.1, je vozidlo vybavené počítačom, batériou a štvorkolesovým pohonom. S vozidlom je taktiež možné komunikovať v rámci uzlov implementovaných v ROS. Pre túto prácu sú dôležité uzly `/husky/husky_kinematic` a `/basic_basic_odom`. Prvý uzol poskytuje rozhranie medzi ROS a počítačom vozidla Husky. Taktiež slúži na príjem správ na kanále `/cmd_vel`, podľa ktorých je ovládaný pohon vozidla. Obsahom týchto správ sú požadované lineárne a uhlové rýchlosti vozidla. V rámci vozidla je implementovaný regulátor, ktorý tieto správy interpretuje ako akčné zásahy na pohony kolies. Uzol `/basic_basic_odom` poskytuje informáciu o odometrii, vypočítanú z meraní rotačných enkodérov. V rámci mojej konfigurácie bola odometria počítaná pomocou diferenciálneho modelu vozidla [37]. Lineárnu rýchlosť v a uhlovú rýchlosť ω je možné získať pomocou nasledujúcich vzťahov

$$v = \frac{v_r + v_l}{2}, \tag{6.1}$$

$$\omega = \frac{v_r - v_l}{w}, \tag{6.2}$$

kde v_r a v_l sú rýchlosti kolies na pravej a ľavej strane vozidla a w je rozchod kolies [38]. Následne je aktuálna poloha v iterácií i vypočítaná pomocou nasledujúcich vzťahov

$$x_i = x_{i-1} + T_s \cdot v \cdot \cos(\omega_{i-1}), \tag{6.3}$$

$$y_i = y_{i-1} + T_s \cdot v \cdot \sin(\omega_{i-1}), \tag{6.4}$$

$$\omega_i = \omega_{i-1} + T_s \cdot \omega, \tag{6.5}$$

kde vektor $p = [x, y, \omega]$ popisuje polohu a orientáciu vozidla a znakom T_s je označená doba od poslednej aktualizácie polohy.

Obrázok 6.1: Robot HUSKY A200 v základnej konfigurácií [38].

Obrázok 6.2: Senzor VLP-16 [39].

6.2.2 VLP-16

VLP-16 je malý 3D LIDAR senzor schopný snímania veľkého množstva bodov v reálnom čase. Tento senzor používa 16 párov laserov a detektorov. Laserové lúče sú rozprestrené po prostredí pomocou rotujúcich zrkadiel. Vzdialenosť nasnímaných prekážok je vypočítaná na princípe časového rozdielu vyžiarenia a prijatia lúču, popísaného v podkapitole 3.5. Základné parametre tohto senzora sú popísané v tabuľke 6.1. Upevnenie tohto senzora na robota HUSKY A200 je zobrazené na obrázku 6.3 a samotný senzor na obrázku 6.2.

Vlastnosti senzora VLP-16	
dosah	100 m
presnosť	± 3 cm
horizontálne zorné pole	± 15°
vertikálne zorné pole	360°
horizontálne rozlíšenie	0.1° - 0.4°
vertikálne rozlíšenie	2.0°
otáčky	5 Hz - 20 Hz
počet bodov za sekundu	až 600 000
typická spotreba	8 W

Tabuľka 6.1: Vlastnosti senzora VLP-16 [39].

6.2.3 GNSS Tribmle BX982

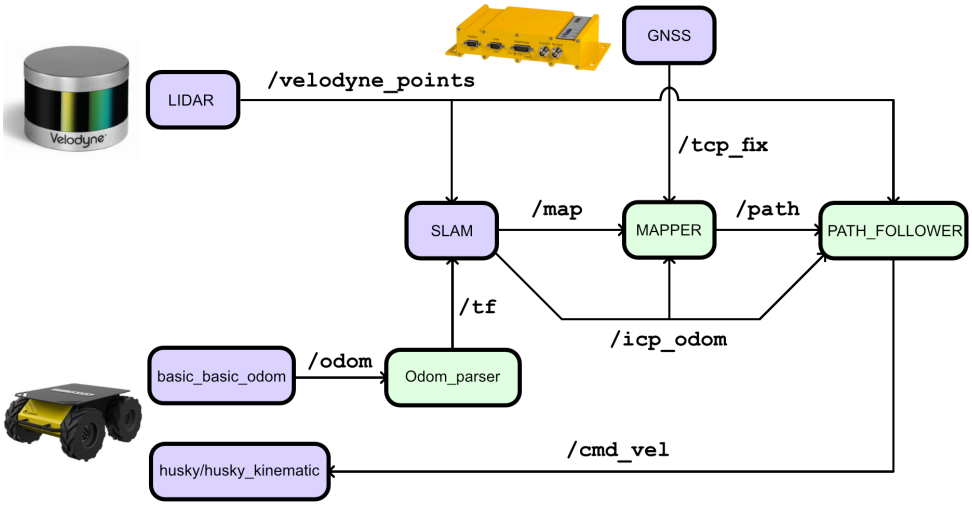
Tento senzor je schopný lokalizácie polohy s presnosťou na centimetre. Trimble BX982 je schopný prijímať signály zo systémov GPS, GLONASS, BeiDou a bude schopný prijímať signály zo systému GALILEO. Senzor obsahuje dva nezávislé moduly a procesor, ktorý spracováva informácie z modulov. So senzorom je možné spojiť sa pomocou ethernetu, čo umožňuje lokalizáciu polohy s frekvenciou v desiatkach Hz. Senzor taktiež podporuje dva antény, čo vylepšuje presnosť lokalizácie. Tento senzor je zobrazený na obrázku 6.4.

Obrázok 6.3: Konfigurácia HUSKY A200 - na robotovi je umiestnený senzor VLP-16 na platforme a GNSS Trimble BX982 v prednej časti vozidla.

Obrázok 6.4: Senzor Trimble BX982 [40].

6.3 Používané ROS uzly

V rámci tejto práce boli použité už implementované uzly z voľne dostupných balíčkov. Taktiež boli použité uzly, ktoré som implementoval a to konkrétne uzly `MAPPER`, `PATH_FOLLOWER` a `Odom_Parser`. Najprv sú popísane uzly, ktorých funkcia nie je komplikovaná. Následne som podrobne rozobral uzly `SLAM`, `MAPPER` a `PATH_FOLLOWER`, keďže tieto uzly sa zaoberajú mapovaním prostredia a navigáciou v danom prostredí. Vzťahy jednotlivých uzlov sú zobrazené na diagrame v obrázku 6.5.



Obrázok 6.5: Diagram uzlov v ROS - farebné boxy reprezentujú uzly a čiarami sú reprezentované komunikačné kanály. Zelenou farbou sú označené uzly ktoré som implementoval.

- `GNSS`: Tento uzol je implementovaný v balíčku `nmea_navsat_driver`[1] a slúži na preklad správ z GNSS senzora do správ v ROS formáte `sensor_msgs/NavSatFix`, ktoré obsahujú informáciu o zemepisnej šírke, zemepisnej dĺžke, nadmorskej výške a čase, v ktorom bola daná pozícia nameraná. Tieto správy sú posielané v rámci kanála `/tcpfix`.

- `LIDAR`: Uzol LIDAR slúži na interpretovanie nasnímaných bodových mrakov zo senzora VLP-16 a následne konvertovanie bodového mraku do správ v ROS formáte `sensor_msgs/PointCloud2`. Tieto správy obsahujú súradnice bodov v bodovom mraku a čas odoslania danej správy. Tieto správy sú posielané kanálom `/velodyne_points`. Implementácia tohto uzla sa nachádza v balíčku `velodyne`[2].

- `husky/husky_kinematic`: Ako už bolo spomenuté, tento uzol slúži ako rozhranie medzi robotom a ROS. Pre túto prácu je dôležitý príjem správ v ROS formáte `geometry_msgs/Twist` z kanálu `/cmd_vel`. Tieto správy sú následne použité na ovládanie pohybu robota.

- `basic_basic_odom`: Tento uzol poskytuje odometrické meranie po kanále `/odom` vo forme ROS správy `nav_msgs/Odometry`. Tento uzol je s uzlom `husky/husky_kinematic` implementovaný v rámci balíčkov[3] dostupných k robotovi HUSKY A200.

- `Odom_parser`: Tento uzol prekladá správy formátu `nav_msgs/Odometry` do správ posielaných kanálom `/tf`. Tento preklad je potrebný na správnu funkciu uzla SLAM, keďže ten prijíma odometrické meranie formou transformácie po kanále `/tf`. Transformácia popisuje posun počiatkov dvoch súradnicových systémov a je súčasťou správy `tf/tfMessage`.

6.3.1 Uzol SLAM

Funkciou tohoto uzla je prevádzka ICP SLAM, podrobne popísaného v kapitole 4. Vstupom sú body nasnímane senzorom VLP-16, ktoré si uzol preberá z kanála `/velodyne_points`. Druhým vstupom je odometrické meranie z kanála `/tf`. Tento uzol následne poskytuje mapu prostredia v rámci kanála `/map` vo forme ROS správy `sensor_msgs/PointCloud2` a lokalizovanú polohu robota v mape pomocou ROS správ `nav_msgs/Odometry` v kanále `/icp_odom`. Zásadným parametrom je `min_dist_new_point`, ktorý označuje minimálnu vzdialenosť bodov vo výslednej mape. Tento parameter má veľký vplyv na rozlíšenie mapy a rýchlosť ICP SLAM algoritmu. Zvyšné dôležité parametre tohto uzla sú popísané v tabuľke 6.2. Počas experimentov som upravil spôsob filtrovania vstupných bodov v tomto uzli. Na výpočet pravdepodobnosti toho, že body v bodovom mraku sú dynamické, bol pôvodne použitý filtrovaný bodový mrak merania. V prípade odstránenia veľkého množstva bodov z tohto bodového mraku, bol následný výpočet spomínanej pravdepodobnosti

[1] http://wiki.ros.org/nmea_navsat_driver
[2] http://wiki.ros.org/velodyne
[3] http://wiki.ros.org/Robots/Husky

vykonaný na menšom počte bodov v bodovom mraku mapy. Uzol SLAM som upravil tak, aby do výpočtu spomínanej pravdepodobnosti vstupoval nefiltrovaný bodový mrak merania.

názov parametra	popis parametra	rozsah hodnôt
min_dist_new_point	minimálna vzdialenosť bodov v mape	$\langle 0, \infty)$
sensor_max_range	maximálna vzdialenosť nasnímaného bodu od senzora	$\langle 0, \infty)$
prior_dynamic	počiatočná pravdepodobnosť toho, že bod je dynamický	$\langle 0,1\rangle$
threshold_dynamic	γ_{max}	$\langle 0,1\rangle$
beam_half_angle	$\theta_{\max}$	$\langle 0, \frac{\pi}{2})$
epsilon_a	e_a	$\langle 0, \infty)$
epsilon_d	e_d	$\langle 0, \infty)$
alpha	α	$\langle 0,1\rangle$
beta	β	$\langle 0,1\rangle$

Tabuľka 6.2: Dôležité parametre uzla SLAM. Popis parametrov $\theta_{\max}$, e_d, e_a, α, β a γ_{max} sa nachádza na konci podkapitoly 4.5.

6.3.2 Uzol MAPPER

Funkciou tohto uzla je vyhodnotenie priechodnosti terénu v mape, ktorú poskytuje uzol `SLAM` a následne nájdenie cesty v danej mape. Vstupom je mapa vo forme bodového mraku z kanála `/map`, lokalizovaná poloha robota v danej mape z kanála `/icp_odom` a cieľová poloha, do ktorej sa má robot dostať z kanála `/ciel`. Cieľová poloha môže byť posielaná formou GNSS súradnice alebo ako karteziánska súradnica. Uzol je členený na triedy Mapper a Pathfinder.

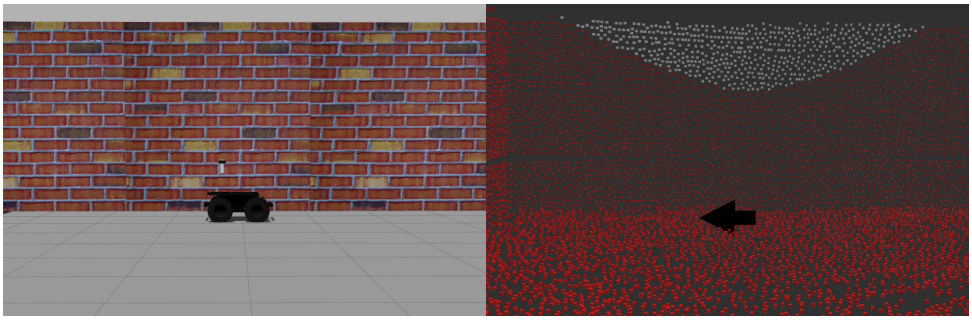
Trieda Mapper

Trieda Mapper sa zaoberá spracovaním mapy. V prvom kroku je bodový mrak reprezentujúci mapu prefiltrovaný. Každý bod $p = [x, y, z]$ je z bodovej mapy odstránený v prípade, že súradnica z_p presahuje limit h_{max1} stanovený rovnicou

$$h_{max1} = h_r + \varphi_{max} \cdot \sqrt{(x_p - x_r)^2 + (y_p - y_r)^2}, \tag{6.6}$$

kde h_r popisuje výšku robota, parameter φ_{max} označuje maximálny uhol pod ktorým môže robot stúpať, a $r = [x, y, z]$ je poloha robota v mape. Cieľom je filtrácia bodov z mapy, ktoré sú voči polohe robota príliš vysoko a neovplyvňujú priechodnosť povrchu, po ktorom môže robot jazdiť. S narastajúcou vzdialenosťou bodu voči polohe robota je limit h_{max} navyšovaný, pretože robot dokáže v prostredí stúpať pod uhlom φ_{max}. Príkladom môžu

byť nasnímané body zo stromov alebo stropov v budovách. Funkcia tohto filtra je zobrazená na obrázku 6.6.



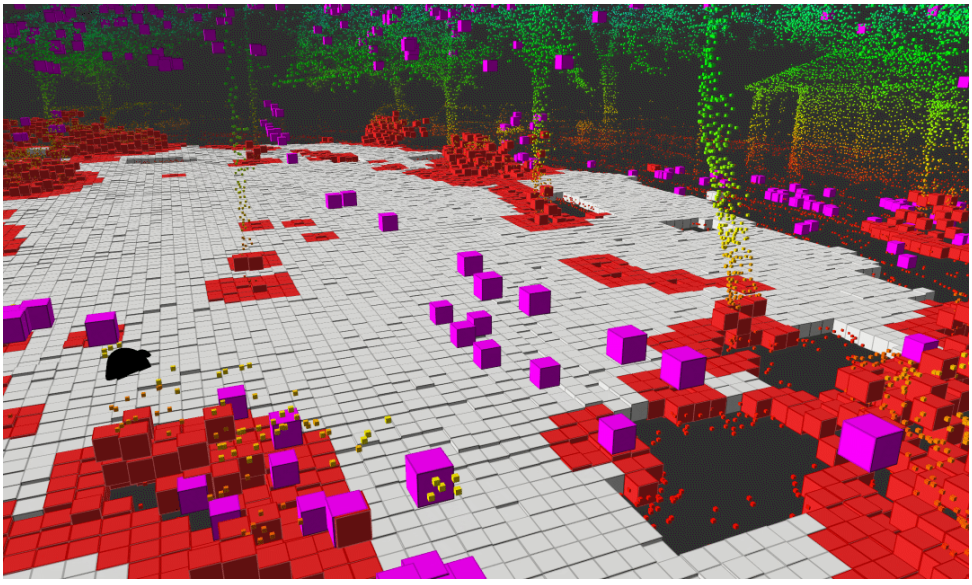
Obrázok 6.6: Funkcia filtra bodového mraku - limit h_{max1} je vypočítaný na základe hodnoty parametrov $h_r = 1.0\ m$ a $\varphi_{max} = 23°$. Bielou farbou sú označené odfiltrované body a červenou farbou sú označené body po filtrovaní.

Filtrácia bodov na základe výpočtu pravdepodobnosti toho, že bod je dynamický, zanechávala v mape body, ktoré boli nasnímane na dynamických objektoch. Väčšina týchto bodov tvorila malé zhluky osamotené v priestore. Pre efektívnejšie odstraňovanie takýchto bodov z mapy, som implementoval nasledujúci filter. Filter z mapy vymaže body, pre ktoré priemerná vzdialenosť k najbližším susedným bodom prekročí stanovený limit. Vstupom sú tri dvojice parametrov $par_k = (knn_k, dist_k)$, kde k popisuje index danej dvojice. Parameter knn popisuje počet susedov, od ktorých je počítaná priemerná vzdialenosť. Parametrom $dist$ je nastavený limit maximálnej možnej priemernej vzdialenosti. Bod je z mapy vymazaný akonáhle kontrola vzdialenosti zlyhá na ľubovoľnej dvojici par_k. Funkcia tohto filtra je zobrazená na obrázku 6.7.

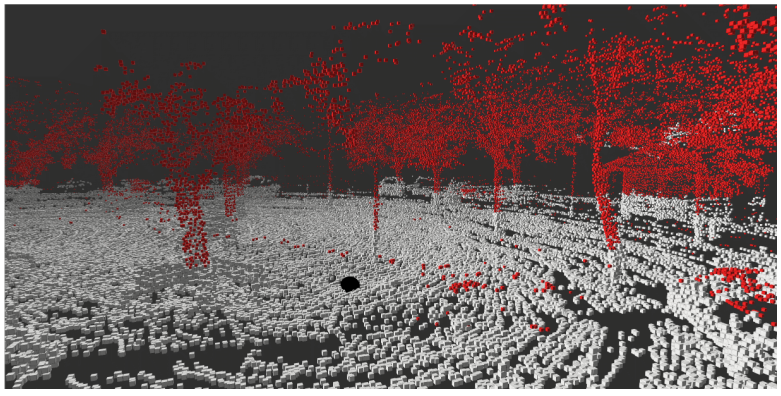
Po filtrácií bodového mraku sú dané body zasadené do 2D mriežky. V tejto mriežke je každá bunka tvorená objektom triedy *Grid_cell*. Tento objekt slúži na ukladanie bodov, pomocných parametrov a stavu priechodnosti danej bunky. Mapa priechodnosti je reprezentovaná práve touto mriežkou. Každá bunka má svoju celočíselnú súradnicu v osiach x,y. Bod $p = [x, y, z]$ je do bunky $cell_1 = [x, y]$ vložený v prípade, že súradnica z_p nepresahuje limit h_{max2}. Každá bunka má stanovený limit h_{max2} s ohľadom na najnižšie body $q = [x, y, z]$ v ostatných bunkách. V prípade, že súradnica z_p je menšia ako z_q dochádza k aktualizácii limitu h_{max2} podľa rovnice

$$h_{max2} = h_r + \varphi_{max} \cdot \sqrt{(x_{cell_1} - x_{cell_2})^2 + (y_{cell_1} - y_{cell_2})^2}, \qquad (6.7)$$

kde $cell_2 = [x, y]$ reprezentuje susednú bunku voči bunke $cell_1$. Tento limit je aktualizovaný v prípade, že je menší ako predošlý limit v bunkách, ktoré niesu vzdialené od bunky $cell_1$ o viac ako dĺžku troch buniek. Cieľom tohto postupu je pri vyhodnocovaní priechodnosti vziať do úvahy len body, ktorých výška nad terénom je maximálne h_r. Porovnanie vložených bodov do mriežky oproti pôvodnému bodovému mraku je zobrazený na obrázku 6.8.



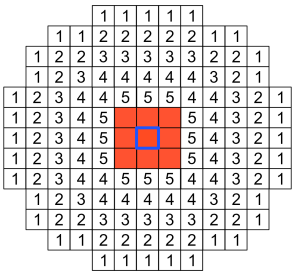
Obrázok 6.7: Filtrovanie dynamických bodov a mapa priechodnosti - na obrázku je možné vidieť mapu priechodnosti, reprezentovanú bielymi a červenými bunkami. Biele bunky popisujú priechodný priestor a červené bunky naopak popisujú priestor nepriechodný. Fialovými kockami sú označené body odfiltrované pomocou filtra, založeného na najbližších susedoch. V strede obrázku je vidieť odfiltrované body, ktoré boli nasnímané na osobe prechádzajúcej parkom. Zvyšné body reprezentujú bodový mrak mapy.



Obrázok 6.8: Body zasadené do buniek mriežky - bielou farbou sú označené body zasadené do 2D mriežky a červenou farbou sú označené odfiltrované body. V pravo hore je na obrázku vidieť zhluk bodov, ktoré boli zasadené do 2D mriežky napriek tomu, že sa nachádzajú vysoko nad povrchom chodníka. Tieto body sa z 2D mriežky vyfiltrujú v momente, kedy bude nasnímaný povrch pod nimi, teda povrch chodníka.

Následne je mapa posunutá na základe polohy robota. Poloha robota v mriežke je vždy centrovaná. Mapa priechodnosti sa teda s pohybom robota taktiež pohybuje. Posledným krokom je vyhodnotenie priechodnosti prostredia, ktoré už bolo popísané v podkapitole 5.4. Kvôli optimalizácií toto hodnotenie neprebieha na celej mape, ale len v určitom okruhu polohy robota. Parametrom

je možné nastaviť polomer tohto okruhu. Všetky bunky, o ktorých sa zatiaľ nerozhodlo alebo sú nepriechodné, sú nafukované. Nafúknutím je myslená zmena stavu priechodnosti susedných buniek. Vzor, podľa ktorého sú tieto bunky nafukované, je vysvetlený a zobrazený na obrázku 6.9. K prejazdným bunkám, ktoré sa nachádzajú blízko neprejazdeným, je taktiež určená vyššia cena pri hľadaní cesty. Ukážka mapy priechodnosti je zobrazená na obrázku 6.7. Dôležité parametre triedy Mapper sú zobrazené v tabuľke 6.3.



Obrázok 6.9: Nafukovanie buniek - nafukovaná bunka je vyznačená modrou hranou. Červenou farbou sú zobrazené bunky, ktoré zmenia svoj stav priechodnosti na stav nafukovanej bunky. Čísla v ostatných bunkách reprezentujú veľkosť penalizácie pri hľadaní cesty cez dané bunky. Počet červených buniek je daný parametrom *obstacle_distance*, ktorý označuje vzdialenosť na ktorú sa daná bunka nafukuje.

Trieda Pathfinder sa zaoberá hľadaním cesty voči cieľu v mape priechodnosti. Cieľ a poloha robota sú interpretované ako body v priestore. Súradnicu cieľa nie je nutné voči mape prepočítavať v prípade, že sa nejedná o GNSS súradnicu. V prípade GNSS súradnice je nutný prepočet do súradnicového systému, v ktorom je vyjadrená poloha robota. V práci som použil prepočet z článku [41], ktorý má na vstupe cieľovú a referenčnú polohu v GNSS súradniciach. Výsledkom tohto prepočtu je vzdialenosť jednotlivých súradníc. Prepočet spočíva v prevode polôh do geocentrickej súradnicovej sústavy. Následne je zavedený súradnicový systém ENU, ktorého počiatkom je poloha referencie, teda poloha robota. V poslednom kroku je do systému ENU prevedená cieľová poloha a vypočítaná vzdialenosť cieľovej polohy voči počiatku. Pre popis výpočtu si definujeme nasledujúce premenné a konštanty:

- ϕ_1: zemepisná šírka,
- λ_1: zemepisná dĺžka,
- h_1: nadmorská výška,
- $R = [\phi_1, \lambda_1, h_1]$: súčasná (referenčná) poloha robota vyjadrená v GNSS súradniciach,
- $C = [\phi_1, \lambda_1, h_1]$: cieľová poloha vyjadrená v GNSS súradniciach.

Nasledujúce rovnice 6.8,6.9,6.10 popisujú rozdiely súradníc cieľovej a referenčnej polohy:

$$d\phi_1 = C_{\phi_1} - R_{\phi_1} \tag{6.8}$$

$$d\alpha_1 = C_{\alpha_1} - R_{\alpha_1} \tag{6.9}$$

$$dh_1 = C_{h_1} - R_{h_1} \tag{6.10}$$

Vzdialenosť $d = [x, y, z]$ cieľovej a referenčnej polohy vypočítame pomocou nasledujúcich rovníc:

$$\chi = \sqrt{1 - e^2 \cdot \sin(\phi_1)^2} \tag{6.11}$$

$$e^2 = 1 - \left(\frac{b}{a}\right)^2 \tag{6.12}$$

$$\begin{aligned} d_x = \left(\frac{a \cdot (1 - e^2)}{\chi^3} + h_1\right) \cdot d\phi_1 + \frac{3}{2} \cdot a \cdot \cos(R_{\phi_1}) \cdot \sin(R_{\phi_1}) \cdot e^2 \cdot d\phi_1^2 \\ + dh_1 \cdot d\phi_1 + \frac{1}{2} \cdot \sin(R_{\phi_1}) \cdot \cos(R_{\phi_1}) \cdot (\frac{a}{\chi} + h) \cdot d\lambda_1^2 \end{aligned} \tag{6.13}$$

$$\begin{aligned} d_y = \left(\frac{a}{\chi} + h\right) \cdot \cos(R_{\phi_1}) \cdot d\lambda - \left(\frac{a \cdot (1 - e^2)}{\chi^3} + h_1\right) \cdot \sin(R_{\phi_1}) \cdot d\phi_1 \cdot d\lambda_1 \\ + \cos(R_{\phi_1}) \cdot d\lambda_1 \cdot dh_1 \end{aligned} \tag{6.14}$$

$$\begin{aligned} d_z = dh_1 - \frac{1}{2} \cdot a \cdot \left(1 - \frac{3}{2} \cdot e^2 \cdot \cos(R_{\phi_1}) + \frac{1}{2} \cdot e^2 + \frac{h_1}{a}\right) \cdot d\phi_1^2 \\ - \frac{1}{2} \cdot \left(\frac{a \cdot cos^2(R_{\phi_1})}{\chi} - h_1 \cdot cos^2(R_{\phi_1})\right) \cdot d\lambda_1^2 \end{aligned} \tag{6.15}$$

Polohu cieľu v mape $ciel = [x, y]$ následne vypočítame pomocou rovníc

$$ciel_x = d_x + robot_x, \tag{6.16}$$

$$ciel_y = d_y + robot_y, \tag{6.17}$$

kde $robot = [x, y]$ je poloha robota v mape v karteziánskych súradniciach. Keďže mapa priechodnosti má len dve rozmery, súradnice v ose z niesu počítané. Poloha cieľu v mape sa aktualizuje pri každom prijatí správy o súčasnej GNSS polohe robota. Aktualizácia neprebieha v prípade, že norma vzdialenosti d je menšia ako 5 metrov, keďže lokalizácia polohy pomocou GNSS senzora nie je úplne presná.

V nasledujúcom kroku je potrebné priradiť cieľ a polohu robota k bunkám v mape priechodnosti. Každá bunka vymedzuje v mape priechodnosti priestor

o tvare štvorca. Cieľ bude priradený do bunky, ktorá pokrýva plochu v ktorej sa nachádzajú súradnice cieľa. Bunka v ktorej sa nachádza robot je v strede mapy, keďže mapa je centrovaná na polohu robota. Bunky v ktorých je poloha cieľu a poloha robota nesmú byť nepriechodné. V prípade, že je poloha robota v nepriechodnej bunke, prehľadávanie cesty končí a cestu nie je možné nájsť. Ak sa cieľ nachádza v nepriechodnej bunke, spustí sa prehľadávanie v okruhu 1 metra. Ak sa v tomto okruhu nenájde prejazdná bunka, hľadanie cesty končí a nie je možné ju nájsť, keďže cieľom je nepriechodné prostredie. Mapa má obmedzené rozmery a môže nastať prípad, kedy nie je možné cieľu priradiť bunku, kvôli jeho vzdialenosti od súčasnej polohy robota. V tomto prípade je vybraná bunka, do ktorej spadajú súradnice priesečníka hrany mapy a úsečky, ktorá spája cieľ so súčasnou polohou robota. Ak je tato bunka nepriechodná, cieľu bude priradená najbližšia priechodná bunka na hrane mapy. V prípade, že takáto bunka neexistuje, hľadanie cesty končí.

Posledným krokom je nájdenie cesty. V tomto momente je dostupná mapa priechodnosti s bunkami reprezentujúcimi cieľ a štart v prípade, že hľadanie cesty nebolo doteraz ukončené. Jedná sa teda o hľadanie cesty v 2D mriežke, čo je jednoduchý problém, ktorý budem riešiť pomocou A* algoritmu. Tento prelomový algoritmus bol predstavený v článku [42]. Jedná sa o algoritmus, ktorý hľadá najlacnejšiu (najkratšiu) cestu v grafe, medzi počiatočným a cieľovým uzlom. V prípade tejto práce je graf tvorený 2D mriežkou, kde uzly grafu reprezentujú jednotlivé bunky a hrany grafu sú reprezentované spojením buniek. Každá bunka, ktorá nie je na hrane mapy, má osem susedných uzlov, spojených ôsmimi hranami. Algoritmus začína v počiatočnom uzle, z ktorého expanduje do susedných uzlov, s ktorými má spoločnú hranu. Pre každý susedný uzol je vypočítaná nasledujúca cenová funkcia

$$f(n) = g(n) + h(n) \tag{6.18}$$

kde $g(n)$ je vzdialenosť od počiatočného uzlu k uzlu n a $h(n)$ je odhad vzdialenosti medzi uzlom n a cieľovým uzlom. Dôležitú rolu zohráva heuristicka funkcia $h(n)$. Algoritmus nájde najkratšiu cestu v prípade, že je táto funkcia prípustná. Funkcia je prípustná, ak je odhad ceny cesty od uzla n do cieľového uzla neprekračuje reálnu cenu tejto cesty. V tejto práci bol použitý tvar heuristickej funkcie daný rovnicou

$$\begin{aligned} distance(n_1, n_2) &= |n_{1x} - n_{2x}| + |n_{1y} - n_{2y}| \\ &+(\sqrt{2} - 2) \cdot \min\{|n_{1x} - n_{2x}|, |n_{1y} - n_{2y}|\}, \end{aligned} \tag{6.19}$$

kde symboly n_1 a n_2 reprezentujú uzly, teda bunky v mape priechodnosti. Do ceny cesty sú tiež zakomponované penalizácie z buniek, ktoré sa nachádzajú v okolí nepriechodných buniek. Ak je pre susedný uzol hodnota tejto cenovej funkcie nižšia ako doterajšia cena, uzol z ktorého bolo prevedené expandovanie je nastavený ako rodič. Následne je vybraný uzol s najlacnejšou (najkratšou) cestou a tento proces sa opakuje. Expandovanie každého uzla môže prebehnúť len raz. Algoritmus končí v prípade, že sa expandovaním narazí na cieľový uzol. Cesta od počiatočného do cieľového je tvorená postupnosťou rodičov

uzlov počínajúc v cieľovom uzle. Pseudokód tohto algoritmu je zobrazený v Algoritme 4.

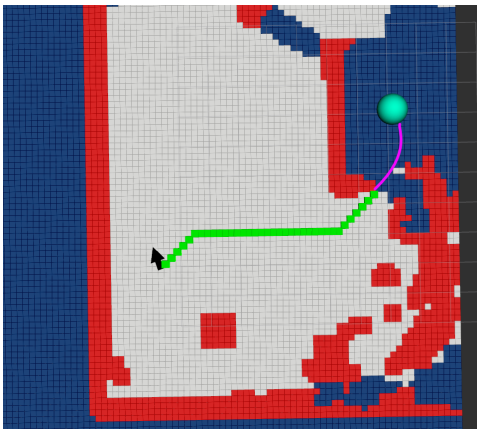
Algorithm 4 A*

```
Require: start_node                                   ▷ Počiatočný uzol
Require: end_node                                     ▷ Cieľový uzol
  Nodes_to_test.insert(start_node)          ▷ Uzly, ktoré budú expandované
  Tested_nodes                          ▷ Zoznam obsahujúci už navštívené uzly
  while end_node not in Tested_nodes do
      Current_node = lowest_f(n)(Nodes_to_test) ▷ Uzol s najmenším f(n)
      Tested_nodes.insert(Current_node)
      for each neighbour of Current_node do
          if neighbour in Tested_nodes then
              break
          end if
          if neighbour not in Nodes_to_test then
              Nodes_to_test.insert(neighbour)
          end if
          new_price = g(Current_node) + distance(Current_node,neigbour)
  + neighbour.penalty
          if new_price < g(neighbour) then
              g(neighbour) = new_price
              f(neighbour) = g(neighbour) + distance(neighbour,end_node)
              neighbour.parent = Current_node
          end if
      end for
  end while
```

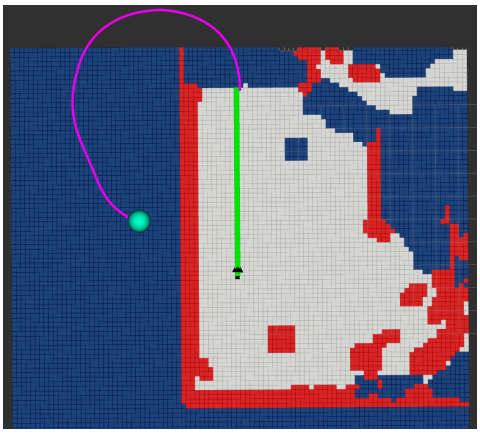
Mapa priechodnosti je najprv prehľadaná do šírky z počiatočnej bunky. Prehľadávanie postupuje po priechodných bunkách a identifikuje hraničné bunky a bunky na hrane mapy. Hraničné bunky sú priechodné bunky, ktoré susedia s bunkou, ktorej nebol doposiaľ pridelený stav priechodnosti. Následne je mapa prehľadaná z cieľovej bunky. V tomto prípade prehľadávanie postupuje po všetkých bunkách, ktorých stav nie je nepriechodný. Cieľom tohto postupu je identifikovanie toho, či existuje v mape cesta od počiatočnej k cieľovej bunke. Cesta neexistuje, ak sa pri prehľadávaní z počiatočnej bunky nenarazilo na bunku cieľovú a neboli identifikované žiadne hraničné bunky alebo bunky na hrane mapy. V takomto prípade je robot uzavretý v priestore, ohraničenom nepriechodnými bunkami viz obrázok 6.10c. Ak sa pri prehľadávaní z počiatočnej bunky našli hraničné body a pri prehľadávaní z bunky cieľovej sa na nejaký z týchto bodov narazilo, cesta od počiatočnej bunke k cieľovej existuje viz obrázok 6.10a. V závislosti na veľkosti mapy môže nastať prípad, kedy existuje možnosť potenciálnej cesty mimo terajšiu mapu priechodnosti. V prípade, že sa pri prehľadávaní z počiatočnej a cieľovej bunky narazí len na bunky na hrane mapy, v mape priechodnosti neexistuje priama cesta medzi danými bunkami. Obe bunky sa nachádzajú v priestoroch

oddelených nepriechodnými bunkami. Existuje však možnosť toho, že cestu je možné nájsť prechodom cez hranu mapy viz obrázok 6.10b. V tomto prípade je hľadanie cesty zastavené v momente, kedy sa pri expandovaní narazí na hranu mapy. Vo všetkých ostatných prípadoch cesta z počiatočnej bunky do cieľovej bunky neexistuje.

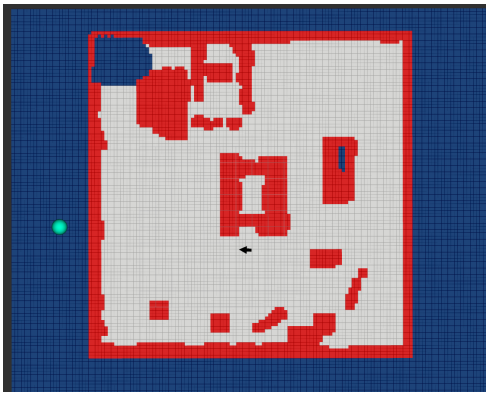
Algoritmus A* nájde vhodnú cestu v prípade, že sa identifikuje existencia danej cesty. Výsledná cesta je vyjadrená formou bodového mraku a je publikovaná na kanál `/path` vo forme ROS správy `sensor_msgs/PointCloud2`. V prípade, že cestu nie je možné nájsť, je robot zastavený a uzol Mapper čaká na zadanie nového cieľa. Dôležité parametre triedy Pathfinder sú zobrazené v tabuľke 6.3.



(a) : Cesta existuje v mape.



(b) : Cesta potenciálne existuje mimo mapy.



(c) : Cesta neexistuje, keďže je robot obklopený nepriechodnými bunkami.

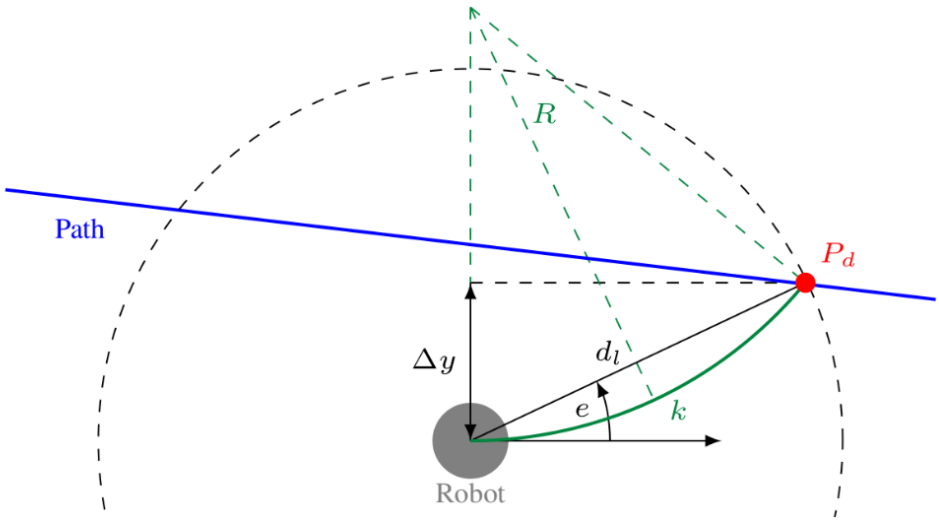
Obrázok 6.11: Popis možných situácií pri hľadaní cesty - na obrázkoch sú bielou farbou vyznačené priechodné bunky, červenou farbou nepriechodné bunky a modrou farbou bunky o ktorých nebolo doposiaľ rozhodnuté. Zelenou farbou sú vyznačené bunky, ktoré tvoria časť cesty po priechodných bunkách a ružovou farbou je vyznačený potenciálny koniec cesty. Čiernou šípkou je označená poloha robota a modrozelenou guľou je označený cieľ.

názov parametra	popis parametra	rozsah hodnôt
num_of_grid_cells	počet buniek v jednom rozmere mapy	$\mathbb{Z}^+$
grid_size	dlžka hrany bunky	$\langle 0, \infty)$
max_height_treshold	výška robota	$\langle 0{,}1 \rangle$
max_angle_treshold	maximálny uhol stúpania	$\langle 0°, 90°)$
max_variance	maximálny možný rozptyl bodov v bunke	$\langle 0, \infty)$
obstacle_distance	vzdialenosť nafukovania prekážok	$\langle 0, \infty)$
procesing_distance	vzdialenosť od polohy robota v ktorej sa vyhodnocuje priechodnosť prostredia	$\langle 0, \infty)$
knn_x	počet susedov pre filter založený na najbližších susedoch	$\mathbb{Z}^+$
dist_x	limit priemernej vzdialenosti susedov od bodu pre filter založený na najbližších susedoch	$\langle 0, \infty)$
points_around_treshold	minimálny počet bodov v okolí bunky pre stanovenie priechodnosti	$\mathbb{Z}^+$

Tabuľka 6.3: Dôležité parametre uzla Mapper.

6.3.3 Uzol PATH_FOLLOWER

Funkciou tohto uzla je sledovanie nájdenej cesty. Nájdená cesta je prijatá formou správy z kanála `/path` a je na ňu aplikovaný filter kĺzavého priemeru. Súčasná poloha robota je prijatá formou správy z kanála `/icp_odom`. Poloha robota je z uzla SLAM aktualizovaná v jednotkách Hz. Za účelom generovania presnejších riadiacich príkazov je táto poloha interpolovaná rovnako ako výpočet odometrie, popísaný v podkapitole 6.2.1. Na sledovanie čiary bol použitý jednoduchý P-regulátor v angličtine nazývaný Pure Pursuit regulátor, predstavený v článku [43]. Sledovanie čiary spočíva v nastavení konštantnej rýchlosti robota a regulácie rotačnej rýchlosti robota na základe odchýlky Δy robota od sledovanej trasy. Táto odchýlka je vypočítaná voči bodu P_d, ktorý je priesečníkom pomyselnej kružnice a sledovanej trasy. Princípom je regulácia rotačnej rýchlosti robota na základe trasy v určitej vzdialenosti pred robotom, ktorá tvorí danú pomyselnú kružnicu. Výpočet je zobrazený na obrázku 6.12.



Obrázok 6.12: Vizualizácia výpočtu odchýlky Δy a polomeru R [44].

Krivka k spája robota z priesečníkom P_d. Polomer tejto krivky je možné vyjadriť ako

$$R = \frac{d_l^2}{2 \cdot \Delta y}. \tag{6.20}$$

Uhlovú rýchlosť je následne možne vypočítať ako

$$\omega = K_p \cdot \frac{v}{R}, \tag{6.21}$$

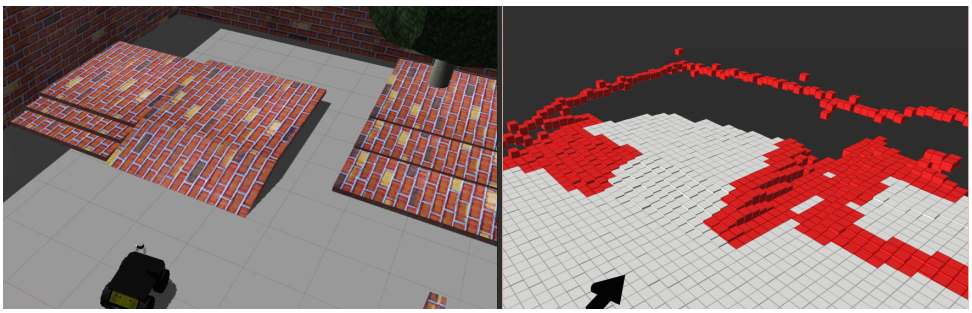
kde K_p je pridaná ladiaca konštanta. Keďže akčný zásah je závislý na hodnote vzdialenosti d_l, konštantou K_p je možné daný regulátor doladiť. Tieto rýchlosti sú následne poslané do kanála `/cmd_vel` formou ROS správy `geometry_msgs/Twist`.

Uzol taktiež prijíma surové dáta z kanála `/velodyne_points`, ktoré obsahujú LIDAR scan zo senzora VLP-16. V prípade, že sa body z LIDAR scanu nachádzajú príliš blízko robota, robot je zastavený poprípade je povolený len rotačný pohyb. Myšlienkou je kontrola dynamických objektov ktoré sa dostali pred robota. Keďže spracovanie mapy a nájdenie cesty trvá nejaký čas, robot prijíma nové informácie o prostredí z dopravným oneskorením. Výsledkom tejto kontroly je okamžite obmedzenie pohybu robota. Parameter d_l je možné parametrizovať spolu s rýchlosťou v. Taktiež je možné parametrami v_{max} a ω_{max} nastaviť maximálnu lineárnu a rotačnú rýchlosť.

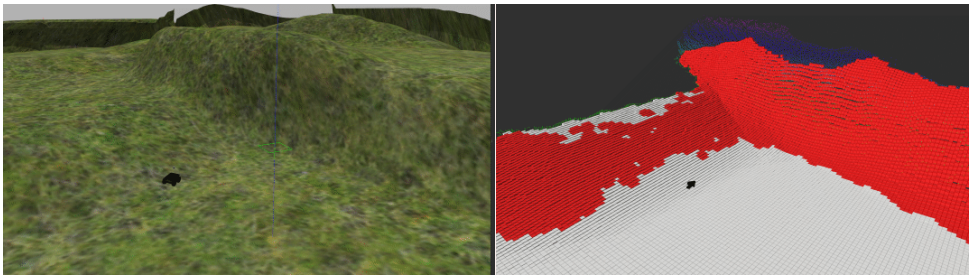
Kapitola 7

Experimentálne výsledky

V rámci experimentov boli vykonané pokusy v simulátore Gazebo, pokusy na reálnom robotovi v laboratóriu a dva samostatné jazdy v mestskom prostredí. Na vyhodnotenie výsledkov nepoužívam žiadnu metriku. Klasifikáciu buniek budem porovnávať s vlastným úsudkom. Keďže sa s robotom nejazdilo vo veľmi náročnom teréne, identifikovať nepriechodné prekážky nie je pre človeka náročné. Parametrizovanie uzla SLAM bolo v priebehu všetkých experimentov rovnaké. Dôležité použité parametre pre jednotlivé moduly sú zobrazené v tabuľke 7.1. Cieľom prvých experimentov v simulátore Gazebo, bolo nájsť vhodné konštanty pre limit rozptylu bodov v bunke mapy priechodnosti a otestovať funkčnosť implementácie. Od mapy priechodnosti som očakával vhodnú klasifikáciu nepriechodných buniek so zameraním na objekty pripomínajúce obrubník alebo schod. Na tento účel boli vytvorené dva jednoduché prostredia. Na obrázkoch 7.1,7.2 je možné vidieť dva z počiatočných testov, na ktorých bol odladený parameter rozptylu. Pokiaľ to nebude inak v popise obrázku dané, červené bunky reprezentujú nepriechodný povrch, biele bunky reprezentujú povrch priechodný a malé sfarbené body reprezentujú bodový mrak mapy.



Obrázok 7.1: Test klasifikácie obrubníkov - cieľom tohto testu bolo správne klasifikovať nepriechodné bunky ktoré ležia na schodoch. Na ľavej časti obrázku sa nachádzajú schody o výške 10 cm a vedľa nich je šikmá plocha. Tieto schody sú pre robota nepriechodné, keďže majú veľký výškový rozdiel. Šikmá plocha je správne klasifikovaná ako priechodná. Na pravej časti obrázku je možné vidieť podobný prípad. Oproti prvému schodu je rozdielna výška druhého a tretieho schodu. Tieto schody majú 5 cm a boli správne vyhodnotené ako prejazdné.



Obrázok 7.2: Test klasifikácie šikmých plôch - na tomto teste bola otestovaná funkcia klasifikácie na šikmých plochách. Algoritmus tieto plochy vyhodnotil správne ako nepriechodné, keďže sú pre robota príliš strmé.

Na základe týchto a viacerých testov boli nastavené parametre uzla MAPPER, ktoré sú zobrazené v tabuľke 7.3. Nastavenie konfigurácie ICP SLAM je z veľkej časti podobné konfigurácií, ktorú používal tým CTU-CRAS-NORLAB v súťaží DARPA Subterranean Challenge [45]. Konfiguráciou som sa rozhodol inšpirovať, pretože poskytovala kvalitnejšiu mapu a lepší výkon ako konfigurácie, ktoré som vypracoval sám. Popis tejto konfigurácie sa nachádza v tabuľke tabuľke 7.1 a nastavenie parametrov uzla SLAM sa nachádza tabuľke 7.2. V rámci uzla PATH_FOLLOWER boli nastavená maximálna lineárna rýchlosť $v_{max} = 0.5\ m \cdot s^{-1}$, maximálna rotačná rýchlosť $\omega_{max} = 0.3\ rad \cdot s^{-1}$ a vzdialenosť $d_l = 1m$.

	Funkcia	Parametre
Filtre merania	FixStepSamplingDataPointsFilter	$n_f = 3$
	BoundingBoxDataPointsFilter	$b = [-0.5, 0.5, -0.3, 0.3, -0.3, 0.3]$
	RandomSamplingDataPointsFilter	$p_r = 0.7$
	SurfaceNormalDataPointsFilter	$knn = 12$, $keep\_densities = true$
	MaxDensityDataPointsFilter	$\rho = 10$
	OrientNormalsDataPointsFilter	$towardCenter = true$
Konfigurácia ICP	KDTreeMatcher	$n_m = 7$, $d_{max} = 1.1\ m$
	RobustOutlierFilter	$k = 3$,
	SurfaceNormalOutlierFilter	$\phi_{sn} = 0.42\ rad \cdot s^{-1}$
	PointToPlaneErrorMinimizer	
	DifferentialTransformationChecker	$\varepsilon_{\theta_{min}} = 0.001$, $\varepsilon_{t_{min}} = 0.01$
	CounterTransformationChecker	$i_{max} = 40$
	BoundTransformationChecker	$\varepsilon_{\theta_{max}} = 0.9$, $\varepsilon_{t_{max}} = 10$
Filtre mapy	DistanceLimitDataPointsFilter	$dist = 30\ m$
	CutAtDescriptorThresholdDataPoints-	$descriptor = probDynamic$
	Filter	$d_{t_{max}} = 0.8$
	SurfaceNormalDataPointsFilter	$knn = 12$

Tabuľka 7.1: Konfigurácia ICP SLAM. Všetky použité moduly, filtre a ich parametre sú popísané v podkapitole 4.4.

názov parametra	hodnota
min_dist_new_point	0.07 m
sensor_max_range	30 m
threshold_dynamic	0.75
beam_half_angle	0.003
epsilon_a	0.01
epsilon_d	0.01
alpha	0.8
beta	0.9

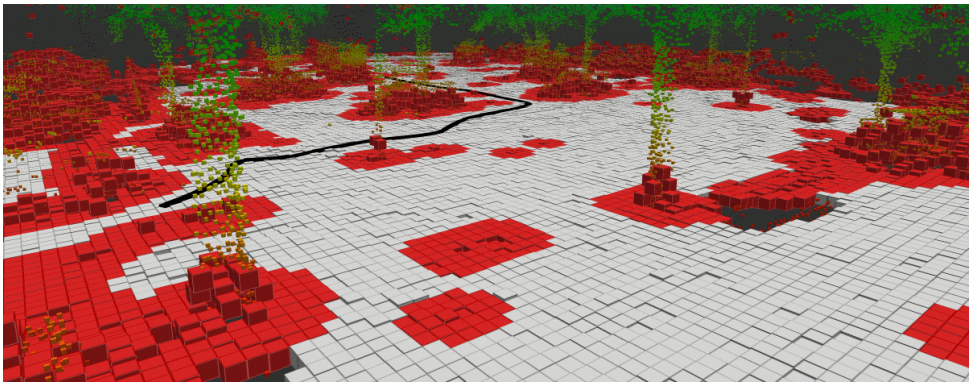
Tabuľka 7.2: Parametre uzla SLAM.

názov parametra	hodnota
num_of_grid_cells	300
grid_size	0.2 m
max_height_treshold	1 m
max_angle_treshold	23°
max_variance	0.0017
obstacle_distance	0.4

Tabuľka 7.3: Parametre uzla MAPPER.

Prvá skúšobná jazda prebehla v malom mestskom parku. Spočiatku bol robot ovládaný manuálne a testovala sa analýza priechodnosti v reálnych podmienkach. Na obrázku 7.3 je možné vidieť správne vyhodnotenú prekážku vo forme obrubníka. Následne som robotovi zadával cieľovú polohu v karteziánskych súradniciach so vzdialenosťou na pár metrov. Robot v tomto teste nevykonal čiste autonómnu jazdu na vzdialenosť viac ako 5 metrov. Na obrázkoch 7.4a, 7.4b je možné vidieť výsledok analýzy priechodnosti v časti parku a taktiež trajektóriu robota, ktorá je výsledkom viacerých zadaných cieľových polôh. V tomto teste boli hlavné problémy spôsobené dynamickými objektami, ktoré zanášali do mapy veľa šumu. Tento šum bol následne vyhodnotený ako nepriechodná oblasť. Ďalší problém spočíval v príliš agresívnej klasifikácií na okrajoch mapy. Analýza priechodnosti často vyhodnotila bunky na krajoch bodového mraku ako nepriechodné. Väčšina týchto buniek bola vyhodnotená ako priechodná v momente, kedy narástol počet bodov v okolí daných buniek. Ďalším problémom bola správna detekcia menších prekážok. Obecne je moja metóda veľmi závislá na kvalite bodového mraku z ICP SLAM. Na obrázkoch 7.5a, 7.5b, 7.5c je možné pozorovať vyhladenie menšieho obrubníka, do ktorého robot narazil, keď nesprávne vyhodnotil terén ako nepriechodný. Nesprávne vyhodnotenie je kombináciou šumu v bodovom mraku, ďalším faktorom je taktiež rozlíšenie mapy, teda minimálna vzdialenosť bodov v bodovom mraku a tiež charakter prekážok v okolí obrubníka. Na tomto prípade je taktiež možné pozorovať nedostatok LIDAR senzora, ktorý detekuje trávu a napadané lístie ako statickú prekážku.

Obrázok 7.3: Klasifikácia obrubníka v reálnych podmienkach - na obrázku je možné pozorovať správne vyhodnotenie priechodnosti obrubníka a jeho okolia. V strede a ľavej strane mapy priechodnosti sa nachádza šum zanesený prechádzajúcou osobou.



(a) : Mapa priechodnosti parku.

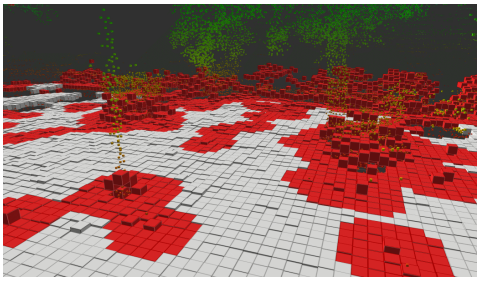
(b) : Fotka parku.

Obrázok 7.4: Mapa priechodnosti parku - na obrázku 7.5a je možné pozorovať správne vyhodnotenie nepriechodných prekážok a obrubníka v ľavej časti obrázku. Čiernou farbou je vyznačená trajektória robota, ktorú robot prejazdil pri zadaní viacerých cieľových polôh. Taktiež je možné pozorovať viacero miest, ktoré sú zašumené vplyvom dynamických objektov. V pravej časti sa nachádza nesprávne vyhodnotená oblasť na okraji bodového mraku.

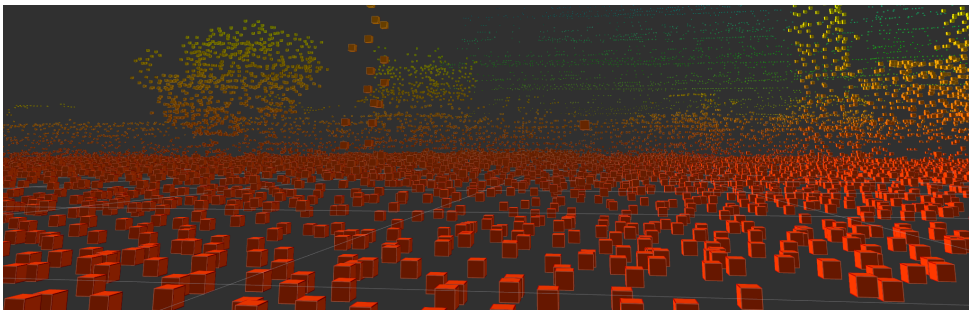
Po tejto jazde som upravil podmienky vyhodnocovania priechodnosti a boli zavedené parametre *procesing_distance* a *points_around_treshold*, popísané v podkapitole 6.3.2. Ďalej som do uzla MAPPER implementoval filter, založený na najbližších susedoch a taktiež som spravil úpravu filtrovania bodového mraku merania v uzle SLAM, spomenutú v podkapitole 6.3.1. Následne som si v laboratóriu postavil prekážky, na ktorých som sa snažil odladiť parametre uzlov na lepšiu detekciu menších prekážok. Na obrázkoch 7.6 je zobrazené porovnávanie kvality odstraňovania dynamických objektov z bodového mraku mapy a taktiež analýza priechodnosti prekážok. Na základe týchto testov bola upravená konštanta $max\_variance = 0.0015$, konštanta $beam\_half\_angle = 0.0025$ a nastavené parametre filtra založeného na najbližších susedoch na hodnoty $knn_1 = 2$, $knn_2 = 3$, $knn_3 = 6$, $dist_1 = 0.08\ m$, $dist_2 = 0.12\ m$ a $dist_3 = 0.19\ m$.

Záverečné jazdy prebehli v okolí budovy Národnej technickej knižnice v Prahe. Celkovo boli vykonané dva autonómne jazdy robota a jedna manuálne riadená jazda, ktorej cieľom bolo zmapovať jemne nerovný terén. Pri prvej

(a) : Fotka obrubnika.



(b) : Vyhodnotenie priechdnosti.

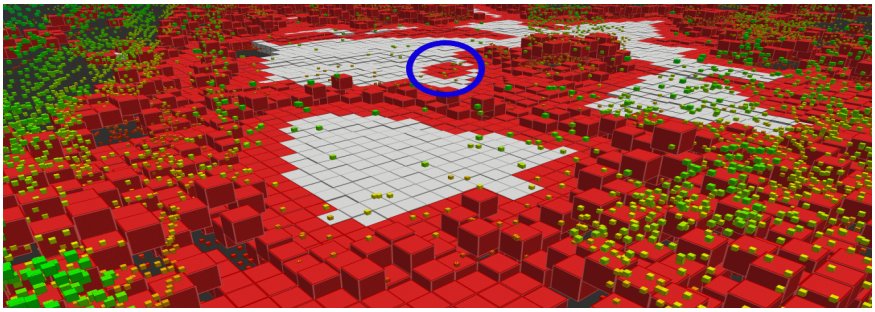


(c) : Bodová mapa v danom mieste.

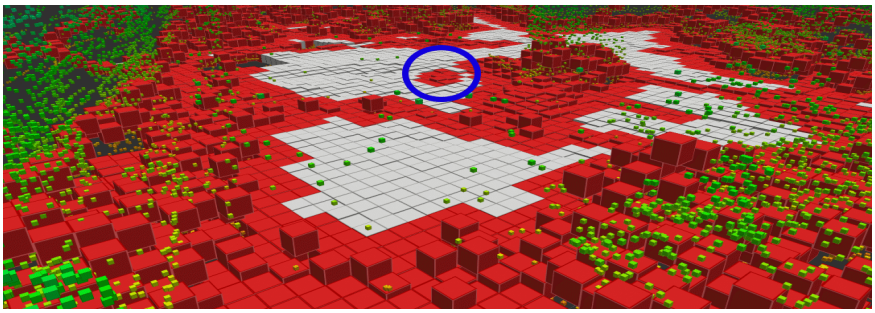
Obrázok 7.5: Prípad nesprávnej klasifikácie malého obrubníka - na bodovej mape je možné pozorovať vyhladenie obrubníka ICP SLAM algoritmom. V bodovom mraku je daný obrubník takmer nepozorovateľný a skôr sa javí ako mierny sklon terénu. Tento obrubník je v mape vyhodnotený ako nepriechodný. Na ľavo od obrubníka sa nachádza šum spôsobený prechádzajúcim menším psom.

jazde bola robotovi zadaná cieľová poloha v GNSS súradniciach. Robot pri tejto jazde úspešne urazil vzdialenosť približne 90 metrov, bez toho aby narazil do statických či dynamických prekážok. Celkovo pri tejto jazde prešlo okolo robota približne 10 ľudí. Robot sa zastavil približne pol metra od cieľovej polohy. Porovnanie mapy priechodnosti prostredia zo satelitnou mapou je zobrazené na obrázkoch 7.7. Vyhodnotenie priechodností zaujímavých prekážok pri jazde, je zobrazené na obrázkoch 7.9, 7.10. Pred druhou jazdou prestal v rámci ROS fungovať uzol GNSS a to vyradilo preposielanie správ z GNSS senzora do uzla MAPPER. Druhú cieľovú polohu som bol prinútený robotovi zadať v karteziánskych súradniciach. Robot bol pred cieľom predčasne zastavený, keďže sa mi už v tomto čase vybíjal laptop, z ktorého bol robot diaľkovo spúšťaný. Robota som zastavil, pretože na ceste od jeho polohy k cieľu sa už nenachádzal náročný terén. Pri tejto jazde robot opäť nenarazil a správne sa navigoval k cieľu okolo ľudí a prechádzajúceho automobilu. V jednom momente sa robot zastavil, pretože vyhodnotil bunku na ktorej práve bol ako nepriechodnú. Robota som preto musel posunúť o pár centimetrov do priechodnej oblasti. Po presunutí robot pokračoval v jazde k cieľovej polohe. Porovnanie mapy priechodnosti prostredia so satelitnou mapou je zobrazené na obrázkoch 7.8. Vyhodnotenie priechodností zaujímavých prekážok pri jazde, je zobrazené na obrázku 7.11 . Výsledky mapovania priechodnosti z manuálnej jazdy sú zobrazené na obrázku 7.12.

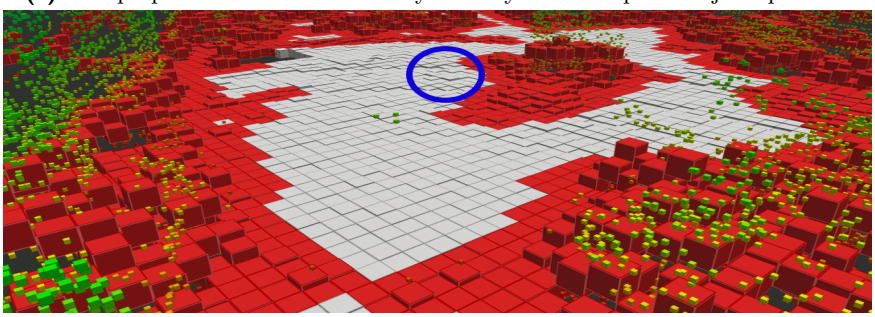
(a) : Fotka postavených prekážok.



(b) : Mapa priechodnosti bez filtra dynamických bodov.



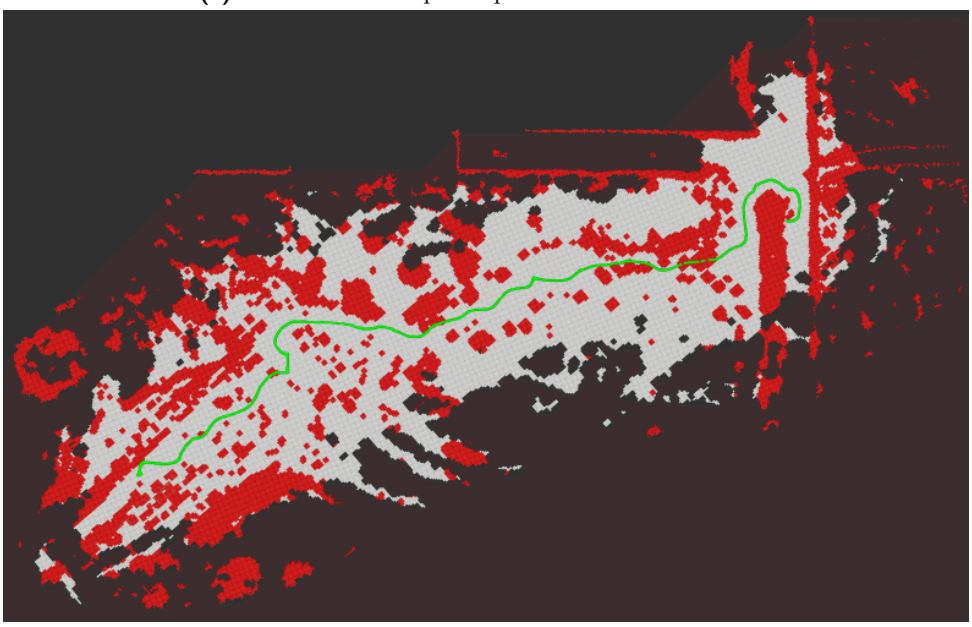
(c) : Mapa priechodnosti s filtrom dynamických bodov pred mojou úpravou.



(d) : Mapa priechodnosti s filtrom dynamických bodov po mojej úprave a s filtrom založenom na najbližších susedoch.

Obrázok 7.6: Porovnanie odstraňovania dynamických bodov - pred prekážkami som sa počas jazdy robota po laboratóriu prešiel, aby som simuloval dynamický objekt. V modrom kruhu je zobrazená prekážka na ktorej je vidieť efekt odstraňovania dynamických bodov na bodovú mapu. Prekážka je na obrázku 7.6d vyhodnotená ako prejazdná. Filtrovanie dynamických bodov ovplyvňuje všetky body v kuželi vyžarovaného laserového lúču. Ak je teda snímaný bod tesne za prekážkou, je možné že niektoré body na hranách danej prekážky budú z bodového mraku odstránené.

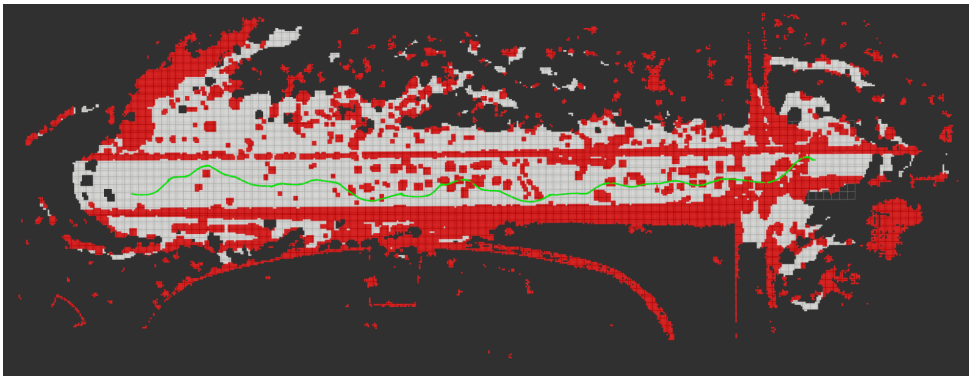
(a) : Satelitná fotka parku pri budove NTK v Prahe.



(b) : Mapa priechodnosti tohoto parku.

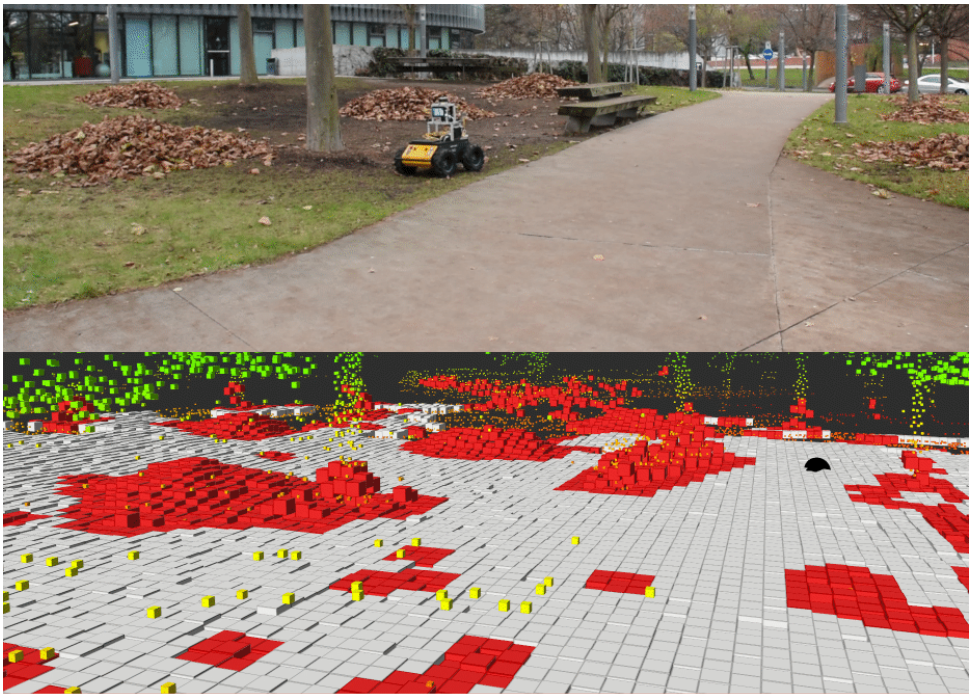
Obrázok 7.7: Porovnanie satelitnej mapy z mapou priechodnosti terénu - táto veľká mapa priechodnosti vznikla priebežným zlučovaním máp, ktoré vznikali pohybom robota. Zelenou farbou je označená trajektória, po ktorej sa robot pohyboval. V mape priechodnosti je možné pozorovať kontúry chodníka a kontúry mierne kopcovitého terénu na spodnej časti obrázku 7.8b. Mapa taktiež obsahuje množstvo nesprávne klasifikovaných buniek na miestach kde sa pohybovali dynamické objekty.

(a) : Satelitná fotka cesty pri budove NTK v Prahe.



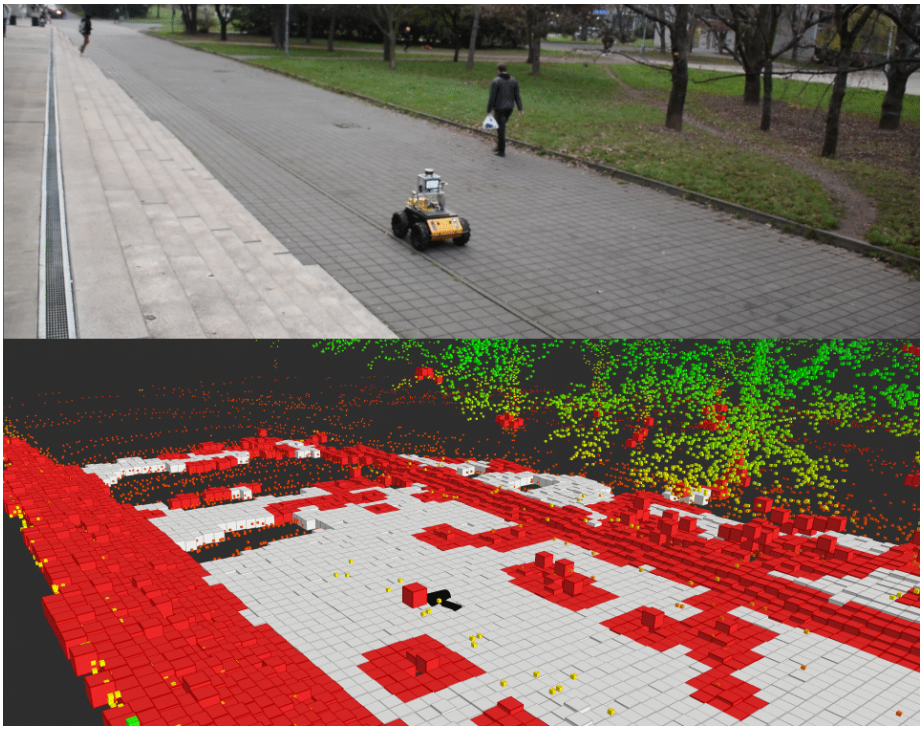
(b) : Mapa priechodnosti tejto cesty.

Obrázok 7.8: Porovnanie satelitnej mapy s mapou priechodnosti terénu - táto veľká mapa priechodnosti vznikla princípom popísaným v predošlom obrázku. Zelenou farbou je označená trajektória, po ktorej sa robot pohyboval. Na obrázku 7.8b je možné pozorovať zväčša správne vyhodnotené obrubníky, schody a veľké prekážky. Mapa obsahuje veľa šumu, spôsobeného osobami a prechádzajúcim automobilom.

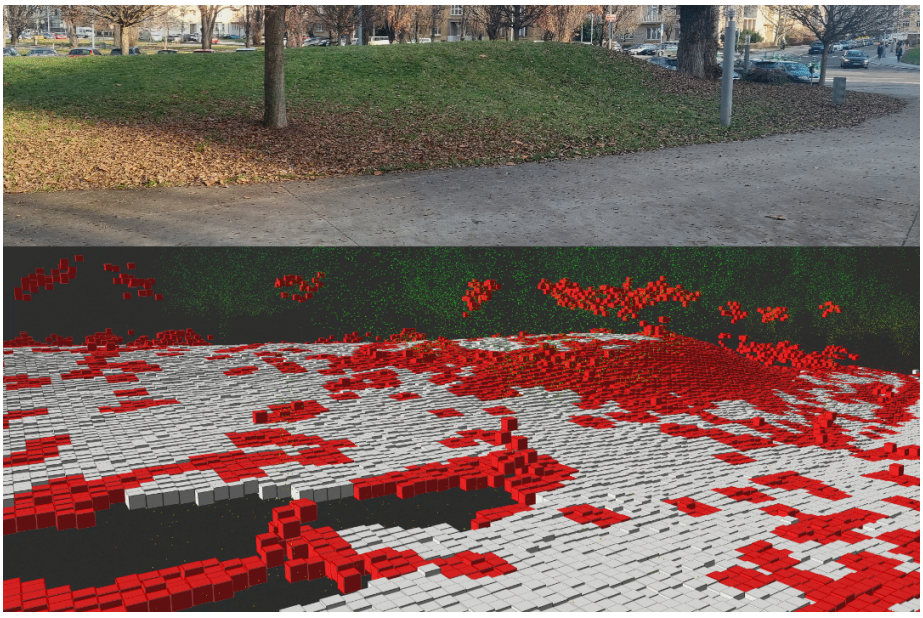


Obrázok 7.9: Mapa priechodnosti parku pri budove NTK - na obrázku je možné pozorovať správne vyhodnotenie veľkých prekážok. Ako malé kopčeky boli vyhodnotené oblasti, kde sa nachádza nahrabané lístie. V spodnej časti obrázku sa nachádza šum spôsobený dynamickými objektami.

Obrázok 7.10: Mapa priechodnosti cesty pri budove NTK - na obrázkoch je možné pozorovať správne vyhodnotené oblasti obrubníka a automobilov.



Obrázok 7.11: Mapa priechodnosti cesty pri budove NTK - na obrázku je možné pozorovať správne vyhodnotenie oblasti obrubníka a schodov. V mape je taktiež vidieť neodfiltrované body, nasnímané na osobe, pohybujúcej sa pred robotom.



Obrázok 7.12: Mapa priechodnosti mierne zvlneného terénu - na obrázku je možné pozorovať pozvoľný prechod priechodných a nepriechodných buniek v oblasti narastajúceho sklonu trávnatej plochy.

7.1 Rozbor výsledkov analýzy priechodnosti

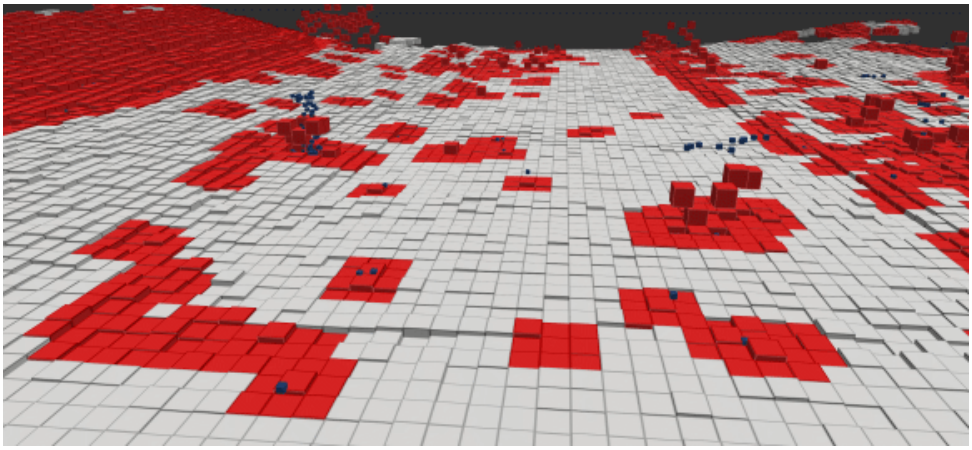
Analýza priechodnosti bola vo väčšine prípadov správna, pokiaľ sa jednalo o oblasti rovín a o prekážky s rozmerom na výšku väčším ako je 10 cm. Príčiny väčšiny nesprávnych klasifikácií buniek je možné rozdeliť do nasledujúcich kategórií:

- Šum v mape z dynamických objektov: ICP SLAM algoritmus potrebuje určitý počet iterácií na odstránenie dynamických bodov. Malé skupiny bodov, ktoré sa nachádzali nad plochami vo vzdialenosti viac ako 10 cm, boli vyfiltrované filtrom založeným na vzdialenosti k najbližším susedom. Problémom ostávajú dynamické body, nachádzajúce sa tesne nad plochou. Tieto body z bodového mraku spomínaný filter neodstráni, keďže majú svojich najbližších susedov príliš blízko. Výsledkom je klasifikácia bunky na nepriechodnú, aj keď sa v realite jedná o priechodnú oblasť. Jeden z prípadov tejto chyby je zobrazený na obrázku 7.13.

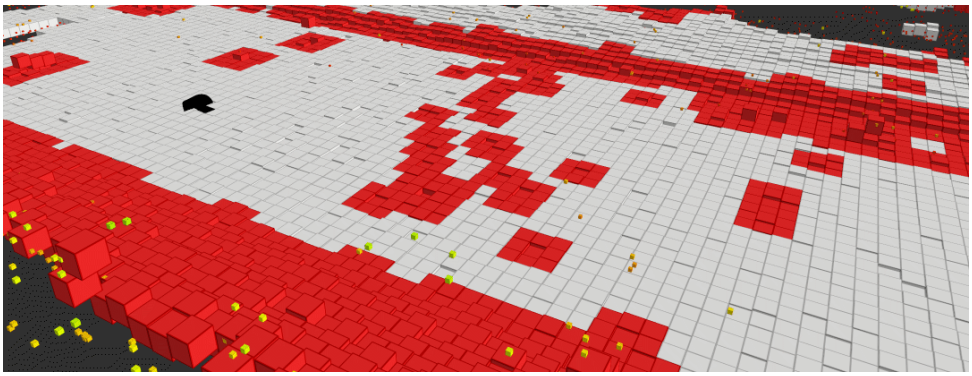
- Prekážky o veľkosti 5 až 10 cm: Do tejto kategórie spadajú nesprávne klasifikované nepriechodné oblasti. Tieto prekážky sú často vyhladené procesom odstraňovania dynamických bodov. Častým prípadom bol taktiež šum, spôsobený okolitou vegetáciou alebo napadaným lístim. Príklad nesprávnej, poprípade čiastočne správnej klasifikácie je zobrazený na obrázkoch 7.5, 7.6.

- Nesprávne pridané meranie do bodového mraku mapy: V niektorých prípadoch nastane pri pridaní nového merania do mapy, nie úplne presné preloženie merania s mapou. Tieto prípady často nastávali v momentoch, kedy sa okolo robota pohybovalo mnoho dynamických objektov súčasne. Body pridané pod alebo nad plochu tvorenú v bodovom mraku, ovplyvňujú hodnotu rozptylu bodov v danej bunke. Takéto bunky často presiahli stanovený limit rozptylu a boli nesprávne vyhodnotené ako nepriechodné. Príklad tejto situácie je zobrazený na obrázku 7.14.

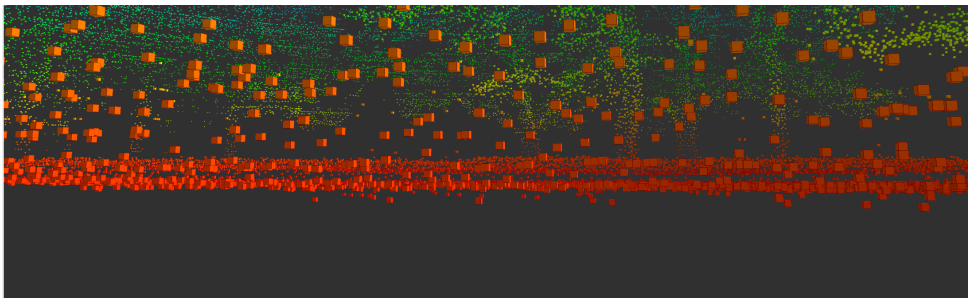
- Zlá predčasná klasifikácia na okrajoch bodového mraku: Na okrajoch bodového mraku sa často nachádzajú nepresné pridané merania. Postupným mapovaním sa tieto chyby opravujú a bodový mrak mapy lepšie reprezentuje reálne kontúry prekážok a plôch. Moja analýza priechodnosti tieto oblasti niekedy vyhodnocuje ako nepriechodné príliš skoro. Následne sa hľadanie cesty týmto oblastiam vyhne a môže nastať prípad, kedy dôjde k nesprávnemu vyhodnoteniu priechodného terénu. Tento problém bol z väčšej časti vyriešený zavedeným parametra *points_around_treshold*, ktorý určuje minimálny počet bodov v okolí bunky nato, aby sa mohlo rozhodnúť o priechodnosti danej bunky. Príklad tejto situácie bez použitia parametra *points_around_treshold* je zobrazený na obrázku 7.4. Pri použití tohto parametra dochádzalo k nesprávnej klasifikácií menej často a ak to nastalo, jednalo sa o oblasť tvorenú pár bunkami.



Obrázok 7.13: Príklad neodstránených bodov tesne nad plochou - na obrázku je možné pozorovať nesprávnu klasifikáciu priechodnej oblasti. Modrou farbou sú označené body zasadené do mriežky. V strede obrázku je možné pozorovať nevyfiltrované dynamické body, nachádzajúce sa tesne nad plochou chodníka.



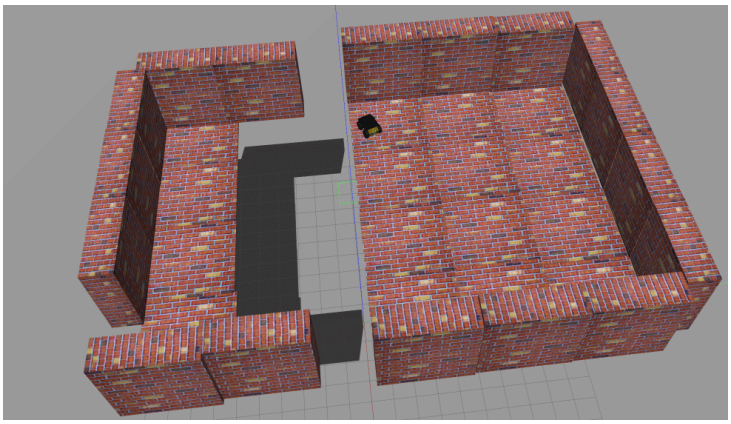
(a) : Mapa priechodnosti cesty.



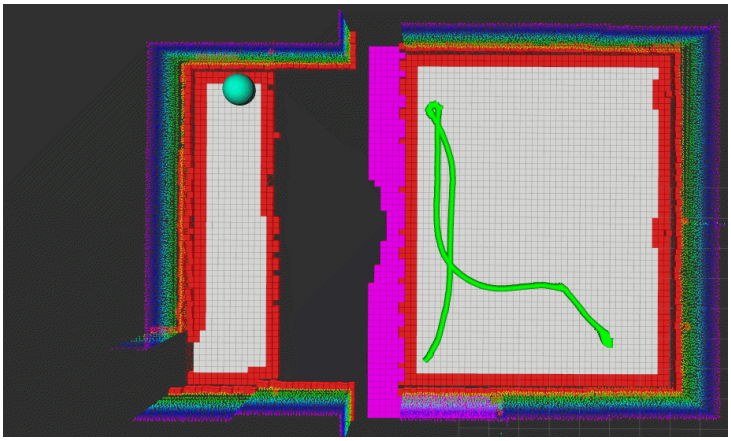
(b) : Nesprávne pridané body do mapy nachádzajúce sa pod cestou.

Obrázok 7.14: Príklad nesprávne pridaných bodov do bodovej mapy - nesprávne pridané body zapríčiňujú nesprávne vyhodnotenie oblasti cesty v strede obrázku 7.14a.

Pri reálnych testoch nenastal prípad, kde by bola použitá poloha robota ako propriocepčný senzor. V teréne, v ktorom sa s robotom jazdilo sa nenachádzali žiadne prepady a podobné prekážky. Rozhodol som sa túto metódu analýzy otestovať v simulácií v rámci simulátora Gazebo. Robot bol usadený na lietajúcu platformu a cieľová poloha bola zadaná na susednej lietajúcej platforme. Tieto platformy neboli spojené žiadnym povrchom a cesta medzi nimi neexistovala. Robot sa k cieľovej polohe snažil dostať, avšak po vyhodnotení priestoru medzi platformami pomocou svojej polohy na priestor nepriechodný, sa robot správne zastavil a vyhodnotil, že cesta neexistuje. Tento test je zobrazený na obrázku 7.15.



(a) : Testovacie platformy.



(b) : Mapa priechodnosti.

Obrázok 7.15: Test propriocepčnej analýzy na základe polohy robota - fialovou farbou sú označené nepriechodné bunky, ktoré boli vyhodnotené na základe polohy robota. Zelenou farbou je označená trajektória po ktorej sa robot pohyboval pri hľadaní cesty k cieľu, ktorý je označený svetlo modrou farbou.

Tento navigačný systém bol schopný bežať v reálnom čase. Pri jazdách v okolí budovy NTK trvalo spracovanie bodovej mapy a vyhodnotenie priechodnosti v priemere 546.669 ms. V najhoršom prípade tento proces trval 818.842 ms. Hľadanie cesty v mape priechodnosti trvalo v priemere 287.303 ms a v najhoršom prípade 547.014 ms. Hľadanie cesty bolo obmedzené na frekvenciu 2 Hz s cieľom poskytnutia čo najväčšieho výpočetného výkonu pre metódu ICP SLAM. Frekvencie preposielania dôležitých správ medzi uzlami v rámci ROS sú zobrazené v tabuľke 7.4. Z ohľadom na zvolenú maximálnu lineárnu rýchlosť $v_{max} = 0.5\ m \cdot s^{-1}$ a maximálnu rotačnú rýchlosť $\omega_{max} = 0.3\ rad \cdot s^{-1}$ je rýchlosť hľadania cesty a vyhodnotenia priechodnosti dostatočná. Robot medzi každou aktualizáciou urazí vzdialenosť približne 0.25 m.

typ správy a kanál	priemerná frekvencia
aktualizácia mapy z ICP SLAM `/map`	0.65 Hz
lokalizácia polohy `/icp_odom`	6.5 Hz
nájdená cesta v mape `/path`	1.4 Hz
požadované rýchlosti `/cmd_vel`	19.8 Hz

Tabuľka 7.4: Frekvencie s ktorými boli preposielané správy medzi uzlami.

Kapitola 8

Záver

Táto bakalárska práca obsahuje prehľad používaných techník SLAM a analýzy priechodnosti terénu. Hlavným zámerom tejto práce bolo implementovať metódu analýzy priechodnosti terénu, na základe ktorej by mohol byť postavený navigačný systém určený pre robota HUSKY A200. Navigačný systém pre robota HUSKY A200 je postavený na implementácií ICP SLAM v rámci ROS balíčka norlab_icp_mapper. Tento algoritmus poskytuje bodovú mapu prostredia spolu s aktuálnou polohou robota v danom prostredí.

Ako nadstavbu som implementoval systém, ktorý bodovú mapu spracuje na mapu priechodnosti pomocou geometrickej analýzy priechodnosti. Tento systém taktiež obsahuje možnosť hľadania cesty k zadanému cieľu spolu s generovaním požadovaných rýchlostí na sledovanie nájdenej cesty. Kombináciou môjho systému a SLAM metódy z balíčka norlab_icp_mapper vznikol navigačný systém, ktorý je schopný navigovať robota k cieľu v neznámom teréne pomocou odometrie, LIDAR senzora a GNSS senzora. Tento systém bol otestovaný v simulácií a v reálnych podmienkach.

Pri analýze priechodnosti som natrafil na určité typy prekážok, ktoré boli nesprávne vyhodnotené ako priechodné poprípade nepriechodné. Dosiahnuť presnejšiu reprezentáciu prostredia bodovou mapou by bolo možné použitím inerciálnej meracej jednotky, ktorá poskytuje presnejší odhad polohy oproti odometrii, počítanej z kolies vozidla. Presnejšia reprezentácia prostredia bodovou mapou by vylepšila presnosť klasifikácie buniek mapy priechodnosti. Systém by bol taktiež vhodnejší pre robota s vyššie postaveným podvozkom, poprípade pre pásového robota. Takýto robot by bol schopný prekonávať prekážky s väčším výškovým rozdielom oproti robotovi HUSKY A200. To by mi umožnilo zväčšiť limit rozptylu v bunkách mapy priechodnosti, čo by naopak zmenšilo vplyv šumu v bodovej mape a zvýšila by sa tak presnosť analýzy priechodnosti.

Hlavný cieľ práce považujem za splnený, keďže boli v rámci experimentov vykonané dve jazdy v reálnom prostredí, pri ktorých sa robot dokázal dostaviť do zadanej cieľovej polohy. V rámci práce bol taktiež neplánovane riešený problém odstraňovania dynamických bodov z generovanej mapy. V priebehu práce som za účelom robustnejšieho odstraňovania dynamických bodov upravil ICP SLAM z balíčka norlab_icp_mapper a implementoval filter bodového mraku založeného na najbližších susedoch. Tieto zmeny boli

úspešne otestované a vylepšili pôvodné odstraňovanie dynamických bodov na úkor orezávania prekážok v bodovej mape.

Ako pokračovanie práce sa ponúka použiť viaceré geometrické deskriptory, ako napríklad normály bodov voči povrchu, na robustnejšiu analýzu priechodnosti terénu. Do systému by taktiež mohla byť implementovaná analýza priechodnosti pomocou stereo kamery, čo by mohlo do určitej miery pomôcť s chybnými meraniami LIDAR senzora, ktoré nastávajú najmä na vegetácií nachádzajúcej sa v prostredí.

Bibliografia

[1] H. Hisahara, Y. Ishii, M. Ota, T. Ogitsu, H. Takemura a H. Mizoguchi, "Human Avoidance Function for Robotic Vacuum Cleaner through Use of Environmental Sensors: Roomba® Making Way for Humans", in *2014 5th International Conference on Intelligent Systems, Modelling and Simulation*, 2014, s. 64–67. DOI: 10.1109/ISMS.2014.19.

[2] D. Patil, M. Ansari, D. Tendulkar, R. Bhatlekar, V. N. Pawar a S. Aswale, "A Survey On Autonomous Military Service Robot", in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, 2020, s. 1–7. DOI: 10.1109/ic-ETITE47903.2020.78.

[3] J. Newman, Z. Sun a D.-J. Lee, "Self-Driving Cars: A Platform for Learning and Research", in *2020 Intermountain Engineering, Technology and Computing (IETC)*, 2020, s. 1–5. DOI: 10.1109/IETC47856.2020.9249142.

[4] B. Siciliano a O. Khatib, *Springer Handbook of Robotics*, 1. vyd. Berlin, Heidelberg: Springer-Verlag, 2007, ISBN: 354023957X.

[5] R. Mázl, "Lokalizace pro autonomní systémy", diz. pr., České vysoké učení technické, 2007.

[6] A. A. S. Souza, R. Maia a L. M. G. Gonalves, "3D Probabilistic Occupancy Grid to Robotic Mapping with Stereo Vision", in *Current Advancements in Stereo Vision*, A. Bhatti, ed., Rijeka: IntechOpen, 2012, kap. 9. DOI: 10.5772/49050. URL: https://doi.org/10.5772/49050.

[7] A. Elfes, "Using occupancy grids for mobile robot perception and navigation", *Computer*, roč. 22, č. 6, s. 46–57, 1989. DOI: 10.1109/2.30720.

[8] T. Nam, J. Shim a Y. Cho, "A 2.5D Map-Based Mobile Robot Localization via Cooperation of Aerial and Ground Robots", *Sensors*, roč. 17, č. 12, s. 2730, nov. 2017. DOI: 10.3390/s17122730.

[9] R. C. Smith a P. Cheeseman, "On the Representation and Estimation of Spatial Uncertainty", *The International Journal of Robotics Research*, roč. 5, č. 4, s. 56–68, 1986. DOI: 10.1177/027836498600500404. eprint:

https://doi.org/10.1177/027836498600500404. URL: https://doi.org/10.1177/027836498600500404.

[10] M. Golfarelli, D. Maio a S. Rizzi, "Elastic correction of dead-reckoning errors in map building", in *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190)*, Victoria, BC, Canada: IEEE, 2002.

[11] J. Konecny, P. Kromer, M. Prauzek a P. Musilek, "Scan Matching by Cross-Correlation and Differential Evolution", *Electronics*, roč. 8, č. 8, s. 856, aug. 2019, ISSN: 2079-9292. DOI: 10.3390/electronics8080856. URL: http://dx.doi.org/10.3390/electronics8080856.

[12] J. Wen, C. Qian, J. Tang, H. Liu, W. Ye a X. Fan, "2D LiDAR SLAM Back-End Optimization with Control Network Constraint for Mobile Mapping", *Sensors*, roč. 18, č. 11, s. 3668, okt. 2018, ISSN: 1424-8220. DOI: 10.3390/s18113668. URL: http://dx.doi.org/10.3390/s18113668.

[13] R. Munguía a A. Grau, "Monocular SLAM for Visual Odometry: A Full Approach to the Delayed Inverse-Depth Feature Initialization Method", *Mathematical Problems in Engineering*, roč. 2012, s. 676 385, sept. 2012, ISSN: 1024-123X. DOI: 10.1155/2012/676385. URL: https://doi.org/10.1155/2012/676385.

[14] M. Zaffar, S. Ehsan, R. Stolkin a K. M. Maier, "Sensors, SLAM and long-term autonomy: A review", in *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, Edinburgh: IEEE, aug. 2018.

[15] Ifte Khairul Alam Bhuiyan, "LiDAR Sensor for Autonomous Vehicle", en, Technische Universität Chemnitz, tech. spr., 2017. DOI: 10.13140/RG.2.2.16982.34887/1. URL: http://rgdoi.net/10.13140/RG.2.2.16982.34887/1.

[16] F. Pomerleau, F. Colas a R. Siegwart, "A Review of Point Cloud Registration Algorithms for Mobile Robotics", *Foundations and Trends in Robotics*, roč. 4, č. 1, s. 1–104, 2015. DOI: 10.1561/2300000035. URL: https://hal.archives-ouvertes.fr/hal-01178661.

[17] F. Pomerleau, F. Colas, R. Siegwart a S. Magnenat, "Comparing ICP variants on real-world data sets Open-source library and experimental protocol", *Autonomous Robots*, roč. 34, č. 3, s. 133–148, 2013. DOI: 10.1007/s10514-013-9327-2. URL: https://hal.archives-ouvertes.fr/hal-01143458.

[18] F. Pomerleau, S. Magnenat, F. Colas, M. Liu a R. Siegwart, "Tracking a depth camera: Parameter exploration for fast ICP", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, United States, 2011. DOI: 10.1109/IROS.2011.6048545. URL: https://hal.archives-ouvertes.fr/hal-01142622.

[19] W. Hou, D. Li, C. Xu, H. Zhang a T. Li, "An advanced k nearest neighbor classification algorithm based on KD-tree", in *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*, Chongqing, China: IEEE, dec. 2018.

[20] F. Pomerleau a S. Magnenat, *libpointmatcher*, `https://github.com/ethz-asl/libpointmatcher`, Citované: 2022-11-25.

[21] P. Babin, "Analysis of error functions for the iterative closest point algorithm", dipl. pr., Université Laval, 2019.

[22] P. W. Holland a R. E. Welsch, "Robust regression using iteratively reweighted least-squares", *Communications in Statistics - Theory and Methods*, roč. 6, č. 9, s. 813–827, 1977. DOI: 10.1080/03610927708827533. eprint: https://doi.org/10.1080/03610927708827533. URL: https://doi.org/10.1080/03610927708827533.

[23] D. Baril, S. Deschênes, O. Gamache et al., "Kilometer-scale autonomous navigation in subarctic forests: challenges and lessons learned", *CoRR*, roč. abs/2111.13981, 2021. arXiv: 2111.13981. URL: https://arxiv.org/abs/2111.13981.

[24] F. Pomerleau, P. Krüsi, F. Colas, P. Furgale a R. Siegwart, "Long-term 3D map maintenance in dynamic environments", in *IEEE International Conference on Robotics and Automation (ICRA)*, Hongkong, China, 2014. DOI: 10.1109/ICRA.2014.6907397. URL: https://hal.archives-ouvertes.fr/hal-01143106.

[25] P. Borges, T. Peynot, S. Liang et al., "A Survey on Terrain Traversability Analysis for Autonomous Ground Vehicles: Methods, Sensors, and Challenges", *Field Robotics*, roč. 2, č. 1, s. 1567–1627, mar. 2022. DOI: 10.55417/fr.2022049. URL: https://doi.org/10.55417/fr.2022049.

[26] F. L. Garcia Bermudez, R. C. Julian, D. W. Haldane, P. Abbeel a R. S. Fearing, "Performance analysis and terrain classification for a legged robot over rough terrain", in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura: IEEE, okt. 2012.

[27] P. Papadakis, "Terrain traversability analysis methods for unmanned ground vehicles: A survey", *Engineering Applications of Artificial Intelligence*, roč. 26, s. 1373–1385, apr. 2013. DOI: 10.1016/j.engappai.2013.01.006.

[28] D. Langer, J. Rosenblatt a M. Hebert, "A behavior-based system for off-road navigation", *IEEE Transactions on Robotics and Automation*, roč. 10, č. 6, s. 776–783, 1994. DOI: 10.1109/70.338532.

[29] D. B. Gennery, "Traversability Analysis and Path Planning for a Planetary Rover", *Autonomous Robots*, roč. 6, č. 2, s. 131–146, 1999. DOI: 10.1023/a:1008831426966. URL: https://doi.org/10.1023/a:1008831426966.

[30] J.-F. Lalonde, N. Vandapel, D. Huber a M. Hebert, "Natural terrain classification using three-dimensional Ladar data for ground robot mobility", *J. Field Robotics*, roč. 23, s. 839–861, okt. 2006. DOI: 10.1002/rob.20134.

[31] J. Larson, M. Trivedi a M. Bruch, "Off-Road Terrain Traversability Analysis and Hazard Avoidance for UGVs", University of California San Diego, tech. spr., jan. 2011.

[32] Y. Zhou, Y. Huang a Z. Xiong, "3D Traversability Map Generation for Mobile Robots Based on Point Cloud", in *2021 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2021, s. 836–841. DOI: 10.1109/AIM46487.2021.9517463.

[33] A. Howard, H. Seraji a E. Tunstel, "A rule-based fuzzy traversability index for mobile robot navigation", in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, Seoul, South Korea: IEEE, 2002.

[34] A. Howard a H. Seraji, "Vision-based terrain characterization and traversability assessment", en, *J. Robot. Syst.*, roč. 18, č. 10, s. 577–587, okt. 2001.

[35] L. Zhou, J. Wang, S. Lin a Z. Chen, "Terrain traversability mapping based on LiDAR and camera fusion", in *2022 8th International Conference on Automation, Robotics and Applications (ICARA)*, Prague, Czech Republic: IEEE, feb. 2022.

[36] G. Haddeler, M. Y. (Chuah, Y. You et al., "Traversability analysis with vision and terrain probing for safe legged robot navigation", *Frontiers in Robotics and AI*, roč. 9, 2022, ISSN: 2296-9144. DOI: 10.3389/frobt.2022.887910. URL: https://www.frontiersin.org/articles/10.3389/frobt.2022.887910.

[37] E. Alhamdi a R. Hedjar, "Comparative Study of Two Localization Approaches for Mobile Robots in an Indoor Environment", *Journal of Robotics*, roč. 2022, s. 1 999 082, jún 2022, ISSN: 1687-9600. DOI: 10.1155/2022/1999082. URL: https://doi.org/10.1155/2022/1999082.

[38] *Husky UGV - outdoor field research robot by clearpath*, https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot, Citované: 2022-12-12.

[39] *Velodyne VLP-16 datasheet*, https://www.amtechs.co.jp/product/VLP-16-Puck.pdf, Citované: 2022-12-12.

[40] *Trimble BX982 datasheet*, https://oemgnss.trimble.com/product/trimble-bx982/, Citované: 2022-12-12.

[41] S. Drake, "Converting GPS coordinates [phi, lambda, h] to navigation coordinates (ENU)", *DSTO*, roč. DSTO-TN, apr. 2002.

[42] P. E. Hart, N. J. Nilsson a B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *IEEE Transactions on Systems Science and Cybernetics*, roč. 4, č. 2, s. 100–107, 1968. DOI: 10.1109/TSSC.1968.300136.

[43] O. Amidi a C. E. Thorpe, "Integrated mobile robot control", Carnegie Mellon University Robotics Institute, tech. spr., 1991.

[44] *Path following pure pursuit scheme*, https://cw.fel.cvut.cz/b202/_detail/courses/aro/tutorials/path_following_pure_pursuit_scheme.png?id=courses%3Aaro%3Atutorials%3Ahomework04, Citované: 2022-12-25.

[45] T. Roucek, M. Pecka, P. Cızek et al., "System for multi-robotic exploration of underground environments CTU-CRAS-NORLAB in the DARPA Subterranean Challenge", *CoRR*, roč. abs/2110.05911, 2021. arXiv: 2110.05911. URL: https://arxiv.org/abs/2110.05911.