VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

LOKALIZACE MOBILNÍHO ROBOTA V PROSTŘEDÍ

LOCALISATION OF MOBILE ROBOT IN THE ENVIRONMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. DANIEL URBAN**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MARTIN VEĽAS**
SUPERVISOR

BRNO 2018

Zadání diplomové práce/20687/2017/xurban46

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Urban Daniel, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Lokalizace mobilního robota v prostředí**
Localisation of Mobile Robot in the Environment
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte základní metody zpracování obrazu a hloubkových dat. Zaměřte se na problematiku lokalizace v prostředí, v kterém se pohybuje mobilní robot, autonomní automobil a pod.
2. Získejte bližší pohled na techniky a existující řešení, které se v současnosti používají v oblasti lokalizace. Zaměřte se na vhodné datové struktury, robustní příznaky, efektivní prohledávání prostoru a detekci vizuálních smyček.
3. Navrhněte jednoduchý systém, který je schopný lokalizovat pozici robota na základě aktuálních senzorických 2D a 3D dat a záznamů z minulosti.
4. Implementujte navržený systém s vhodným využitím existujících řešení.
5. Získejte testovací datovou sadu pro testování.
6. Navrhněte a realizujte experimenty s Vaší implementací. Zhodnoťte dosažené výsledky a možnosti budoucího vývoje.
7. Vytvořte stručný plakát, který prezentuje obsah vaší práce a dosažené výsledky.

Literatura:

- Dle pokynů vedoucího.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Veľas Martin, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Táto diplomová práca sa zaoberá problematikou lokalizácie mobilného robota v prostredí, na základe senzorických 2D a 3D dát a záznamov z minulosti. Práca je zameraná na detekciu robotom už navštívených miest. Implementovaný systém je vhodný k detekcii slučiek, k čomu využíva 3D Gestalt deskriptory. Výstupom systému sú odpovedajúce pozície, na ktorých sa už robot nachádzal. Funkčnosť systému bola testovaná a vyhodnotená na LiDAR-ových dátach.

Abstract

This diploma thesis deals with the problem of mobile robot localisation in the environment based on current 2D and 3D sensor data and previous records. Work is focused on detecting previously visited places by robot. The implemented system is suitable for loop detection, using the Gestalt 3D descriptors. The output of the system provides corresponding positions on which the robot was already located. The functionality of the system has been tested and evaluated on LiDAR data.

Kľúčové slová

Lokalizácia mobilného robota, uzatváranie slučiek, detekcia slučiek, mobilný robot, rozpoznanie polohy, odometria, akumulácia chyby, deskriptory, kľúčové body, 3D Gestalt deskriptor.

Keywords

Localisation of mobile robot, loop closure, loop detection, mobile robot, place recognition, odometry, error accumulation, descriptors, keypoints, 3D Gestalt descriptor.

Citácia

URBAN, Daniel. *Lokalizace mobilního robota v prostředí.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Veľas

Lokalizace mobilního robota v prostředí

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Ing. Martina Veľasa.

.....
Daniel Urban
22. mája 2018

Poďakovanie

Týmto by som chcel poďakovať svojmu vedúcemu Ing. Martinovi Veľasovi za odbornú pomoc pri tvorbe tejto práce. Ďalej za kvalitné a časté konzultácie, počas ktorých som dostal mnoho rád, doporučení a pripomienok k práci.

Obsah

Kapitola 1

Úvod

Táto práca je zameraná na problém lokalizácie mobilného robota - jeden z kľúčových problémov mobilnej robotiky. V súvislosti s lokalizáciou sa hovorí o zistení vlastnej aktuálnej pozície mobilného robota pomocou dát, ktoré sú dostupné z rôznych senzorov. V tejto práci budú použité dáta zo senzoru LiDAR Velodyne.

Pozíciu mobilného robota je možné sledovať pomocou nazhromaždených informácií o jeho relatívnej zmene pozície v čase. Tu nastáva problém, že odhadnutá pozícia mobilného robota sa môže líšiť od skutočnej pozície v dôsledku akumulovanej odchýlky. K zmenšeniu takto vzniknutej odchýlky môže slúžiť detekcia a následné uzatváranie slučiek - to znamená schopnosť rozpoznania už navštívených pozícií.

Cieľom tejto práce je navrhnúť a vytvoriť systém, ktorý je schopný lokalizovať pozíciu robota na základe senzorických 3D dát a záznamov z minulosti. Následne je potrebné získať testovaciu dátovú sadu, zhodnotiť dosiahnuté výsledky a možnosti budúceho vývoja. Lokalizácia a rozpoznanie polohy robota sú riešené s využitím 3D Gestalt deskriptorov. Teoretická časť sa zaoberá základnými metódami lokalizácie v prostredí, kde sa pohybuje mobilný robot a tiež poskytuje prehľad o technikách a existujúcich riešeniach, ktoré sa v súčasnosti používajú v oblasti lokalizácie.

Kapitola 2 pojednáva o základných pojmoch pri lokalizácii a rozpoznaní polohy mobilného robota. Taktiež sa tu nachádza teoretický základ pre použitú metódu ako vysvetlenie odometrie, akumulácie chyby, voxelovej mriežky, párovania deskriptorov alebo problému uzatvárania slučiek.

State of the art je predstavený v kapitole 3, kde sú bližšie popísané už existujúce riešenia uzatvárania slučiek a taktiež v krátkosti uvedená metóda, na ktorej je založený návrh a implementácia systému v tejto práci.

V kapitole 4 je popísaný návrh systému pre lokalizáciu robota na základe senzorických dát. Návrh obsahuje model systému, kde sú popísané jeho jednotlivé časti.

Implementačné detaily, postupy a problémy, ktoré bolo potrebné vyriešiť sa nachádzajú v kapitole 5 a zhodnotenie implementovaného systému s dosiahnutými výsledkami sa nachádza v kapitole 6.

Kapitola 2

Lokalizácia mobilného robota a rozpoznanie polohy

Skôr ako prejdeme k samotnému popisu rozpoznania polohy a lokalizácie, je vhodné zadefinovať pojmy robot a robotika, ktoré úzko súvisia s touto problematikou.

- **Robot** je automatické zariadenie, ktoré je schopné reagovať na podnety okolia a na toto okolie spätne pôsobiť [7].
- **Robotika** je vedný odbor, ktorý sa zaoberá výskumom a vývojom robotov [7].

Jedným zo základných problémov mobilnej robotiky je lokalizácia. Lokalizácia by sa dala popísať odpoveďou na otázku *"Kde som?"*. Jedná sa o určenie polohy mobilného robota v prostredí. Táto informácia je potrebná pre riešenie rôznych kľúčových úloh robota, ako je napríklad orientácia v priestore, plánovanie cieľa a podobne.

2.1 Metódy lokalizácie

Metódy lokalizácie je možné rozdeliť podľa rôznych kritérií, či už podľa schopností, charakteru merania, mapy prostredia a iných. Lokalizačné metódy sa podľa charakteru merania delia na relatívne a absolútne [18]. Podľa mapy prostredia je možné metódy deliť na tie, ktoré majú mapu prostredia známu a lokalizácie bez známej mapy prostredia [25, 24].

- **Absolútne metódy lokalizácie** - Ich úlohou je určiť absolútnu polohu robota v prostredí, na základe jednorázového merania, nezávisle na predchádzajúcich meraniach. Poloha, ktorá je určená, má z pravidla podobu hustoty pravdepodobnosti výskytu robota na jednotlivých pozíciach priestoru, v ktorom sa má robot lokalizovať. Keďže sa využíva jednorázové meranie, prostriedky absolútnej lokalizácie nie sú zaťažené problémom akumulácie chyby. To znamená, že môžu slúžiť k oprave odhadu polohy. Medzi absolútne metódy patrí napríklad lokalizácia pomocou orientačných bodov. Orientačným bodom môže byť akýkoľvek príznak v prostredí, ktorý je detekovaný pomocou robota a ktorého absolútna poloha v priestore je robotovi známa. Radia sa tu aj globálne satelitné systémy ako napríklad Globálny Pozičný Systém (ďalej len GPS).
- **Relatívne metódy lokalizácie** - Pri relatívnych metódach sú na určenie polohy potrebné znalosti predchádzajúcej polohy robota v prostredí. Využíva sa posunutie a

rotácia v rovine oproti predchádzajúcej polohe. Následne sa výsledná zmena polohy určuje skladaním jednotlivých čiastočných zmien polohy, oproti počiatočnej alebo poslednej známej polohe. V prípade, že je známa počiatočná pozícia robota v prostredí, je možné tieto metódy krátkodobo využívať aj na absolútnu lokalizáciu. Relatívne metódy sa prevažne používajú pre odhad polohy robota na kratšie vzdialenosti. To je zapríčinené najmä tým, že v jednotlivých meraniach dochádza k nepresnostiam. Keďže je týchto meraní vykonávaných niekoľko, celková chyba sa v závislosti na prejdenej vzdialenosti zväčšuje. Patrí tu napríklad vizuálna odometria, ktorá bude popísaná nižšie.

V oblasti lokalizácie, mapovania a riadenia pohybu je veľmi dôležitým problémom rozpoznanie polohy a zistenie, či dané miesto už bolo pozorované. V prípade, že áno, ako tieto viaceré pozorovania súvisia. Spoľahlivé rozpoznanie polohy má mnoho využití v mobilnej robotike, pokiaľ ide o priestorovú reprezentáciu a odhad stavu. Vo všeobecnosti, efektívne riešenie pre rozpoznávanie miesta poskytuje tri základné funkcie [6]:

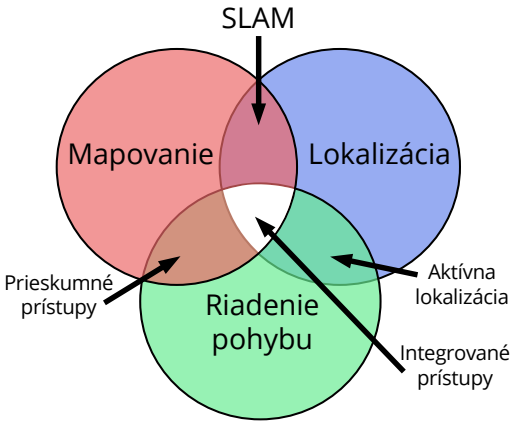
- **Globálna lokalizácia** - Umožňuje lokalizovať robota bez akejkoľvek znalosti o jeho pozícii. Jedná sa o schopnosť automaticky detekovať pozíciu senzora na mape v prípade neprítomnosti špecializovanej infraštruktúry, vrátane GPS.
- **Možnosť uzatvárať ľubovoľne veľké slučky v SLAM (Simultaneous Localization and Mapping) systémoch** - Registračné algoritmy zvyčajne vyžadujú hrubé počiatočné zarovnanie, aby sa priblížili k platnému riešeniu. Autonómny postup, schopný generovať spoľahlivé hrubé zarovnanie bez ohľadu na počiatočné registračné chyby, by umožnil zvýšiť rozsah dátových súborov, ktoré môžu byť spracované efektívnym spôsobom.
- **Schopnosť zlúčiť viacero dátových súborov, ktoré obsahujú určitý stupeň prekrytia** - Spoľahlivé a výkonné zlučovanie dátových súborov môže zvýšiť výkonnosť (viac robotov môže pracovať súčasne), flexibilitu (dáta môžu byť zozbierané v rozličných časoch) a validitu (získané chyby alebo anomálie môžu byť detekované a potenciálne opravené, ak aspoň jeden dátový súbor obsahuje v problémovej oblasti spoľahlivé dáta).

2.1.1 Získavanie modelov prostredia

Všeobecne platí, že úloha získavania modelov neznámych prostredí vyžaduje riešenie troch podúloh - mapovania, lokalizácie a riadenia pohybu. Pri úlohe mapovania nastáva problém integrácie informácie získanej pomocou senzorov robota do danej reprezentácie. V úlohe riadenia pohybu nastáva otázka, ako viesť robota, aby ho bolo možné účinne riadiť na požadované miesto alebo pozdĺž plánovanej trajektórie. Odhad pozície robota je problémom lokalizácie, ktorý už bol spomínaný vyššie. Na obrázku 2.1 je teda zobrazené prekrytie oblastí týchto troch úloh pomocou diagramu [19].

- **Simultánne Lokalizovanie a Mapovanie (ďalej len SLAM)** - Ide o problém budovania mapy, založený na odhade pozície, kde súbežne prebieha lokalizácia robota v rámci mapy, ktorá bola dovtedy vytvorená.
- **Aktívna lokalizácia** - Snaží sa riadiť robota do miest vnútri mapy k zlepšeniu odhadu pozície.

- **Prieskumné prístupy** - Na rozdiel od aktívnej lokalizácie, takzvané prieskumné prístupy sa zameriavajú na efektívne vedenie robota v prostredí, s cieľom tvoriť mapu.
- **Integrované prístupy** - Zameriavajú sa na mapovanie, lokalizáciu a riadenie pohybu súčasne. Nachádzajú sa v strede diagramu na obrázku 2.1.



Obr. 2.1: Diagram prekrytia mapovania, lokalizácie a riadenia pohybu.

2.2 Odometria a akumulácia chyby

Táto podkapitola je zameraná na odometriu ako najjednoduchšiu formu Dead-reckoningu (takzvaná navigácia výpočtom) a s ňou súvisiacu akumuláciu chyby [3, 18].

2.2.1 Dead-reckoning

Metóda, ktorá pracuje na základe merania prírastku zmeny polohy a orientácie robota sa nazýva Dead-reckoning. Ide o matematickú úlohu pre určenie aktuálnej polohy robota. Výsledná zmena polohy je určená skladaním jednotlivých čiastočných zmien polohy, oproti počiatočnej alebo poslednej známej absolútnej polohe robota. Nevýhoda tejto metódy je nárast polohovej chyby, ktorý je spôsobený prítomnosťou chýb v jednotlivých prírastkoch pri zmenách polohy.

2.2.2 Odometria

Odometria je najjednoduchšou formou Dead-reckoningu. Jedná sa o relatívnu lokalizáciu založenú na odhade zmeny pozície a orientácie robota s kolesami. Odhad zmeny pozície a orientácie robota získame prostredníctvom údajov o otáčaní jeho kolies, ktoré sú merané pomocou rotačných enkóderov (zariadenie, ktoré prevádza rotačný pohyb na elektrický signál) [4]. Jej podstata je založená na sčítaní jednotlivých prírastkov polohy v čase. Pohyb robota sa určí na základe prejdenej vzdialenosti a smeru, prípadne rýchlosti a času pohybu. Odometria poskytuje dobrú krátkodobú presnosť. V prípade, že sa robot pohybuje priamočiarym pohybom a po rovnej trajektórii, aktuálna pozícia sa vypočíta ako súčet predchádzajúcej pozície a zmeny v polohe, kde prejdená vzdialenosť je D v smere θ.

$$x_n + 1 = x_n + Dcos(\theta) \tag{2.1}$$

$$y_n + 1 = y_n + Dsin(\theta) \tag{2.2}$$

Odometria môže byť, v prípade známeho priebehu lineárnej rýchlosti ($V(t)$), uhlovej rýchlosti ($\omega(t)$), počiatočnej polohy (x_0, y_0) a orientácie robota (θ_0), definovaná aj presnejšie. Priebeh polohy a orientácia robota bude v tom prípade následovný:

$$x(t) = x_0 + \int_0^t V(t)cos\theta(t)dt \tag{2.3}$$

$$y(t) = y_0 + \int_0^t V(t)sin\theta(t)dt \tag{2.4}$$

$$\theta(t) = \theta_0 + \int_0^t \omega(t)dt \tag{2.5}$$

Medzi výhody odometrie sa radí trvalá schopnosť odhadovať polohu robota. Nevýhodou je, že generuje polohovú chybu.

2.2.3 Vizuálna Odometria

Ide o proces postupného odhadu pozície vozidla skúmaním zmien, ktoré sú vyvolané pohybom obrazu na palubných kamerách. Na rozdiel od odometrie kolies, vizuálna odometria nie je ovplyvnená prešmykovaním kolies, nerovným terénom alebo inými nepriaznivými podmienkami. Techniky vizuálnej odometrie umožňujú extrakciu informácie o odometrii z monokamery, stereokamery prípadne pomocou kamery Kinect. Ak je použitá monokamera je potrebné použiť techniku Structure From Motion, ktorá slúži k odhadu 3D štruktúry scény zo sekvencií 2D obrazov. Kamera Kinect dokáže spoločne s klasickým RGB obrazom dodávať mapu vzdialeností k predmetom danej scény. Z RGB kamery je potom s každým obrazom extrahovaný príznak v danom prostredí. K extrakcii príznakov je možné využiť napríklad konvolučné neurónové siete, rozdiel Gaussových funkcií (Difference of Gaussians) a iné. S ďalším obrazom kamery je tento príznak znovu zaznamenaný a sú porovnané vzdialenosti medzi polohou kamery a daným príznakom po sebe idúcich snímkov. Z týchto snímkov je vypočítaný výsledný pohyb kamery (resp. robota) a pozícia robota spoločne s mapou prostredia [23].

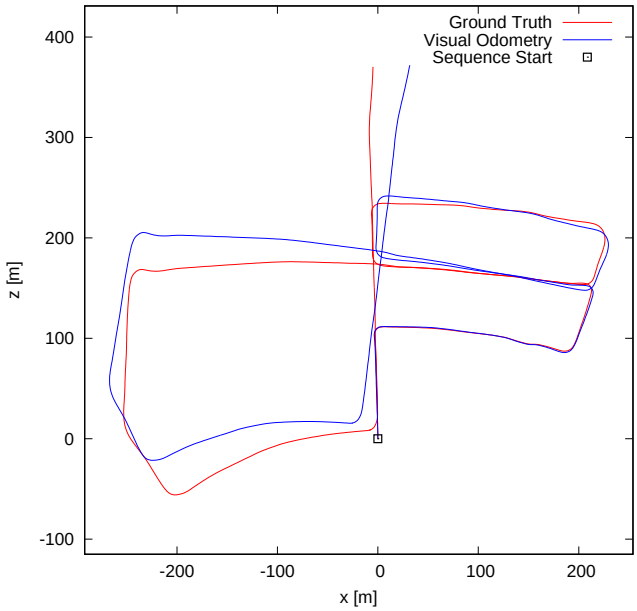
2.2.4 Akumulácia chyby

Ako už bolo spomenuté vyššie, odometria generuje polohovú chybu, ktorá neustále narastá. Táto akumulácia je spôsobená z rôznych chýb, ktoré je možné rozdeliť na systematické a nesystematické. K systematickým chybám patrí napríklad, vychýlenie kolies z priameho smeru alebo rôzne priemery kolies. Nerovný terén alebo prešmykovanie kolies, patrí zas k nesystematickým chybám.

Pri vizuálnej odometrii sa cesta kamery počíta postupne. Chyba sa akumuluje pri každom pohybe medzi obrazmi, čo vytvára odchýlku odhadovanej trajektórie od skutočnej cesty. Ako je možné vidieť na obrázku 2.2, chyba sa môže začať prejavovať veľmi rýchlo [23].

2.3 Uzatváranie slučiek

Slučka vzniká ak sa robot vráti na už navštívené miesto, s čím súvisi prvý krok pri uzatváraní slučiek a to je potreba rozpoznania miest, ktoré už boli navštívené. Problém uzatvárania



Obr. 2.2: Ukážka akumulácie chyby, skutočná cesta je znázornená červenou farbou.

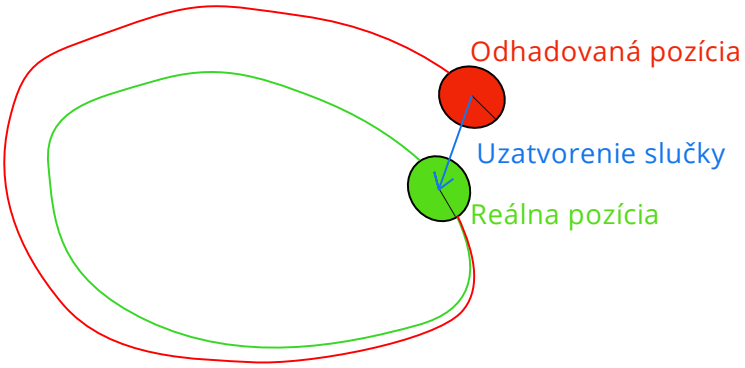
slučiek sa prejavuje po prechode robota neznámym terénom, pri návrate do už zmapovanej oblasti. V takejto situácii nastáva často stav, kedy pri prechode neznámeho prostredia dôjde k integrácii drobných chýb a znova navštívené miesto sa bude v tvorenej mape javiť ako nové miesto. Tento stav taktiež nastane, ak nie je známa počiatočná pozícia robota alebo ak dôjde k výpadku niektorého senzora robota. K týmto problémom využívajú vizuálne metódy obraz prostredia, ktorý nesie viac informácií, použiteľných k identifikácii lokácie.

Detekcia uzavretia slučky je účinný spôsob, ako eliminovať chyby a zlepšiť presnosť lokalizácie robota a mapovania. Ako je možné vidieť na obrázku 2.2, v každom kroku existuje určitý stupeň chyby pozorovania - akumulácia chyby. Na obrázku 2.3 je zobrazená schéma uzatvorenia slučky. Vďaka uzatváraniu slučiek je akumulovaná chyba značne znížená.

Detekciu už navštívenej pozície sťažuje problém detekcie nesprávnych kandidátov slučiek, ktorá môže viesť k zhoršeniu výsledkov. Zle detekované spoje sa dajú rozdeliť do dvoch kategórií. Jednou je, ak ide o správne slučky, ktoré by mali byť detekované, ale systém ich nedetekoval. Príčinou tejto chyby môže byť spôsob popisu obrazu, slabé osvetlenie, nedostatok kľúčových bodov (bodov v ktorých sa budú počítať deskriptory), prípadne rovnako vyzerajúce miesta. Iným prípadom sú nesprávne rozpoznané miesta. To znamená, že dve miesta, ktoré sú odlišné, vyzerajú podobne. Tento problém je možné redukovať pridaním verifikačného kroku do systému. Jednou z najzákladnejších metód, ako kontrolovať či sa jedná o nesprávne rozpoznanie, je porovnávanie nájdených kľúčových bodov. Pri porovnávaní kľúčových bodov je možné pre ich detekciu a extrakciu príznakov využiť napríklad metódy SURF,SIFT [25, 16, 30].

2.3.1 Point cloud

Point cloud alebo tiež mračno bodov je definované ako množina bodov, ktoré reprezentujú n-rozmerné dáta. Tieto jednotlivé body sú definované n súradnicami. V tejto práci budú uvažované jednotlivé body v trojrozmernom priestore, je definovaný troma priestorovými súradnicami označenými ako X, Y, Z. Mračná bodov je možné získavať z rôznych skenova-

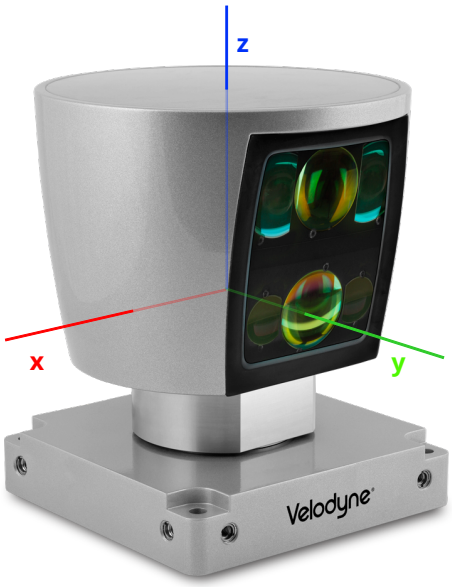


Obr. 2.3: Uzatvorenie slučky - redukcia chyby

cích zariadení, ako sú napríklad laserové senzory, a tým získať veľmi dobrú reprezentáciu priestoru [20].

2.4 Vstupné dáta

Vhodné vstupné dáta pre použitie v tejto práci sú dáta získané zo senzorov, konkrétne dáta z laserových senzorov. Vstupná sada, ktorá je vytvorená pomocou laserových senzorov obsahuje niekoľko množín bodov, kde každá takáto množina, pozostáva z niekoľko tisíc bodov. Tieto jednotlivé množiny bodov popisujú príslušné miesto, kde sa robot v tej chvíli nachádzal. Keďže ide o laserový senzor, vstupné dáta sú v trojrozmernom súradnicovom systéme, čo znamená, že každý bod obsahuje súradnice x, y, z. S takými množinami bodov je možné ďalej pracovať ako s mračnami bodov.

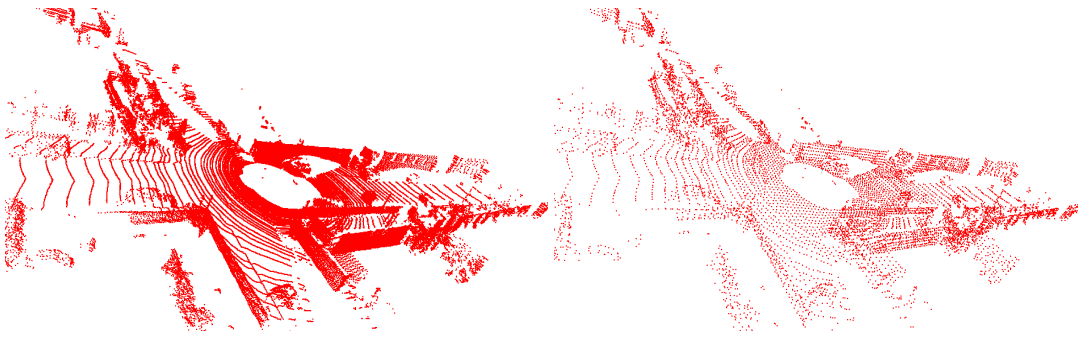


Obr. 2.4: Model Velodyne HDL-64E s naznačeným súradnicovým systémom.

Pre vstupné laserové dáta sú vhodné senzory Velodyne. Tieto senzory využívajú metódu diaľkového merania vzdialenosti LiDAR (Light Detection And Ranging). LiDAR pracuje na základe výpočtu doby šírenia pulzu laserového lúča odrazeného od snímaného objektu. Túto metódu je možné používať napríklad pre meranie vzdialenosti, mapovanie terénu alebo meranie vlastností atmosférických javov. Konkrétne technológia Velodyne sa zameriava na použitie LiDAR technológie v autonómnych vozidlách, bezpečnostných systémoch pre vozidlá, pri 3D mapovaní prostredia a podobne. Existujú rôzne modely Velodyne senzorov, ako napríklad Puck, HDL-32E, HDL-64E alebo VLS-128. V závislosti od použitého modelu je dosah až 300 metrov, horizontálne je pokryté okolie 360 stupňov a vertikálne je poskytnuté zorné pole od 30 do 40 stupňov. Na obrázku 2.4 je Velodyne senzor HDL-64E s naznačeným súradnicovým systémom, ktorý používajú senzory Velodyne LiDAR [1].

2.5 Podvzorkovanie s využitím voxelovej mriežky

Predtým ako je možné pracovať s pojmom voxelova mriežka (voxel grid), je potrebné zadefinovať pojem **voxel**. Jedná sa o objemovú časticu, ktorá reprezentuje hodnoty v pravidelnej mriežke v trojrozmernom prostredí.



Obr. 2.5: Podvzorkovanie s využitím voxelovej mriežky. Vľavo - plné mračno bodov. Vpravo - podvzorkované mračno bodov.

V tejto práci sa využíva podvzorkovanie pomocou voxelovej mriežky. Body, ktoré prislúchajú rovnakému voxelu v mriežke sa odstránia a nahradia jedným so súradnicami ich ťažiska. S využitím voxelovej mriežky je možné znížiť počet bodov v mračne bodov ako je možné vidieť na obrázku 2.5, kde v vľavo je mračno bodov pred podvzorkovaním a vpravo po podvzorkovaní. S takto získanou sadou dát je možné lepšie pracovať a ak je výsledkom podvzorkovania dobré rozloženie, vykonanie výpočtov bude rýchlejšie.

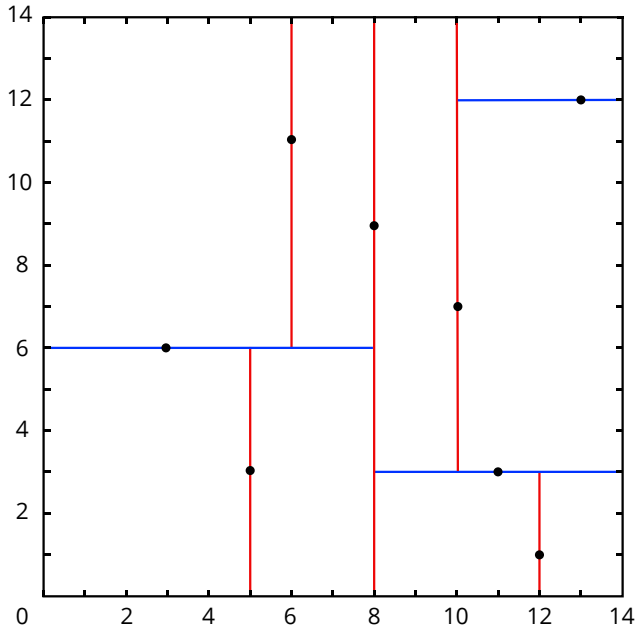
2.6 Vyhľadávanie susedov pomocou KdTree

Kd-strom (KdTree), alebo k-dimenzionálny strom je dátová štruktúra využívaná v informatike k organizovaniu bodov v priestore s dimenziou k. Táto užitočná dátová štruktúra sa využíva pre vyhľadávanie s multidimenzionálnym vyhľadávacím kľúčom, kde patrí napríklad vyhľadávanie s daným okolím alebo vyhľadávanie najbližších susedov. Vstupom kd-stromu

je množina bodov, prípadne zložitejších geometrických objektov v k-dimenzionálnom priestore [22].

Kd-strom je jedna z možností, kedy je priestor rozdeľovaný postupným spracovaním jednotlivých osí, tak že sa za bod delenia zvolí medián súradníc bodov v príslušnom podintervale, ktorý vznikol rozdelením predchádzajúceho intervalu. Bod delenia je definovaný ako bod, ktorým prechádza príslušná nadrovina, kolmá k danej osi. V prípade dvoch dimenzií sa uvažuje napríklad s osami x a y. Ukážku dvojdimenzionálneho rozdelenia priestoru je možné vidieť na obrázku 2.6.

Vyššie spomenutým postupom sa dospeje k vyváženému stromu, ktorý rozdeľuje priestor až na podpriestory obsahujúce jeden bod. Existujú aj rôzne alternatívy konštrukcie kd-stromu, kde sa vkladajú body postupne a tým delia príslušné priestory, no takéto stromy sa môžu jednoducho stať nevyváženými.



Obr. 2.6: Využitie Kd-stromu k rozdeleniu dvojdimenzionálneho priestoru.

V tejto práci je využité vyhľadávanie susedov okolo kľúčového bodu s rádiusom, pomocou KdTree. Keďže sa pracuje s mračnom bodov v troch dimenziách, pre účely tejto práce bude vhodné použiť trojrozmerný kd-strom. Vyhľadávanie bude potrebné k výpočtu deskriptora kľúčového bodu [13].

2.7 Deskriptory a ich párovanie

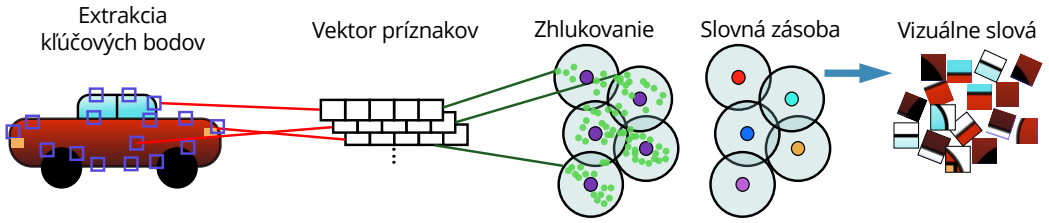
Deskriptor je pole hodnôt, ktoré určitým spôsobom popisuje okolie bodu v obraze, mračne bodov či surfeloch, za účelom jeho neskoršieho rozpoznania.

Dôležitou úlohou pri deskriptoroch zohráva ich párovanie. Účelom je spárovanie zakódovanej, dostatočne odlišnej lokálnej štruktúry v obrazovej informácii/mračne bodov s jej odpovedajúcou štruktúrou v inej obrazovej informácii/mračne bodov. Medzi najjednoduchšie metódy párovania deskriptorov sa radí metóda hrubej sily, kde sa porovnáva každý deskriptor s každým. Metódy pre párovanie deskriptorov sú založené na Euklidovskej vzdia-

lenosti, Hammingovej vzdialenosti, prípadne kosínusovej podobnosti. Dôležitou úlohou pri párovaní je funkčnosť aj v prípade zmien orientácie, merítka, kontrastu a podobne.[17]

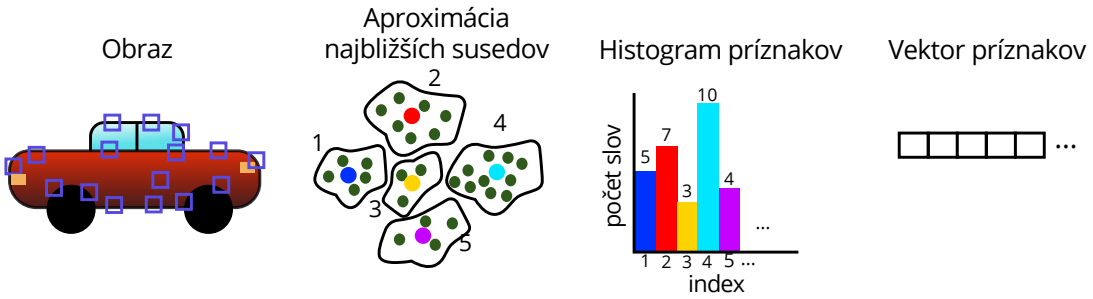
2.8 Bag-of-Words a Bag-of-Raw-Features

Jedná sa o jednu z populárnych obrazových metód, ktorá sa zaoberá uzavretím slučiek. BoW prístup využíva proces, pri ktorom sú príznaky z trénovacích obrazov extrahované a deskriptory zoskupené do zhlukov. Zhlukovaním sa rozdelí množina na disjunktné podmnožiny, keďže množina všetkých deskriptorov je príliš veľká. Centrá zhlukov potom slúžia



Obr. 2.7: Vizuálna slovná zásoba je získaná extrakciou deskriptora príznakov z reprezentatívneho obrazu.

ako vizuálne slová a kolekcia týchto slov tvorí vizuálny slovník alebo slovnú zásobu viď. 2.7 [26]. Z daného obrazu sú vizuálne príznaky vektorovo kvantizované s využitím algoritmu najbližšieho suseda (navštívený je uzol, ktorý je najbližšie). Obrazový deskriptor je potom vytvorený na základe histogramu vizuálnych slov, ktoré sa objavili v obraze. (viď Obrázok 2.8) Kandidátne obrazy, ktoré sú podobné danému obrazu môžu byť efektívne získané s



Obr. 2.8: Detekcia a extrakcia príznakov z obrazu a využitie aproximácie najbližších susedov ku konštrukcii histogramu príznakov pre každý obraz.

využitím inverzného indexu. Vizuálny slovník je tvorený offline. Online prebieha extrakcia príznakov, vyhľadávanie najbližšieho suseda a indexovanie v danom obraze. K **uzatváraniu slučiek** dochádza, keď aktuálny pohľad robota zodpovedá jednej z kľúčových snímkov v jeho mape. S metódou BoW sú však spojené aj dva kľúčové nedostatky. Prvým je už spomínaný offline krok k vytvoreniu vizuálneho slovníka z trénovacích obrazov, kde trénovanie obrazov nemusí predstavovať zodpovedajúcim spôsobom budúce výhľady robota. Druhým nedostatkom je krok vektorovej kvantizácie, ktorá prevádza vizuálne príznaky do vizuálnych slov požadovaných indexáciou, čo spôsobuje tzv. *perceptual aliasing*. To je prípad, keď je napríklad vysoká podobnosť medzi rôznymi miestami. Následkom čoho je to, že párovanie

príznakov vektorovej kvantizácie nemusí vybrať správnych kandidátov uzatvorenia slučiek. Pre overenie nesprávne rozpoznaných kandidátov sa využíva multiple-view geometry (ďalej len MVG) [14, 28, 29].

Metóda Bag-of-Raw-Features je prezentovaná v [29] a je zameraná na problém online párovania obrazov pre účely detekcie uzatvorenia slučiek. Metóda nie je závislá na vizuálnom slovníku a používa výber podmnožiny vizuálnych príznakov priamo pre popis obrazu, bez vektorovej kvantizácie. Využíva sa tu popis deskriptorov, ktorý je invariantný voči rotácii a vzdialenosti od kamery popisovaného objektu (SIFT alebo SURF). Tento deskriptor sa ukázal sa ako hlavný prístup k extrakcii príznakov v celej rade aplikácii, vrátane navigácie robota. BoRF (Bag-of-Raw-Features) nevyžaduje offline tvorbu vizuálneho slovníka, ako to bolo u BoW. Problém nastáva pri riešení zložitosti spojenej s veľkým počtom príznakov po výbere príznakov, pri extrakcii príznakov a aj pri popise obrazu. Slabinou tejto metódy je to, že zložitosť rastie s počtom obrazov, ktoré definujú mapu robota. Tento problém je možné riešiť vhodným výberom príznakov. Metóda poskytuje výborný detekčný výkon s veľmi vysokou presnosťou a nemusí sa využívať MVG overenie nesprávne rozpoznaných kandidátov. Vďaka použitiu vysoko diskriminačných príznakov deskriptorov pri porovnaní obrazov sú produkované lepšie výsledky pri detekcii uzatvárania slučiek, ako to bolo pri metóde BoW. Vďaka tomu je táto metóda preferovaná v existujúcich metódach, kde je detekcia uzatvorenia slučiek kritická.

Kapitola 3

State of the art

Detekcia situácii, kedy robot navštívil už skôr navštívenú oblasť (spomínaný problém uzatvárania slučiek), je základným problémom v simultánnej lokalizácii a mapovaní - SLAM. Informácia o detekcii slučiek je potrebná k tomu, aby robot mohol znížiť a obmedziť chyby a nepresnosti v odhadovanej pozícii a mape okolia.

3.1 Metódy založené na Bag-of-Words

Teoretický úvod k BoW sa nachádza v kapitole 2.8. Táto kapitola sa venuje prehľadu metód, ktoré reálne využívajú metódu BoW pri detekcii uzatvorenia slučiek.

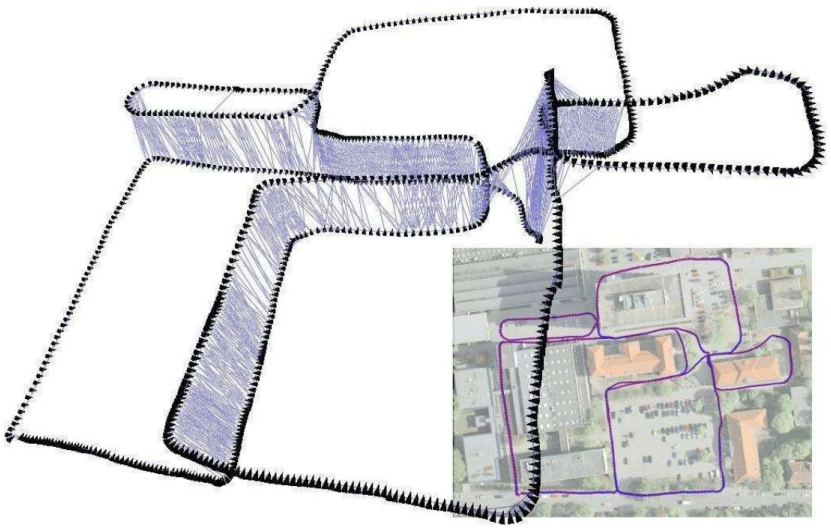
Autor Angeli a kolektív, vo svojej práci [2] využili BoW k definícii obrazového deskriptora a vypočítali hodnotu likelihood aktuálneho obrazu, s ohľadom na jeden z predchádzajúcich pozorovaných kľúčových snímkov. Likelihood slúži ako kľúčový prvok Bayesovho frameworku pre detekciu uzatvorenia slučiek.

Newman a Cummins vo svojích prácach [9], [8], pre Bayesov framework k detekcii uzatvorenia slučiek taktiež využívajú koreláciu medzi vizuálnymi slovami pri výpočte hodnoty likelihood.

Fraundorfer implementoval vo svojej práci [11] vizuálny navigačný algoritmus, ktorý využíva strom slovnej zásoby (vocabulary tree) [21], takže daný obraz môže efektívnejšie indexovať vizuálnu slovnú zásobu (visual vocabulary).

V každej z týchto implementácii boli použité, či už príznaky invariantné voči rotácii a vzdialenosti od kamery popisovaného objektu (SIFT a SURF), alebo afinné kovariančné príznaky (MSER) ako vizuálne príznaky. K-means alebo jeho stromová implementácia bola použitá k vektorovej kvantizácii príznakov deskriptorov.

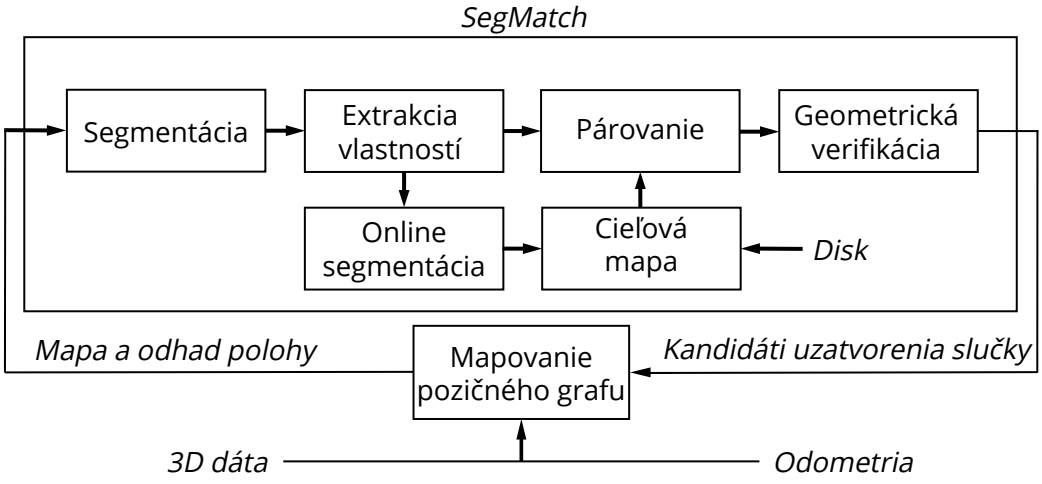
V práci [27] Steder a kolektív prezentovali algoritmus založený na 3D dátach, ktorý dokáže spoľahlivo odhaliť už navštívené oblasti a zároveň vypočíta presnú transformáciu medzi príslušnými nájdenými dvojicami. Systém, ktorý vytvorili, používa na vyhodnotenie kandidáta odhad transformácie, čím robustnejšie odmieta falošné alarmy pri rozpoznávaní miesta. BoW prístup tu slúži ako krok predspracovania k dosiahnutiu vyššej výkonnosti. Na obrázku 3.1 je možné vidieť príklad využitia systému, kde sú vypočítané transformácie použité na uzatváranie slučiek v pozičnom grafe.



Obr. 3.1: Výsledky systému pre rozpoznávanie miest, použité na dátovej sade Hanover2. Čierné uzly predstavujú graf trajektórie a modré/sivé čiary detekované slučky [27].

3.2 SegMatch

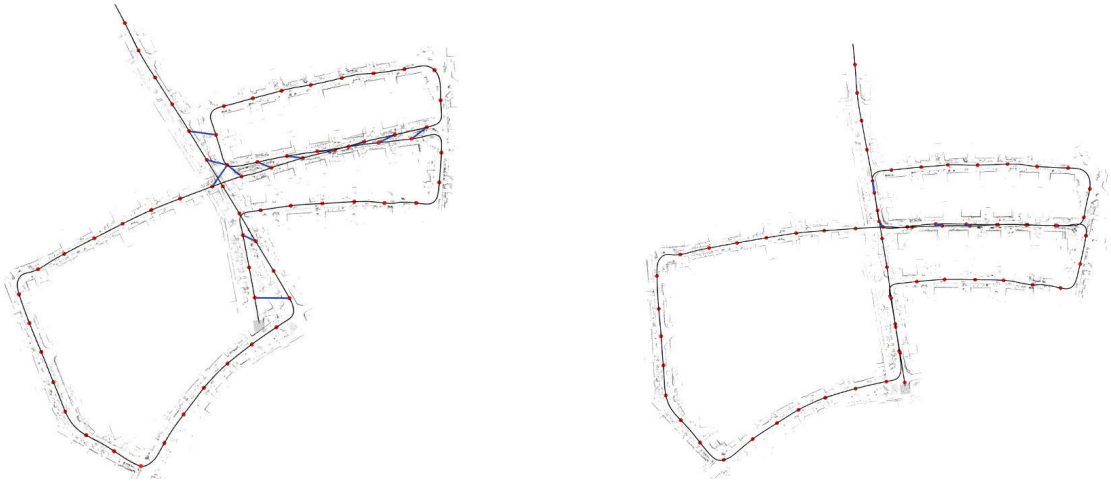
Jedná sa o metódu, ktorá umožňuje autonómnym vozidlám rozpoznať už navštívené oblasti založené na extrakcii a párovaní 3D segmentov [10]. Táto dôležitá schopnosť autonómnych vozidiel lokalizovať samých seba vo svojom prostredí sa používa pre stavbu presných 3D máp a pre plnenie rôznych úloh, ako je napríklad riadenie vozidla do cieľa.



Obr. 3.2: Blokový diagram SegMatch, algoritmu pre detekciu slučiek.

Metóda SegMatch je založená na detekcii segmentov v blízkosti vozidla a na párovaní týchto segmentov s už extrahovanými segmentami z cieľovej mapy. Tieto zhody segmentov môžu byť priamo preložené do presných informácii lokalizácie, ktoré umožňujú presnú konštrukciu

3D mapy. Segmenty poskytujú kompromis medzi lokálnymi a globálnymi popismi, zahrňujú ich silné stránky a redukujú ich individuálne nevýhody. Cieľová mapa môže byť vytvorená z reálnych údajov zaznamenaných laserom (je možné použiť napríklad dáta z KITTI datasetu). Cieľová mapa môže byť načítaná skôr alebo môže byť tvorená online. Systém SegMatch pozostáva zo štyroch rôznych modulov: segmentácia mračna bodov, extrakcia príznakov, párovanie segmentov a geometrická verifikácia, čo je zobrazené na obrázku 3.2. V prípade uzatvárania slučiek, trajektória robota je znovu odhadnutá a pozície cieľových segmentov sú aktualizované s tým, že je známy pôvod segmentácie vzhľadom k trajektórii. Segmenty pridané do cieľovej mapy môžu byť duplikátmi starších segmentov, čo znamená, že sú z rovnakej časti objektu ale boli segmentované v rôznych časoch. Keďže odometria je lokálne presná, môžu byť tieto duplikáty efektívne detekované porovnávaním vzdialeností stredov najbližších segmentov. Ak je detekcia úspešná, budú segmenty cieľovej mapy správne zarovnané a môže byť bezpečne vykonaná filtrácia k odstráneniu duplikátov. Filtrovanie sa uskutočňuje od najnovších po najstaršie segmenty.



Obr. 3.3: Uzatváranie slučiek pomocou SegMatch [10].

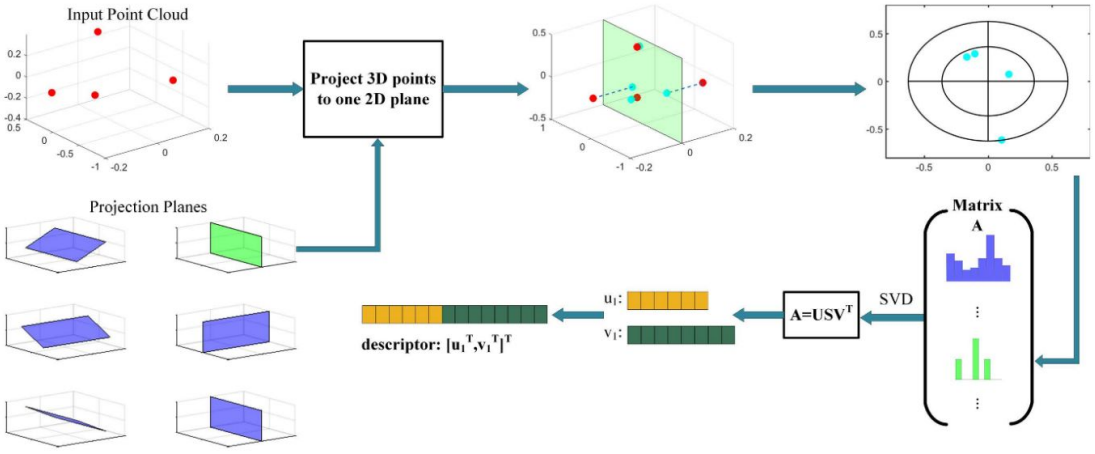
Na obrázku 3.3, je možné vidieť ilustráciu uzatvárania slučiek pomocou metódy SegMatch. V ľavej časti je možné vidieť slučky, ktoré boli detekované v reálnom čase. Červené bodky predstavujú lokácie, kde bola vykonaná segmentácia a detekcia uzatvárania slučiek. Modré čiary znázorňujú nájdené slučky. V pravej časti je znázornený výsledok po zarovnaní.

3.3 M2DP deskriptor

Globálny deskriptor M2DP pre 3D mračná bodov je prezentovaný a aplikovaný na problém detekcie uzatvárania slučiek v práci [15]. V M2DP sa 3D mračno bodov premietne do viacerých 2D rovín a pre každú rovinu je generovaný popis hustoty bodov. Ľavé a pravé singulárne vektory popisu sú použité ako deskriptor 3D mračna bodov.

Pre pochopenie navrhnutej metódy môže byť M2DP vysvetlený na dvojrozmernom príklade a následne je možné túto myšlienku použiť aj na viac ako dva rozmery. Uvažujme, že máme 3D mračno bodov P a dve 2D roviny X a Y. Označíme premietnuté body mračna bodov P na roviny X a Y ako P_x a P_y. Za predpokladu, že X a Y nie sú paralelné a pri premietaní nedochádza k žiadnej oklúzii, je možné rekonštruovať P z P_x a P_y a uhol tiež medzi rovinami X a Y. Odvodením popisu P_x ako v_x a P_y ako v_y, je možné reprezentovať

P popisnou maticou $A = [v_x^T v_y^T]^T$, takže je možné porovnať dve mračná bodov na základe ich popisných matíc. V prípade, ak je rovín viac, je získaná rozšírená popisná matica A. K dosiahnutiu kompaktného popisu, je možné použiť rozklad na singulárne hodnoty (Singular values decomposition) k zmenšeniu počtu dimenzií. Ako už bolo vyššie spomenuté použije sa prvý ľavý a pravý singulárny vektor popisovej matice A ako deskriptor mračna bodov. Tento algoritmus je podrobnejšie rozobratý v práci autorov Li He, Xiaolong Wang a Hong Zhang [15]. Na obrázku 3.4 je možné vidieť grafickú štruktúru algoritmu M2DP.



Obr. 3.4: Schéma algoritmu M2DP [15].

3.4 3D Gestalt deskriptor

V práci [6] autori Bosse a Zlot extrahujú kľúčové body priamo z mračna bodov a popisujú ich pomocou 3D Gestalt deskriptora.

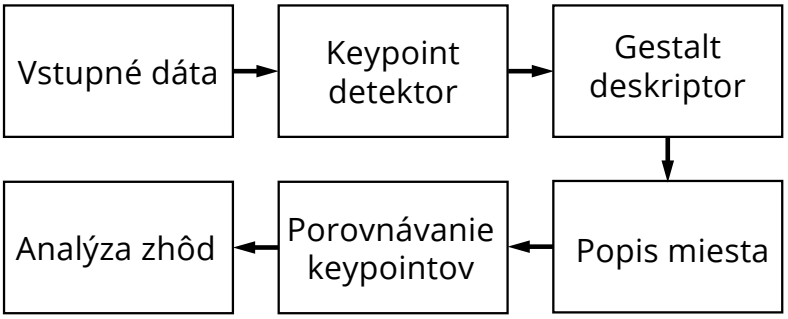
Gestalt deskriptory - Jedná sa o štrukturálne deskriptory, ktoré umožňujú rozpoznanie miest výlučne na základe 3D bodov. Vzhľadom na to, že jeden bod nie je diskriminačný, štrukturálne deskriptory sa spoliehajú na získavanie príznakov vyššej úrovne z mračna bodov. Kľúčové body potom hlasujú pre ich najbližších susedov v takzvanej hlasovacej matici (vote matrix), ktorá je finálne prahovaná pre rozpoznanie miest.

Táto práca je inšpirovaná publikáciou vyššie spomínaných autorov. Prostredníctvom senzorických dát bol implementovaný systém pre lokalizáciu pozície robota. Návrh a implementácia budú popísané v nasledujúcich kapitolách.

Kapitola 4

Návrh

V tejto kapitole sa nachádza popis systému, založený na teoretických znalostiach z predchádzajúcich kapitol, ktorého úlohou je na základe senzorických dát lokalizovať pozíciu robota v prostredí, ktoré už navštívil. Systém vychádza z práce [6]. Prvým krokom návrhu bolo vytvorenie základnej schémy systému 4.1.



Obr. 4.1: Základná schéma systému.

4.1 Načítanie vstupných dát

Ako je uvedené v kapitole 2.4, vstupné dáta sú získané z laserových senzorov, takže sa nachádzajú v trojdimenzionálnom súradnicovom systéme. Takéto množiny 3D bodov (mračná bodov) popisujú jednotlivé miesta, kde sa robot nachádzal.

4.2 Detektor kľúčových bodov

Implementácia detektoru kľúčových bodov (keypoint detektoru) bude dôležitou časťou, keďže kľúčové body určujú, kde a v akej časti sa počíta deskriptor.

Pred samotným procesom výberu kľúčových bodov je vhodné vykonať krok predspracovania, ktorým je aplikácia voxelovej mriežky so zadaným rozlíšením na obsiahle mračno bodov. Tým dôjde k zredukovaniu počtu bodov v mračne bodov, ako je možné vidieť na obrázku 2.5.
Takto upravenú sadu dát bude možné ďalej spracovávať k získaniu finálnych kľúčových bodov. K tomu sa využijú viaceré kroky, medzi ktoré patrí:

- **Náhodný výber desiatich percent kľúčových bodov** - Zo zredukovanej sady dát sa vyberie náhodným výberom desať percent bodov, ktoré sa prehlásia za kľúčové body.

- **Určenie orientácie kľúčových bodov a natočenie okolia** - K určeniu orientácie kľúčového bodu je vhodné vypočítať kovariančnú maticu 3×3 z okolia daného kľúčového bodu. K určeniu okolia bude použitá **KdTree** vyhľadávacia štruktúra, ktorá okrem iného slúži aj na vyhľadávanie susedných bodov v danom rádiuse. Takto spočítanú kovariančnú maticu je vhodné rozložiť na vlastné čísla a vlastné vektory. Potom sa použije orientácia najmenšieho vlastného vektora k rotácii okolia kľúčového bodu a to následovne. Najmenší vlastný vektor sa premietne na horizontálnu rovinu a tento vektor, ktorý leží v horizontálnej rovine, bude nová x-ová súradnica. To spôsobí, že x-ová súradnica celého okolia bude natočená v smere najmenšieho vlastného vektora. Ak sa teda vrátime k tomu istému kľúčovému bodu, ale napríklad z opačnej strany, tak okolie bude natočené rovnakým smerom. Prakticky je potrebné k rotácii vykonať 2 kroky. Nastaviť y-ovú súradincu na 0, aby sa najmenší vlastný vektor premietol na horizontálnu rovinu a vypočítať uhol α:

  $$\alpha = arctan(\frac{z}{x}) \tag{4.1}$$

  To bude uhol, o ktorý sa natočí celé okolie. Tento postup bude potrebné aplikovať na všetky kľúčové body a ich okolia.

- **Výpočet rovinnosti a cylindricity** - Z vlastných vektorov získaných v predchádzajúcom kroku sa vypočíta rovinnosť (označovaná aj ako planarity) a cylindricita, ktoré budú súčasťou surového deskriptora. Okrem toho, hodnota rovinnosti bude použitá aj pri redukcii kľúčových bodov. Vzťahy pre výpočet rovinnosti a cylindricity (p, c) sú nasledujúce (λ_i sú príslušné vlastné hodnoty a Σ_i je súčet hodnôt vlastných vektorov):

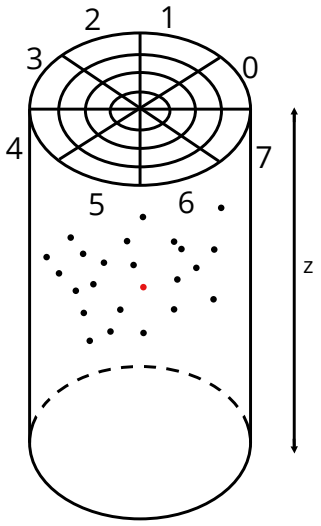
  $$p = \frac{2 * (\lambda_2 - \lambda_1)}{\Sigma_i(\lambda_i)} \tag{4.2}$$

  $$c = \frac{(\lambda_3 - \lambda_2)}{\Sigma_i(\lambda_i)} \tag{4.3}$$

- **Redukcia kľúčových bodov na základe rovinnosti** - Ide o ďalšie filtrovanie kľúčových bodov, tentokrát na základe rovinnosti. Na základe článku [6] budú zahodené kľúčové body s rovinnosťou p väčšou ako 0.9, pretože okolie by malo malú deskriptívnu hodnotu. Po aplikácii tohto kroku dostaneme finálne kľúčové body.

4.3 Surový deskriptor

Ďalšou časťou je zakódovanie okolia každého kľúčového bodu do vektora = deskriptora, čím sa dostávame k extrakcii príznakov. Preto je potrebné rozdeliť priestor okolo kľúčového bodu pomocou tzv. polar gridu. Polar grid je mriežka vytvorená v okolí kľúčového bodu. Je rozdelená na 8 azimutálnych a 4 lineárne rozmiestnené radiálne časti, ako je to možné vidieť na obrázku 4.2. Dokopy na 32 častí (binov). Červený bod ilustruje kľúčový bod a okolo neho sú ostatné body - reálne pôjde o veľké množstvo bodov. Po úspešnom rozdelení priestoru budú všetky body zaradené do jednotlivých binov a v každom bine sa vypočíta



Obr. 4.2: Ilustrácia rozdelenia priestoru okolo kľúčového bodu polar gridom.

priemer a odchýlka výšok a zahrnú sa do deskriptora. Získa sa takto 64 dimenzií deskriptora, ku ktorým sa pridajú vypočítané hodnoty rovinnosti a cylindricity. Výsledný surový deskriptor bude obsahovať 66 dimenzií. Je vhodné počítať aj s prípadom, ak by niektorý bin nemal dostatočnú populáciu. V takom prípade, budú hodnoty prekopírované z nasledujúceho validného binu v smere kľúčového bodu. Následne bude vhodné zo spočítaného surového deskriptora získať kompaktný a spoľahlivý deskriptor, ktorý bude porovnateľný euklidovskou vzdialenosťou. To sa bude vykonávať normalizovaním a redukciou surového deskriptora do vektora príznakov s menším počtom dimenzií na základe článku [5].

4.4 Normalizácia surového deskriptora a redukcia jeho dimenzií

Pre normalizáciu a následnú redukciu surového deskriptora je vhodné vykonať niekoľko krokov. Keďže jednotlivé dimenzie deskriptora nemusia mať Gaussovské rozloženie, je vhodné ich do takého rozloženia namapovať. Potom je vhodné výsledné vektory deskriptorov lineárne transformovať aby sa zväčšili vzdialenosti medzi zhodujúcimi (match) a nezhodujúcimi (non-match) sa pármi deskriptorov. Následne sa vypočíta L_2 vzdialenosť medzi týmito vektormi k získaniu likelihood ratio test (LRT) štatistiky. Nakoniec sa použije redukcia dimenzií, aby boli odstránené prvky s nízkym pomerom signál-šum (signal-to-noise) a opäť výsledky vizualizujeme pomocou LRT štatistiky.

4.4.1 Mapovanie do normálneho rozloženia

Lineárna transformácia, ktorá bude použitá na mapovanie L_2 vzdialeností pre optimálny likelihood ratio test, predpokladá gaussovské rozloženie. Surové deskriptory, ktoré však aktuálne máme, nemusia mať Gaussovské rozloženie. Preto je na základe [5], vhodné jednotlivé dimenzie deskriptora transformovať tak, aby mali Gaussovské rozloženie. Takže ak bude X_d reprezentovať d-tú dimenziu deskriptora, môžme transformovať X_d do $N(0,1)$ distribučnej funkcie náhodnej veličiny Z_d:

$$Z_d = \Phi^{-1}(F(X_d)), \tag{4.4}$$

kde Φ^{-1} je inverzná kumulovaná distribučná funkcia - **cdf** (cumulative distribution function) a $F(X_d)$ je cdf X_d. Cdf je funkcia $F : \mathbb{R} \to [0, 1]$ definovaná predpisom:

$$F(x) = P(X \leq x), \tag{4.5}$$

kde X je náhodná premenná z určitého rozdelenia a x je ľubovoľné reálne číslo.

V praxi je získaná transformačná funkcia nafitovaním polynómu (polynomial fitting) do:

$$\Phi^{-1}(\hat{F}_n(x_i)) \tag{4.6}$$

$\hat{F}_n(x_i)$ je empirická distribučná funkcia - **edf** (empirical distribution function) aplikovaná na súbor dát pre jednotlivé dimenzie deskriptora x_i.
Formálna definícia edf je následovná: Nech $\xi = [x_1...x_n]$, je vzorka o veľkosti n, kde $x_1, ...x_n$ je n pozorovaní zo vzorky. Empirická distribučná funkcia vzorky ξ je potom funkcia $\hat{F}_n : \mathbb{R} \to [0, 1]$ definovaná ako:

$$\hat{F}_n(x_i) = \frac{1}{n} \sum_{i=1}^{n} 1_{\{x_i \leq x\}}, \tag{4.7}$$

kde $1_{\{x_i \leq x\}}$ je indikátor funkcie, ktorý je rovný 1 ak je $x_i \leq x$ a 0 v ostatných prípadoch.

4.4.2 Likelihood ratio test

V tejto časti je vypočítaná vzdialenosť L_2 medzi pármi surových deskriptorov x_a a x_b ako $||Ax_a - Ax_b||_{L_2}$ (pre zhodujúce aj pre nezhodujúce sa páry), pre určenie likelihood ratio test, čo je štatistika, ktorá ukazuje ako dobre je natrénovaný systém.

Zhodujúce a nezhodujúce sa páry deskriptorov sa získajú následovne:

- **Zhodujúce sa deskriptory (matched descriptors)** - Tieto páry zhodujúcich sa deskriptorov M budú získané výberom dvojíc kľúčových bodov z okolia na základe vhodnej okolitej vzdialenosti. Vzdialenosť v ktorej budú páry hľadané sa môže líšiť v závislosti od scény.
- **Nezhodujúce sa deskriptory (unmatched descriptors)** - Nezhodujúce sa páry U budú vytvorené premiešaním hodnôt na pozícii zhôd v zhodujúcich sa párach M medzi sebou.

K tomu aby bolo možné L_2 vzdialenosť medzi deskriptormi vypočítať je vhodné vypočítať maticu A, ktorá bude použitá k premapovaniu surových deskriptorov. Matica A sa získa ako:

$$A^T A = \Sigma_M^{-1} - \Sigma_U^{-1}, \tag{4.8}$$

kde Σ_M je kovariancia diferencií prvkov v sade zhôd $\{x_a - x_b | x_a, x_b \in M\}$. Podobným spôsobom sa spočíta aj Σ_U pre sadu nezhodujúcich sa prvkov. Prakticky bude matica A získaná aplikovaním cholevského dekompozície na $\Sigma_M^{-1} - \Sigma_U^{-1}$. Aby bolo možné aplikovať cholevského dekompozíciu na maticu, musí byť pozitívne semi-defintná. Stav keď nie je matica pozitívne semi-definitná ide vyriešiť aplikáciou eigen dekompozície, ktorá rozloží napríklad maticu M na:

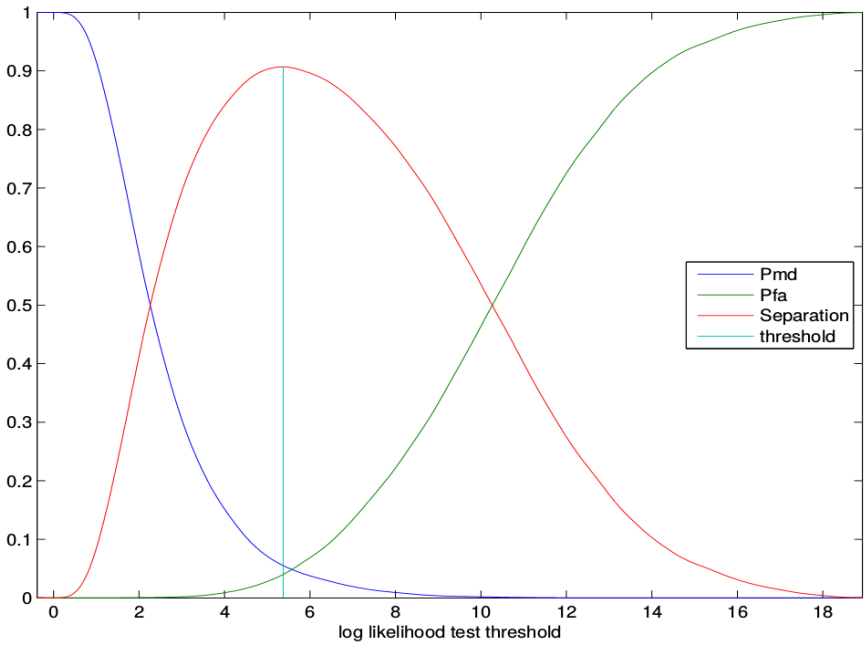
$$M = Q \Lambda Q^{-1} \tag{4.9}$$

Q je štvorcová matica $(N * N)$, ktorej i-ty stĺpec je vlastný vektor q_i matice M a Λ je diagonálna matica, ktorej prvky na diagonále prislúchajú vlastným číslam.
Aby sa získala pozitívne semidefinitná matica M je potrebné nahradiť záporné vlastné čísla na diagonále matice Λ nulami alebo veľmi malými číslami. Takouto upravenou maticou Λ_U sa nahradí pôvodná matica Λ a vykoná sa súčin matíc $Q\Lambda_U Q^{-1}$, ktorého výsledkom bude pozitívne semi-defintná matica.

K určeniu pravdepodobnosti či jednotlivé dvojice deskriptorov sú správne spojené, bude vhodné vypočítať separáciu S medzi zhodami a nezhodami deskriptorov podľa vzťahu:

$$S(t) = (1 - P_{MD}(t))(1 - P_{FA}(t)), \tag{4.10}$$

kde $P_{MD}(t)$ je pravdepodobnosť missed detections (pmd - match medzi deskriptormi mal byť nájdený ale nebol). Táto pravdepodobnosť je určená ako počet zhodujúcich sa párov, ktorých L_2 vzdialenosť je väčšia ako prah (threshold) t, predelená celkovým počtom zhodujúcich sa párov. $P_{FA}(t)$ je pravdepodobnosť false alarms (pfa - match medzi deskriptormi nemal byť nájdený ale bol), ktorá je získaná ako počet nezhodujúcich sa párov, ktorých L_2 vzdialenosť je menšia ako threshold t, predelená celkovým počtom nezhodujúcich sa párov. Takto určená separácia umožní porovnať kvalitu získaných párov. Tam kde je maximálna je najlepší pomer medzi missed detections a false alarms, takže je najmenší počet missed detections a false alarmov.
Príklad vyhodnotenia separácie na základe spomínaných pravdepodobností je na obrázku 4.3, kde je hodnota separácie zhruba 0.9. To znamená, že pravdepodobnosť toho, že nedôjde ani k "prehliadnutiu" matchov a ani nesprávny match nebude falošne klasifikovaný za správny je zhruba 0.9. Ide o veľmi dobrý výsledok, ktorý bol dosiahnutý v práci [5].



Obr. 4.3: Na tomto grafe sa nachádza príklad vyhodnotenia prahu na základe separácie s využitím pravdepodobností missed detections a false alarm [5].

4.4.3 Redukcia dimenzií

Ďalším krokom by mohla byť redukcia dimenzií, ktorou by sa mala zlepšiť separácia. K tomu je potrebné odstrániť tie dimenzie, ktoré majú najnižší signal-to-noise pomer (SNR). Dimenzie, ktoré majú najlepší SNR budú určené na základe vlastných vektorov s najvyššou hodnotou, získaných z transformovaných diferencií nezhodujúcich sa párov deskriptorov.

$$A_k = V_k^T A, \tag{4.11}$$

kde V_k je k najväčších vlastných vektorov získaných z $A^T \Sigma_U A^T$. Následne bude určené koľko dimenzií sa zachová (to bude odpovedať hodnote k). Potom budu vypočítané L_2 vzdialenosti ako $||A_k x_a - A_k x_b||_{L_2}$

Takto získaný spoľahlivý deskriptor obsahujúci k dimenzií bude potrebný pre porovnávanie deskriptorov v reálnej lokalizácii pomocou nejakej jednoduchej metriky, napríklad Euklidovskej vzdialenosti. Keby sa porovnávalo priamo 66 dimenzií Euklidovsky, bez predchádzajúcej normalizácie, nebolo by to spoľahlivé. Spoľahlivé deskriptory sú potrebné preto, aby sme vypustili šum a nevypovedajúce dimenzie a mohli tak vykonať analýzu toho, čo je rozhodujúce pre určenie matchov respektíve non-matchov.

4.5 Porovnávanie keypointov

V skutočnosti budeme mať viacero mračien bodov, ktoré je vhodné spojiť do jedného veľkého mračna, prípadne spojiť niekoľko predošlých a niekoľko nasledujúcich mračien, aby sa získalo dostatočne husté mračno bodov. Následne sa budú kontrolovať kľúčové body v aktuálnom mračne bodov voči kľúčovým bodom v databáze.

V prípade, ak sa nájde dostatočný počet zhôd, tak pred prehlásením, že ide o rovnaké miesto, bude takto získané zhody vhodné analyzovať. Prehlásiť, že ide o rovnaké miesto nie je možné ihneď, pretože veľké množstvo zhôd by mohli byť falošné alarmy a tým pádom by výsledky nemuseli byť vôbec správne. Preto bude v takom prípade vhodné vykonať filtračný krok, kde by sa na základe overenia zhôd medzi deskriptormi dospelo k menšiemu počtu falošných alarmov. Zhody nájdené po kroku filtrácie, bude možné prehlásiť za rovnaké miesta.

4.6 Filtrácia a nájdenie zhodných miest

Pôjde o filtráciu na úrovni matchov deskriptorov. Môže byť založená na usporiadaní všetkých zhôd a následnom hľadaní maximálnych hodnôt, čím by sa malo dospieť k malému počtu falošných alarmov.

Filtrácia je založená na:

- **Rozdelení matchov medzi jednotlivými mračnami bodov** - Podľa zhôd deskriptorov sa rozdelia matche s príslušnými mračnami bodov (napríklad pointcloud č. 0 a k nemu všetky prislúchajúce matche), takže vzniknú dvojice mračien bodov napríklad `(0 500, 0 501, 0 510, 0 500, 0 501 ...)`. To sa vykoná pre všetky matche medzi mračnami bodov.

- **Určení zhôd medzi mračnami bodov** - Všetky matche sa prejdú a určia sa počty zhôd medzi rovnakými mračnami bodov.

- **Zoradení a redukcii zhôd** - Zhody sa zoradia podľa maximálnych hodnôt a množnia sa zredukuje o tie zhody, ktoré nemajú dostatočný počet resp. dostatočné zastúpenie. To sa bude diať na základe filtračnej hranice, ktorá určuje minimálne zastúpenie jednotlivých zhôd. Najvyššia hodnota predstavuje najviac zhôd medzi deskriptormi dvoch miest a tým pádom vysokú pravdepodobnosť, že sa bude jednať o rovnaké miesto.

- **Odstránení kolízie medzi mračnami bodov** - V prípade ak nastane situácia, že jeden pointcloud má dostatočný počet zhôd s viacerými inými pointcloudami, prebehne porovnanie medzi týmito N dvojicami a podľa väčšieho počtu zhôd sa vyhodnotí najlepšia dvojica. Získavame tak finálne dvojice.

- **Prehlásení daného miesta, za už navštívené** - Keďže indexy jednotlivých mračien bodov môžu symbolizovať aj pohyb v čase, môžeme prehlásiť, že jednotlivé finálne dvojice určujú miesta, na ktorých sa robot už nachádzal, takže sa dostal na rovnaké miesto.

Kapitola 5

Implementácia

Táto kapitola podrobnejšie rozoberá praktickú časť práce. Popisuje výber nástrojov na implementáciu, použité knižnice, niektoré implementované funkcie a riešené problémy.

Pri implementácii systému k lokalizácii mobilného robota bol použitý programovací jazyk `C++` a tiež programovacie prostredie `Matlabu`. Jazyk `C++` je použitý hlavne pre prácu s mračnami bodov k čomu využíva knižnice `Eigen, PCL, but_velodyne_lib`, ktorých funkcia a využitie sú popísané tiež v tejto kapitole. `Matlab` je využitý pre implementáciu skriptov k zložitejším matematickým operáciám ako sú kovariancia, cdf, edf, cholevského dekompozícia a podobne. `C++` bol zvolený jednak pre väčšie skúsenosti s týmto jazykom, a taktiež pre lepšiu použiteľnosť s knižnicami but_velodyne_lib či PCL, ktoré sú v ňom napísané. Operačný systém, na ktorom bola vyvíjaná táto konzolová aplikácia je `Ubuntu 16.04 LTS`.

5.1 Načítanie vstupných dát

V tejto práci sú využité vstupné dáta získané z Velodyne LiDAR senzoru, ktoré poskytuje `KITTI` benchmark[1] odometrie. Ide o známu, obsiahlú dátovú sadu, ktorá sa využíva pri práci s 3D laserovými dátami, čo je aj dôvodom jej zvolenia v tejto práci.
Tento benchmark obsahuje 22 sekvencií. Ďalej obsahuje dáta z laserov, kalibrácie, pozície a iné. Dostupné sú dáta zo senzorov pre snímanie 2D obrazov, ale aj z laserového senzoru pre 3D dáta. Pre 3D dáta sa využíva laserový senzor Velodyne HDL-64E, ktorý bol predstavený v prechádzajúcich kapitolách. Detailnejšie informácie sú dostupné v technickej špecifikácii [12].

Knižnica `but_velodyne_lib`[2]

Jedná sa o voľne dostupnú knižnicu pre prácu s 3D laserovými dátami zo senzorov Velodyne. Vyvíja sa na fakulte informatiky Vysokého učení technického v Brne skupinou `Robo@FIT`. Pre správnu funkciu knižnice je potrebné mať nainštalované knižnice:

- **Eigen3** - Knižnica pre lineárnu algebru - matice, vektory, numerické riešenia a súvisiace algoritmy. Využíva sa aj v implementovanom systéme pri práci s maticami a vektormi.

[1] KITTI dáta su dostupné na http://www.cvlibs.net/datasets/kitti/eval_odometry.php

[2] Knižnica but_velodyne_lib je voľne dostupná pod GNU licenciou na github https://github.com/robofit/but_velodyne_lib

- **OpenCV verzie 2.4.9** - Knižnica pre manipuláciu s obrazom.
- **PCL 1.79** - Knižnica pre spracovanie 2D/3D obrazov a mračien bodov. Využíva sa aj v implementovanom systéme pri práci s mračnami bodov.

Systém, ktorý bol implementovaný v tejto práci, používa `but_velodyne_lib`, konkrétne funkcie z `VelodynePointCloud` na načítanie vstupných mračien bodov. Nad týmito mračnami sa následne vykonávajú všetky ďalšie operácie.

5.2 Implementácia podvzorkovania voxelovou mriežkou

Prvým krokom - predspracovaním v návrhu bolo podvzorkovanie vstupných dát s využitím voxelovej mriežky. Prakticky k implementácii bola využitá už spomínaná knižnica `PCL`, ktorá má pre takéto podvzorkovanie príslušné funkcie. Pre aplikáciu voxelovej mriežky bolo potrebné zadefinovať filter s určitou veľkosťou, ktorá je v implementovanom systéme zvolená na 0.4 m, ako je uvedené v práci [6].

5.3 Získanie kľúčových bodov

Kľúčové body sa získavajú postupom aký bol popísaný v kapitole 4.2. Implementačne zaujímavý je konkrétne krok určenia orientácie kľúčových bodov, kde je potrebné vypočítať kovariančnú maticu z okolia daného kľúčového bodu. K získaniu okolia sa v implementácii systému využíva KdTree algoritmus pre vyhľadávanie všetkých susedných bodov v zadanom okolí, prípadne algoritmus pre nájdenie `K` najbližších susedov od zadaného bodu z knižnice `PCL`.

Prvým krokom je vytvorenie kdTree objektu a nastavenie vstupného mračna bodov, v ktorom sa bude vyhľadávať. Následne je vytvorený bod (`searchPoint`), okolo ktorého sa bude vyhľadávať a 2 vektory (`radiusSearchPoint, radiusSearchPointDistance`) pre udržiavanie informácii o susedných bodoch. Potrebné je tiež zadefinovať okolie (`radius`), v ktorom sa bude vyhľadávať.

Ďalším krokom je spočítanie kovariancie z okolia bodu `searchPoint`. K tomu bola využitá `PCL` funkcia `computeCovarianceMatrix`, ktorej vstupom je získané okolie a výstupom je kovariančná matica o veľkosti 3×3.

Pomocou `SelfAdjointEigenSolver` z knižnice `Eigen`, ktorej vstupom je kovariančná matica, sa spočítajú vlastné vektory a vlastné čísla. S ich využitím sa určí natočenie okolia ako to bolo popísané v kapitole 4.2. Okolie sa rotuje pomocou `Eigen` funkcie `rotate` a následná transformácia prebieha použitím `PCL` funkcie `transformPointCloud`.

5.4 Rozdelenie priestoru k získaniu surového deskriptora

Zaujímavou časťou z pohľadu implementácie bolo aj rozdelenie priestoru okolo kľúčového bodu pomocou mriežky polar grid. Návrh, ktorého sa drží aj implementácia bol popísaný v kapitole 4.3. Rozdelenie okolia do jednotlivých binov prebieha v implementovanej funkcii `getFinalIndexes`, kde sa využíva opäť KdTree algoritmus pre vyhľadávanie susedných bodov v zadanom okolí, kde rádius prezentuje okolie, v ktorom sa tvorí cylinder.

- **getPolarBinIndex** - Táto funkcia slúži na priradenie každého bodu okolia do jednej z 8 azimutálnych častí. Pre každý bod sa spočíta jeho uhol α podľa vzťahu 4.1. Keďže

ide o cylinder rozdelený na 8 častí, každá časť má 45 stupňov a index príslušného bodu je určený ako:

$$a_i = \frac{\alpha}{45} \tag{5.1}$$

- **getPointsDistance** - Táto funkcia slúži na priradenie každého bodu do jednej zo 4 lineárne rozmiestnených radiálnych častí, podľa vzdialenosti od kľúčového bodu. Pre každý bod okolia sa spočíta vzdialenosť od kľúčového bodu ako:

$$d = \sqrt{(\Delta x)^2 + (\Delta z)^2}, \tag{5.2}$$

kde x, z sú súradnice bodu. Získanie indexu radiálnej časti, do ktorej bod patrí, je implementované ako:

$$d_i = \frac{d}{\frac{radius}{4}}, \tag{5.3}$$

kde *radius* je okolie v ktorom sa tvorí cylinder.

- **getFinalMeansAndVariances** - V tejto funkcii sa počíta priemer a odchýlka pre každý bin v polar gride. Okrem toho je v tejto funkcii ošetrený prípad, ak danému binu neprislúcha žiaden bod, kedy sú skopírované hodnoty z najbližšieho binu.

Po vykonaní vyššie uvedených funkcií sú dostupné všetky potrebné dáta pre surový deskriptor o veľkosti 66 dimenzií. To znamená 64 dimenzií deskriptora je získaných vypočítaním priemeru a odchýlky pre každý bin polar gridu, plus hodnota rovinnosti a cylindricity, ktoré boli vypočítane podľa vzorcov v kapitole 4.2.

5.5 Normalizácia deskriptora s využitím Matlabu

Všetky doteraz popísané kroky prebiehali v jazyku `C++` hlavne z dôvodu veľmi dobrej knižnice pre prácu s mračnami bodov. Pre zložitejšie matematické operácie bolo vhodnejšie použiť programovacie prostredie `Matlab`.
Boli implementované 3 skripty pre normalizáciu surového deskriptora: `genMappingFce.m`, `mappingDescriptors.m` a `matrixFactor.m`.

Generovanie mapovacích funkcií

K vygenerovaniu funkcií, ktorými sa premapujú surové deskriptory tak, aby jednotlivé dimenzie deskriptorov mali Gaussovské rozloženie, bol vytvorený skript `genMappingFce.m`. Postup ako mapovanie prebieha a aké vzorce sú k tomu použité sa nachádza v kapitole 4.4.1. Keďže bol k implementácii použitý `Matlab`, všetky tieto funkcie sú už implementované.

Použité funkcie:

- **ecdf** - Ide o empirickú distribučnú funkciu, ktorá bola aplikovaná na dáta príslušných dimenzií deskriptora.
- **norminv** - Funkcia, ktorá vykonáva inverznú kumulovanú distribúciu. Jej vstupom je výstup z funkcie `ecdf`.
- **polyfit** - Posledným krokom v tomto skripte je nafitovanie polynómu štvrtého stupňa na výstup z funkcie `norminv`. Takto získané polynómy sú uložené do súboru, aby s nimi bolo možné ďalej pracovať.

Premapovanie deskriptora

Skript `mappingDescriptors.m`, ktorý prechádza všetky dimenzie každého deskriptora a prenásobuje ich príslušnými mapovacími funkciami, ktoré boli vypočítané v predchádzajúcom kroku.

Matched a unmatched deskriptory

Po ukončení skriptu premapovania pokračuje implementácia systému v `C++`, kde je implementovaný matching deskriptorov s využitím `KdTree` vyhľadávania `nearestKSearch` pre nájdenie najbližších K susedov od zadaného bodu. Vhodná vzdialenosť okolia sa môže líšiť od scény.

Pre testované scény bola zvolená vzdialenosť $0.5m$, čo vyplynulo z testovania, kde boli rôzne sekvencie spustené s rôznymi vzdialenosťami v ktorých sa vyhľadávali najbližšie susedné body. Výsledky testov je možné vidieť v kapitole 6.1.

V tomto kroku bolo potrebné zaistiť aj to, aby matche nevznikali s deskriptormi v mračnách bodov, ktoré bezprostredne nasledovali, teda aby vznikali až po prejdení určitej vzdialenosti. Bolo to prakticky implementované tak, že určitý počet následujúcich mračien bodov nebol braný v úvahu.

Unmatched deskriptory boli implementované s využitím funkcie `random_shuffle` z `C++` knižnice `algorithm`.

Určenie vzdialeností medzi deskriptormi a redukcia dimenzií

Po získaní sady matched a unmatched deskriptorov je potrebné spočítať maticu A resp. A_k k určeniu L_2 vzdialeností medzi pármi deskriptorov, ako je popísané v kapitole 4.4.2. Pre tento účel bol implementovaný skript `matrixFactor.m` v `Matlabe`, ktorý tieto matice spočíta a vypočíta aj príslušné L_2 vzdialenosti. Skript tiež vykonáva redukciu dimenzií na základe vlastných vektorov s najvyššou hodnotou, získaných z transformovaných diferencií nezhodujúcich sa párov deskriptorov, ako bolo uvedené v návrhu.

Použité funkcie:

- **cov** - Funkcia použitá k výpočtu kovariancie diferencií prvkov v sade zhôd a tiež v sade nezhodujúcich sa prvkov.
- **cholcov** - Táto funkcia je použitá k zisteniu, či je matica pozitívne semi-definitná. V prípade ak nie, je taká matica upravená tak, aby bola pozitívne semi-definitná.
- **eig** - Slúži k získaniu vlastných čísel a vlastných vektorov. Táto funkcia bola použitá pri úprave matice na pozitívne semi-definitnú.
- **chol** - Vykonanie Cholevského dekompozície.
- **eigs** - Funkcia, ktorá vráti k najväčších vlastných čísel, na základe ktorých sa redukujú dimenzie.

Vizualizácia dát

Okrem spomínaných skriptov, bol vytvorený aj pomocný program v `C++ - visualizer`, ktorý slúžil pri implementácii systému na vykresľovanie medzivýsledkov ako napríklad vy-

kreslenie mračien bodov pre jednotlivé úlohy, vykreslenie nájdených dvojíc bodov a podobne. Tento program využíva vizualizér z knižnice `but_velodyne_lib`. Ukážka možného pomocného výstupu z `visualizer.cpp` je na obrázku 5.1.

Obr. 5.1: Príklad výstupu z programu `visualizer` - mračno kľúčových bodov.

5.6 Filtrácia a nájdenie zhodných miest

Výpočet hodnôt pmd, pfa a separácie sú implementované v `C++`, keďže nejde o zložité funkcie. Výsledné hodnoty sú zapísané do súborov, ktoré využíva `plot_graph.m`, skript vytvorený v `Matlabe` na vykreslenie grafu ich závislosti. Tam kde je separácia najvyššia, je najlepší pomer medzi **pmd** a **pfa**, takže je najmenší počet missed detections a false alarmov.

V prípade ak výsledky neboli dostačujúce, bolo vhodné implementovať krok filtrácie, ktorým by sa výsledky zlepšili, čiže znížil počet falošných alarmov. Takýto krok, spolu s hľadaním už navštívených miest, je implementovaný v `Matlabe` ako `results.m` a je založený na popise v kapitole 4.6

Zobrazenie výsledných zhôd

K zobrazeniu výsledných dvojíc bol v `Matlabe` vytvorený skript `plot_places.m`, ktorý slúži k vykresleniu prejdenej trajektórie na ktorej vyznačí jednotlivé dvojice zhôd získaných po filtračnom kroku.

5.7 Riešené problémy

Počas implementácie sa vyskytli rôzne problémy, ktoré bolo potrebné vyriešiť. Medzi nich patrí napríklad skutočnosť, že pôvodne mala byť implementácia systému získaná od autorov článku [6] a nad ich systémom mali byť vykonané rôzne testy a experimenty.

Implementácia systému a jeho možné odchýlky

Keďže sa implementáciu od autorov získať nepodarilo, hlavným cieľom sa stala implementácia systému pre lokalizáciu mobilného robota, kde bolo cieľom nájsť už navštívené miesta robotom s využitím článkov [6, 5]. Problémom však bolo, že v týchto článkoch nie sú popísané všetky detaily pre implementáciu, a tak neboli dostupné všetky hodnoty potrebných premenných. Medzi nich patrí napríklad veľkosť okolia, či už pre tvorbu cylindrov, alebo pre vyhľadávanie matchov, ako je riešené aby sa netvorili matche s bezprostredne nasledujúcimi mračnami, aký stupeň polynómov zvoliť pri mapovaní do normálneho rozloženia a podobne. Všetky takéto premenné bolo potrebné otestovať a na základe výsledkov a pozorovaní určiť vhodné premenné. Týmto mohlo dôjsť k rôznym odchýlkam a nepresnostiam pri implementácii.

Práca s veľkým množstvom dát

Pôvodne načítanie dát prebiehalo z jedného súboru so všetkými dátami pre jednotlivé úlohy a rovnakým spôsobom prebiehal zápis výstupných dát. To fungovalo bez problému pre nie až taký veľký objem dát, ktorý sa pohyboval v stovkách tisícov riadkov. Ak bol objem dát v jednom súbore príliš veľký (niekoľko miliónov riadkov), `Matlab` sa s tým nedokázal vysporiadať, a teda bolo potrebné nájsť riešenie tohto problému. Osvedčilo rozdeliť dáta na niekoľko nie až tak veľkých častí, ktoré obsahovali tisíce riadkov a načítať ich sekvenčne za sebou. To isté bolo aplikované na výstupné dáta pri ich zápise. Výsledkom je, že načítanie a zápis veľkého množstva dát prebieha spoľahlivo, aj keď za cenu dlhšieho trvania týchto operácií.

Kapitola 6

Systém a jeho výsledky

Táto kapitola uvádza výsledky a vyhodnotenie práce. Sú tu tiež popísané možné parametre systému pre lokalizáciu mobilného robota a nachádza sa tu aj prehľad a zhodnotenie výsledkov dosiahnutých bez premapovania deskriptorov, s premapovaním deskriptorov a s redukciou dimenzií deskriptorov.
Okrem iného sú v tejto kapitole popísané a zhodnotené výsledky pred a po kroku filtrácie, kde je možné vidieť robotom už navštívené miesta na rôznych sekvenciách.
Ďalej je zhodnotený v kroku filtrácie vplyv premennej, ktorá určuje hranicu pri rozhodovaní či ide o správny spoj. Táto premenná môže byť nazvaná prahom pri filtrácii. Pri testovaní boli využité 3D laserové dáta z `KITTI` benchmarku.
V závere tejto kapitoly sa nachádza celkové zhodnotenie systému a jeho budúce možné rozšírenia.

Parametre systému

Systém je možné spustiť s rôznymi parametrami, ktorých prehľad s vysvetlením sa nachádza v tabuľke 6.1:

Parameter	**Vysvetlenie**	**Povinný**
-d "cesta"	parameter s cestou k laserovým KITTI dátam	áno
-p "cesta"	parameter s cestou k súboru s transformačnou maticou pre transformáciu kľúčových bodov v jednotlivých	áno
-m	vykonanie mapovania	nie
-f	vykonanie redukcie dimenzií	nie
-l	vykonanie len posledného kroku systému (len v prípade dostupných všetkých predch. dát)	nie
-params 80 1 0.5 20	parameter pre nastavenie nasledujúcich hodnôt: - počet ignorovaných mračien bodov pri hľadaní zhôd - okolie pre tvorbu cylindrov (m) - vzdialenosť v ktorej sa majú hľadať zhody (m) - max. počet hľadaných susedov pri hľadaní zhôd	nie

Tabuľka 6.1: Prehľad parametrov systému.

Príklad spustenia systému pre sekvenciu `07` s defaultnými hodnotami pre `-params` a s vykonaním krokov normalizácie deskriptora:

```
./system -d "../kitti_data/sequences/07/velodyne/" -p
"../kitti_data/poses/07.txt" -m -f
```

6.1 Vhodné okolie pre susedné body

Tabuľky v tejto sekcii sú výsledkom testovania, kde bolo experimentované s rôznym nastavením okolia v ktorom sa vyhľadávali najbližšie susedné body. Pri zvolení veľmi malej vzdialenosti, ktorá sa pohybovala v okolo $0.1m$ nebolo získané dostatočné množstvo bodov v okolí. Pri zväčšovaní okolia sa množstvo bodov zvyšovalo a pri nastavení okolia väčšieho ako $1m$, bolo bodov v okolí veľké množstvo, čo komplikovalo výpočet (pre stroj na ktorom prebiehali výpočty bola ideálna hodnota nájdených susedov okolo 20000), keďže sa počítalo s veľkým množstvom dát. Tabuľky 6.2, 6.3, 6.4 a 6.5 zobrazujú priemerný počet nájdených najbližších bodov v okolí pre rôzne sekvencie a tiež celkový priemer.

Je vidieť, že pre sekvencie `05` a `06` by bolo vhodnejšie zvoliť nižšiu hodnotu okolia, pretože už aj pri hodnote $0.5m$ je nájdených bodov veľké množstvo.

Sekvencia	Počet nájdených bodov
01	1642
05	3898
06	4192
07	2139
priemerne	2767

Tabuľka 6.2: Okolie $0.1m$.

Sekvencia	Počet nájdených bodov
01	6982
05	12864
06	14360
07	7812
priemerne	10504

Tabuľka 6.3: Okolie $0.2m$.

Sekvencia	Počet nájdených bodov
01	16365
05	31364
06	35360
07	19963
priemerne	25736

Tabuľka 6.4: Okolie $0.5m$.

Sekvencia	Počet párov
01	32563
05	65628
06	70750
07	42712
priemerne	52913

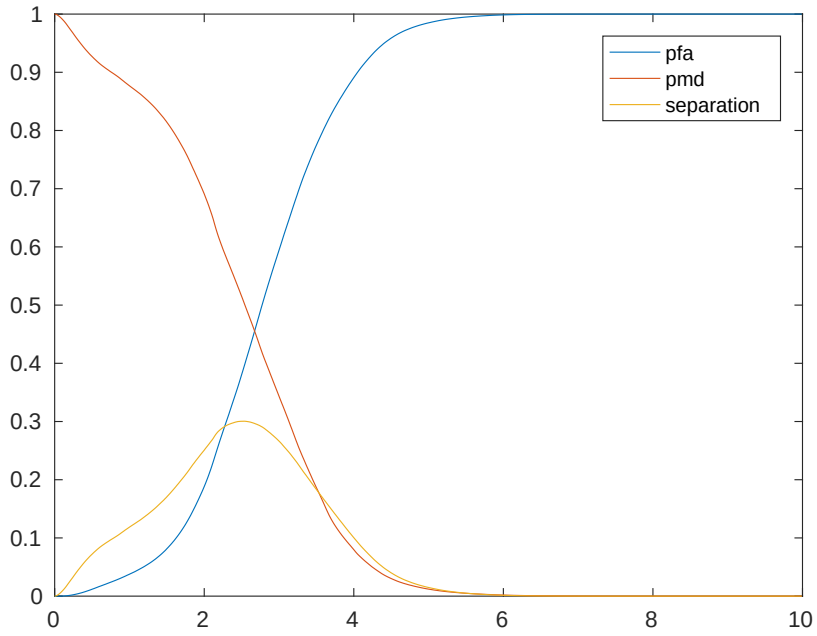
Tabuľka 6.5: Okolie $1m$.

6.2 Výsledky po premapovaní a redukcii dimenzií deskriptora

Táto sekcia zobrazuje a porovnáva výsledky získané spustením vytvoreného systému s dátami z `KITTI` datasetu zo sekvencií `01,05,06,07`. Postupne sa tu nachádzajú grafy závislosti medzi hodnotami pmd a pfa na surových neupravených deskriptoroch, premapovaných deskriptoroch a nakoniec na deskriptoroch s redukovanými dimenziami. Najlepší pomer medzi pmd a pfa hodnotami určuje najvyššia hodnota separácie, kde najvyššia separácia znamená najmenší počet missed detections a false alarmov.

Vyhodnotenie bez premapovania deskriptorov

Na obrázku 6.1 je možné vidieť vyhodnotenie separácie na surových deskriptoroch. Podľa očakávania, krivka separácie nie je príliš vysoko a maximálna hodnota sa nachádza pod hodnotou 0.3. Takže pravdepodobnosť toho, že nedôjde ani k "prehliadnutiu" matchov a ani nesprávny match nebude falošne klasifikovaný za správny je zhruba 0.3. Keďže vyhodnotenie prebiehalo na surových deskriptoroch, ide o očakávanú výslednú hodnotu. Pre zlepšenie výsledkov bolo aplikované premapovanie deskriptora na základe postupu popísaného v kapitole 4.4.



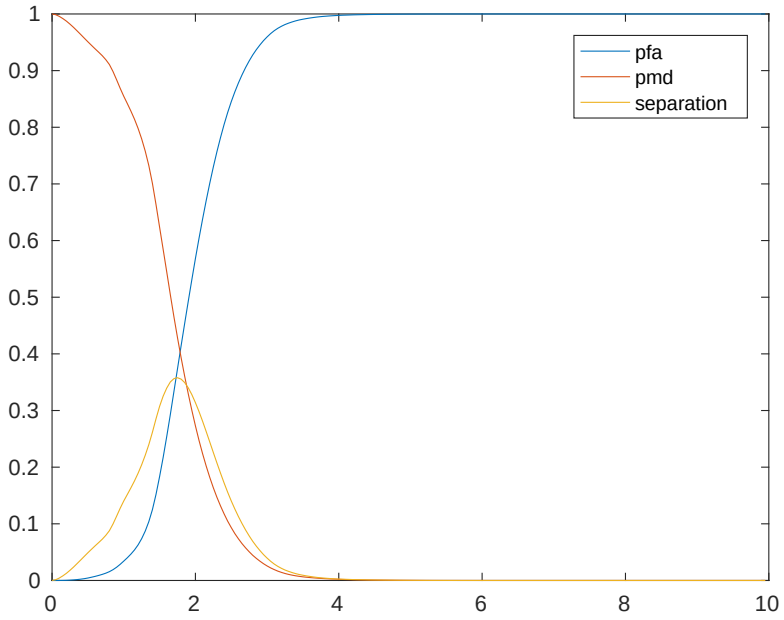
Obr. 6.1: Vyhodnotenie separácie na surových deskriptoroch.

Vyhodnotenie po premapovaní deskriptorov

Po spustení systému s premapovaním deskriptorov boli získané výsledky, ktoré je možné vidieť na obrázku 6.2. Výsledky sú v porovnaní s vyhodnotením bez premapovania lepšie, separácia má maximálnu hodnotu približne 0.4, čo ale nie je najlepší výsledok. Preto bolo vhodné implementovať a použiť ďalšiu optimalizáciu, ktorá by mohla zlepšiť výsledky a tou je redukcia dimenzií.

Vyhodnotenie po redukcii dimenzií deskriptorov

Posledným krokom v optimalizácii separácie bola redukcia dimenzií, ktorá by mala o niečo zlepšiť separáciu. Zredukovanie dimenzií tiež odstráni nevypovedajúce dimenzie, zníži šum a zníži výpočetnú náročnosť, keďže sa bude počítať s menším počtom dimenzií. Na obrázku 6.3 je vyhodnotenie po tomto kroku redukcie dimenzií. Na základe sploštenia grafu je vidieť, že dimenzie boli zredukované no separácia je stále nízka. Keďže sa týmito krokmi nepodarilo získať spoľahlivé výsledky, bola vykonaná filtrácia, na základe ktorej by malo byť odstránené



Obr. 6.2: Vyhodnotenie separácie po premapovaní deskriptorov.

čo najväčšie množstvo falošných alarmov. Výsledky tohto filtračného kroku sú rozobrané v nasledujúcej kapitole.

6.3 Filtračný krok

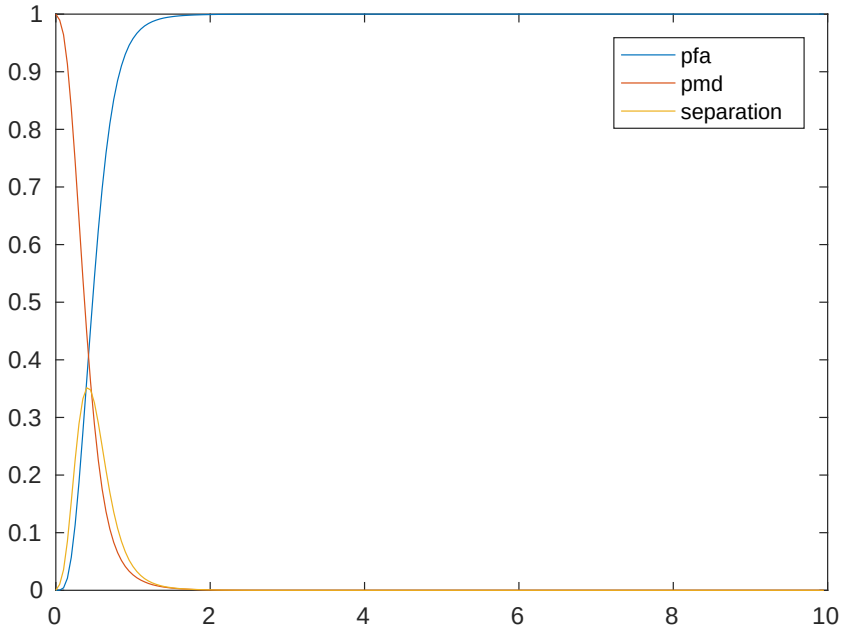
Ide o posledný implementovaný krok v systéme popísaný v kapitole 4.6, ktorý slúži k zlepšeniu výsledkov. Keďže mapovanie a redukcia dimenzií deskriptorov nepriniesli uspokojivé výsledky, bolo vhodné implementovať filtračný krok, ktorého výsledkom by malo byť výrazné zredukovanie falošných alarmov a tým získanie spoľahlivejších výsledkov. Ak bude filtrácia úspešná, budú získané zhodné miesta - dvojice, ktoré reprezentujú robotom už navštívené miesto. Výsledky po filtrácii na rôznych sekvenciách sú uvedené v tejto podkapitole.

Nasledujúce vyhodnotenie porovnáva výsledky pred filtráciou a po filtrácii. Testovanie prebiehalo na viacerých sekvenciách. Výsledné grafy zastupujú skupinu sekvencií, kedy sa robot dostal na rovnaké miesto raz alebo viac krát a kedy sa na rovnaké miesto nedostal vôbec. Pre objektívne porovnanie bola pri všetkých nasledujúcich vyhodnoteniach nastavená hranica pri rozhodovaní či ide o správny spoj na hodnotu `90`.

Vyhodnotenie sekvencie, kedy sa robot dostal na rovnaké miesto

Vhodná sekvencia, kedy sa robot vrátil na už navštívené miesto je sekvencia `06`, ktorá obsahuje celkovo 1100 pozícii robota. Porovnanie pred filtráciou a po filtrácii, ktoré je možné vidieť na obrázku 6.4 naznačuje, že sa výrazne zmenšil počet falošných alarmov. Zhody medzi pozíciami sú znázornené červenými spojmi.

Po dôslednejšom preskúmaní tejto sekvencie ale bolo zistené, že ide o prípad, kedy sa robot vracia po rovnakej trase, ale opačným smerom a tým pádom dovidí na rovnaké pozície



Obr. 6.3: Vyhodnotenie separácie po redukcii dimenzií deskriptorov.

ako dovidel, keď bol v opačnom smere. Krokom filtrácie boli teda odfiltrované aj správne zhody, ktoré bolo možné využiť pri detekcii slučiek. Keďže sa jedná o širokú cestu predelenú trávnatou časťou, tak body v opačnom smere sú ďalej, a teda zhôd medzi deskriptormi je menej. To je dôvod, prečo boli tieto zhody odfiltrované.

Prípad, kde sa robot dostal na rovnaké miesto a nedovidí na opačný smer je napríklad na sekvencii `07`, ktorá obsahuje celkovo 1100 pozícii robota. Výsledky je možné vidieť na obrázku 6.5. Nachádza sa tam viacero falošných spojov, keďže vyhodnotenie prebiehalo s prahom `90`.

Vyhodnotenie sekvencie, kedy sa robot dostal na rovnaké miesto viac krát

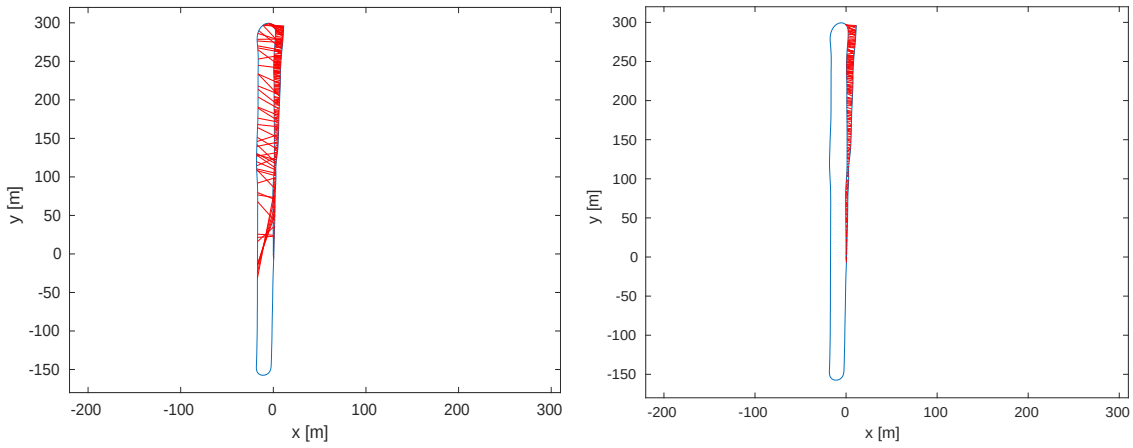
Na obrázku 6.6, je možné vidieť výsledky ktoré boli dosiahnuté, keď robot navštívil rovnaké miesto viac krát. Testovanie prebiehalo na sekvencii `05`, ktorá obsahuje celkovo 2760 pozícii robota. Opäť bol filtračným krokom dosiahnutý výrazne lepší výsledok aj keď už na prvý pohľad je vidieť, že niekoľko false alarmov ostalo aj po tomto kroku.

Vyhodnotenie sekvencie, kedy sa robot nedostal na rovnaké miesto

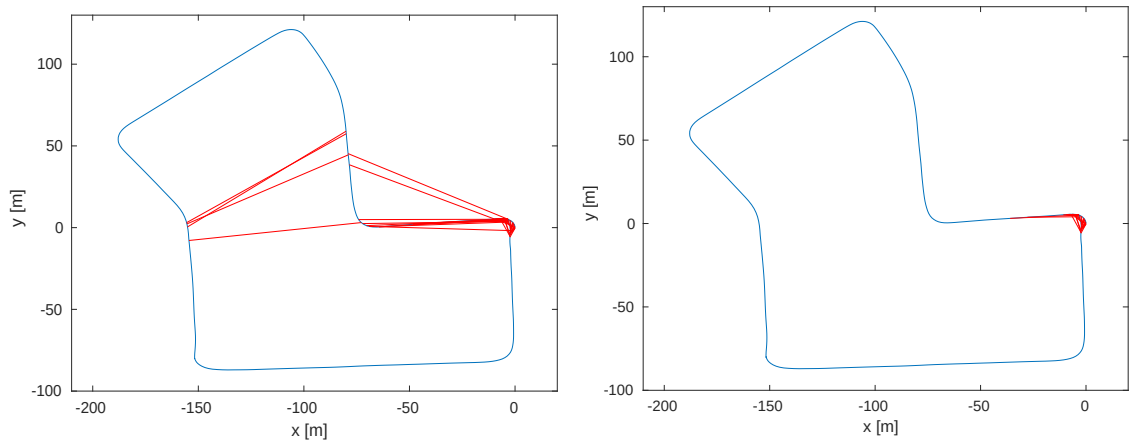
Skupinu vyhodnotení, kedy sa robot nedostal vôbec na rovnaké miesto zastupuje sekvencia `01`, ktorá obsahuje 1100 pozícii robota. Na obrázku je 6.7 vidieť, že po filtrácii boli odstránené všetky falošné alarmy a nenachádzajú sa tam žiadne iné spoje, čo znamená, že sa robot ani raz nedostal na rovnakú pozíciu.

Vyhodnotenie filtrácie

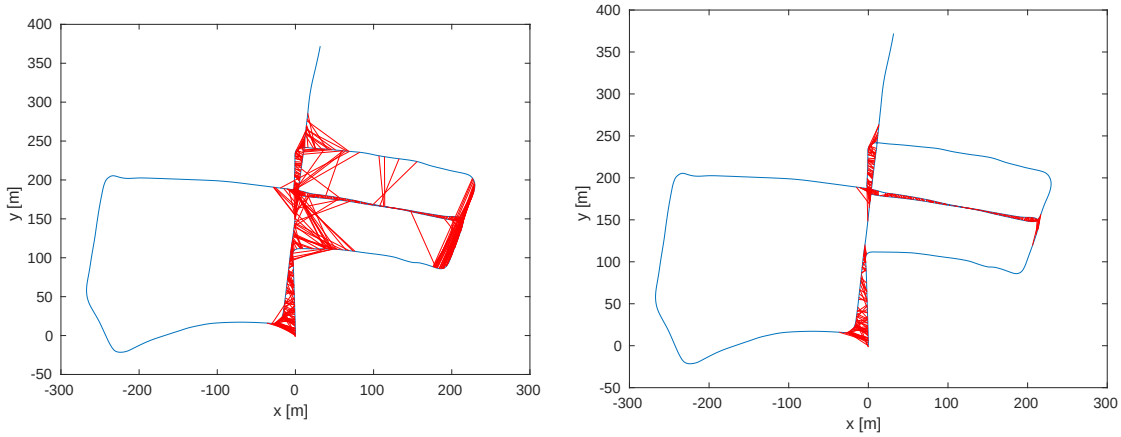
Na obrázkoch 6.8 a 6.9 sa nachádza percentuálne zastúpenie false alarmov a správnych spojov, ktoré boli nájdené, pred filtráciou a po filtrácii pre jednotlivé sekvencie.



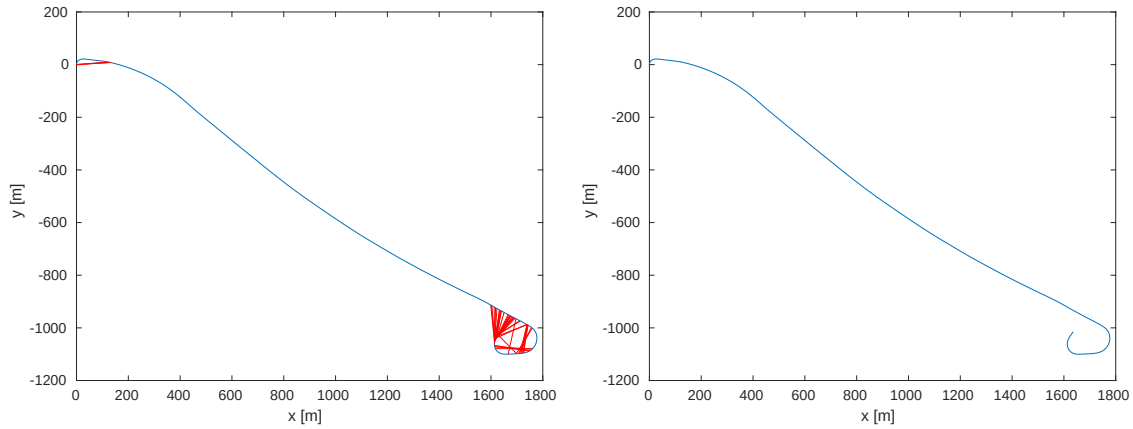
Obr. 6.4: Sekvencia 06: Vľavo výsledok pred filtráciou. Vpravo výsledok po filtrácii. Červené čiary znázorňujú nájdené rovnaké pozície.



Obr. 6.5: Sekvencia 07: Vľavo výsledok pred filtráciou. Vpravo výsledok po filtrácii. Červené čiary znázorňujú nájdené rovnaké pozície.

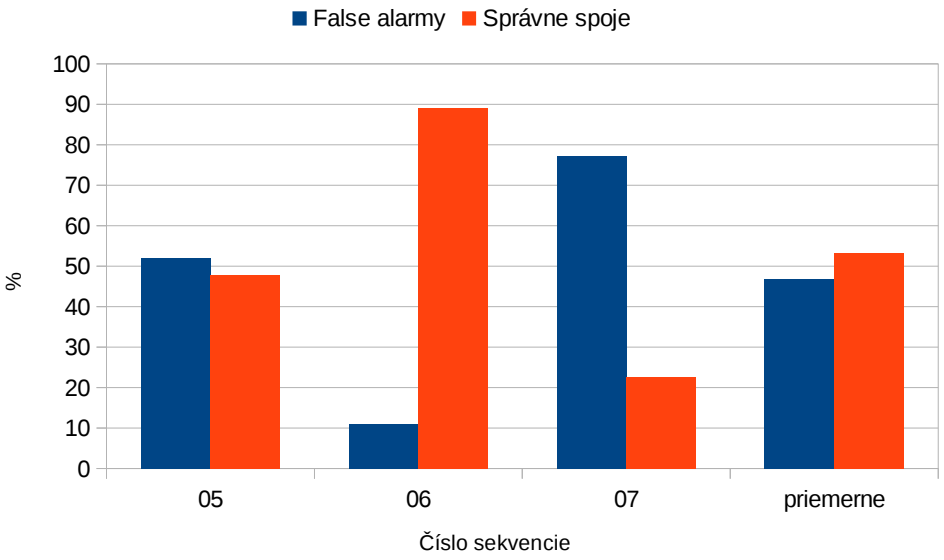


Obr. 6.6: Sekvencia 05: Vľavo výsledok pred filtráciou. Vpravo výsledok po filtrácii. Červené čiary znázorňujú nájdené rovnaké pozície.



Obr. 6.7: Sekvencia 01: Vľavo výsledok pred filtráciou. Vpravo výsledok po filtrácii. Červené čiary znázorňujú nájdené rovnaké pozície.

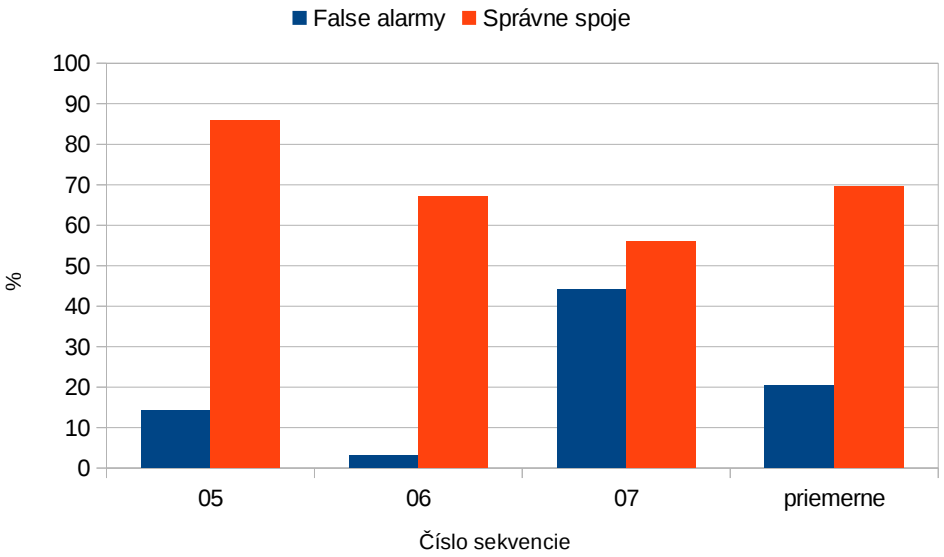
Kým pred filtráciou bolo priemerne 50% nájdených spojov falošnými alarmami a 50% spojov boli správne detekované pozície, tak filtráciou bolo odstránených v priemere približne 65% falošných alarmov. Zlepšili sa tak výsledky nájdených správnych spojov, ktorých je priemerne 70%. Týmito krokmi boli detekované pozície, ktoré už robot navštívil.



Obr. 6.8: Vyhodnotenie nájdených spojov pred filtráciou.

Vplyv filtračnej hranice na nájdené zhody pozícií robota

Filtračná hranica - prah je dôležitý parameter, ktorý tvorí hranicu medzi tým, či je dostatočný počet nájdených zhôd medzi deskriptormi pre dané dvojice pozícií, alebo nie je. Na obrázkoch 6.10 a 6.11 je možné vidieť vplyv zvyšujúceho sa prahu pri filtrácii na sekvencii 07.
Prah s hodnotou 0 znamená, že sú vykreslené všetky nájdené páry medzi pozíciami (false alarmy aj správne páry pozícií). So stúpajúcim prahom klesajú false alarmy, no je potrebné



Obr. 6.9: Vyhodnotenie nájdených spojov po filtrácii.

ustriehnuť, aby prah nebol príliš vysoký a aby neboli odstránené správne páry. Najvhodnejší prah je práve vtedy, ak je nájdených najmenej false alarmov a najviac správnych párov medzi pozíciami.

6.4 Celkové zhodnotenie systému a jeho možné vylepšenia

Výsledkom implementácie je systém, ktorý je schopný lokalizovať pozíciu robota, so zameraním sa na miesta, kde sa už robot predtým nachádzal. Systém bol implementovaný na základe článku [6] a využíva premapovanie deskriptora a redukciu dimenzií na základe článku [5].

Systém po implementácii funkcií pre mapovanie a redukciu deskriptorov neposkytuje najlepšie výsledky, z dôvodu veľkého množstva falošných alarmov. Môže to byť spôsobené rôznymi odchýlkami pri výpočtoch, prípadne nepresne zvolenými parametrami.

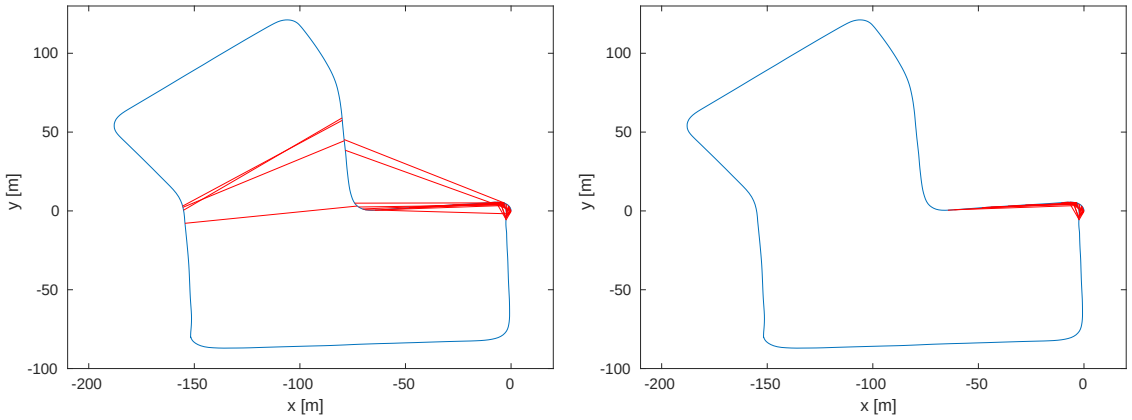
K zlepšeniu výsledkov bol implementovaný krok filtrácie, ktorý prebieha na úrovni zhôd medzi deskriptormi. Pomocou tejto filtrácie bolo odstránené veľké množstvo falošných alarmov. Po aplikácii tohto kroku na testované sekvencie, sa počet falošných alarmov znížil v priemere až o 30%.

Problémom v niektorých špecifických prípadoch je, ak sa napríklad robot vracia po trase, kedy dovidí na miesta, kde už bol (napríklad v opačnom smere cesty, ktorá je predelená len trávnatým povrchom a stromami) a filtráciou sú odstránené aj zhody, ktoré mohli byť využité pri detekcii slučiek.

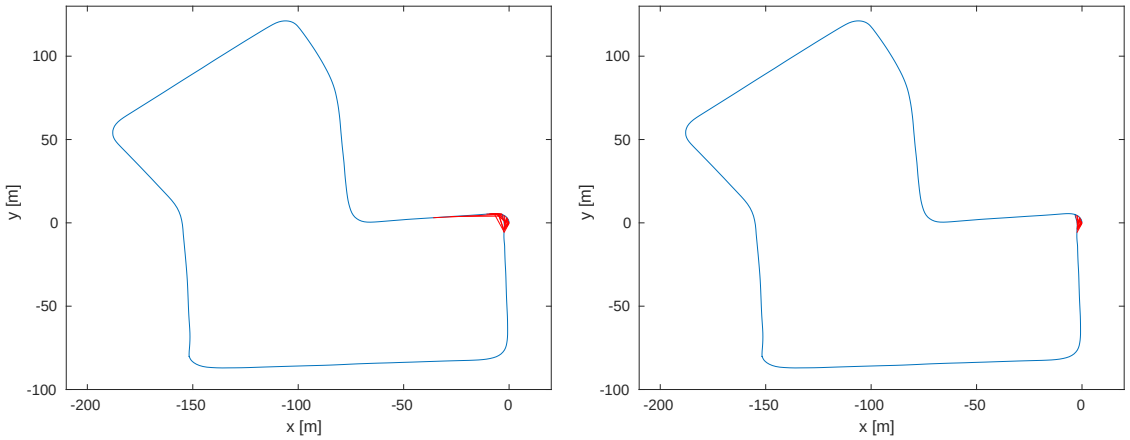
Problémom pri systéme ako celku je závislosť na vstupných parametroch, ktoré je potrebné vhodne zvoliť pre jednotlivé sekvencie, keďže každá sekvencia je iná.

Vyhodnotenie parametrov sytému

Keďže momentálne je systému potrebné jednotlivé parametre zadávať ručne, jedným z budúcich rozšírení systému by mohlo byť zautomatizované vyhodnotenie vhodných parametrov pre beh systému - či už automatické generovanie vhodných parametrov pre rôzne sekvencie,



Obr. 6.10: Sekvencia 07: Vľavo prah 0. Vpravo prah 50.



Obr. 6.11: Sekvencia 07: Vľavo prah 100. Vpravo prah 150.

prípadne vytvorenie univerzálnych parametrov. Takéto vyhodnotenie by mohlo byť na základe rôznych informácií o sekvenciách ako napríklad dĺžka sekvencie, analýza dát rôznych sekvencií, analýza výsledkov z rôznych behov, a podobne.

Zvolenie hodnoty hranice pri filtrácii

Ako bolo spomenuté v predchádzajúcej kapitole, krok filtrácie je závislý na vhodne zvolenej filtračnej hranici. Zautomatizovanie zvolenia tejto hodnoty by mohlo byť tiež vhodným vylepšením tejto práce. To by sa mohlo diať napríklad automatickým vyhodnocovaním pre rôzne hodnoty filtračnej hranice. Vyhodnocovanie by mohlo prebiehať na základe množstva odstránených spojov medzi jednotlivými hodnotami hranice, kde by bolo samozrejme potrebné zisťovať či boli odstránené falošné alarmy alebo správne nájdené páry. Po takomto vyhodnotení by boli vrátené výsledky, ktoré by obsahovali najmenší počet falošných alarmov a najväčší počet správne nájdených párov.

Nástroj k uzatváraniu slučiek

Ďalším rozšírením práce by mohol byť nástroj, ktorý by slúžil k uzatváraniu slučiek na základe detekovaných pozícii, kde sa robot nachádzal viac krát. Vstupom takého nástroja by boli pozície, ktoré sú výstupom implementovaného systému. Nad týmito spojmi bude vhodné ešte vykonať geometrickú verifikáciu pre overenie správnosti slučiek, ktorej výsledkom môžu byť transformácie medzi odpovedajúcimi pozíciami v slučke. Následne môžu byť slučky uzatvorené pomocou vhodného optimalizátora. Uzatvorenie slučky nutne neznamená, že dôjde k prekrytiu dvoch pozícií. Správne uzavretá slučka je aj vtedy, ak sa transformáciou upraví vzdialenosť medzi dvoma pozíciami.

Optimalizácia systému

Vylepšením práce by tiež bola lepšia optimalizácia celého systému, ktorý je momentálne výpočetne veľmi náročný a pre niektoré veľmi dlhé sekvencie nie je možné na bežnom počítači dospieť k výsledkom.

Kapitola 7

Záver

Hlavným cieľom tejto práce bol návrh, realizácia a vyhodnotenie systému, ktorý by mal byť schopný lokalizovať pozíciu robota na základe senzorických dát.

Návrh je inšpirovaný existujúcou metódou pre lokalizáciu a rozpoznanie polohy, s využitím 3D Gestalt deskriptorov na dátach zo senzoru LiDAR Velodyne z databázy `KITTI`.

Bol implementovaný detektor kľúčových bodov, čo obnášalo vyfiltrovanie kľúčových bodov, získaný surový deskriptor, na ktorý bolo aplikované premapovanie a vykonanie redukcie dimenzií deskriptora. Keďže výsledky po premapovaní a redukcii dimenzií deskriptora neboli uspokojivé v dôsledku veľkého množstva falošných alarmov, bol implementovaný filtračný krok, ktorým bol znížený počet falošných alarmov až o 65%. Po tomto kroku boli nájdené zhody pozícií, medzi ktorými sa nachádza len malé množstvo falošných alarmov, a teda na základe výsledkov bolo možné prehlásiť viackrát navštívené miesta robotom za rovnaké pozície.

Na záver boli vykonané a vyhodnotené vhodné testy pre zistenie presnosti systému. Celkové zhodnotenie systému a jeho možné vylepšenia a rozšírenia, ako napríklad lepšie zvolenie hodnoty hranice pri filtrácii, nástroj k uzatváraniu slučiek a iné, sú rozobraté v sekcii 6.4. Okrem implementácie hlavného systému v `C++` boli vytvorené rôzne skripty v `Matlabe`, či už pre filtráciu, vykreslenie nájdených rovnakých pozícií, zobrazení grafu s vyhodnotením separácie, vypočítanie mapovacích funkcií, alebo pre samotné mapovanie deskriptorov. Taktiež bol implementovaný pomocný nástroj v `C++` pre vizualizáciu medzivýsledkov.

Možným pokračovaním práce by mohla byť implementácia geometrickej verifikácie pre overenie správnosti slučky, ktorej výstupom môžu byť transformácie medzi odpovedajúcimi pozíciami v slučke. Následne bude ešte potrebné použiť optimalizátor (napríklad SLAM++, g2o a pod.), ktorý vykoná uzavretie slučky a prepočet pozícií.

Literatúra

[1] Velodyne LiDAR. http://velodynelidar.com/index.html, [Online; navštívené 3.3.2018].

[2] Angeli, A.; Filliat, D.; Doncieux, S.; aj.: *A fast and incremental method for loop-closure detection using bags of visual words.* 2008 IEEE Transactions On Robotics, Special Issue on Visual SLAM, 2008, 1027–1037 s., [Online; navštívené 21.12.2016].
URL http://ieeexplore.ieee.org/document/4633680/

[3] Arvin, F.; Bekravi, M.: *Encoderless position estimation and error correction techniques for miniature mobile robots.* Turkish Journal of Electrical Engineering & Computer Sciences, 2013, [Online; navštívené 30.12.2016].
URL http://journals.tubitak.gov.tr/elektrik/issues/elk-13-21-6/elk-21-6-10-1109-65.pdf

[4] Bekey, G. A.: *Autonomous Robots: From Biological Inspiration to Implementation and Control.* The MIT Press, 2005, ISBN 0-262-02578-7, 481 s., [Online; navštívené 21.3.2018].
URL https://books.google.cz/books?id=3xwfia2DpmoC&printsec=frontcover&hl=sk#v=onepage&q&f=false

[5] Bosse, M.; Zlot, R.: *Keypoint design and evaluation for place recognition in 2D lidar maps.* Elsevier B.V., 2009, 1211-1224 s., [Online; navštívené 10.11.2017].
URL https://www.sciencedirect.com/science/article/pii/S0921889009000992

[6] Bosse, M.; Zlot, R.: *Place recognition using keypoint voting in large 3D lidar datasets.* 2013 IEEE International Conference on Robotics and Automation, 2013, 2677-2684 s., [Online; navštívené 10.11.2016].
URL http://ieeexplore.ieee.org/document/6630945

[7] Brukker, G.; Opatíková, J.: *Veľký slovník cudzích slov.* Robinson s.r.o., 2006, [Online; navštívené 17.12.2016].
URL http://www.voltaire.netkosice.sk/docs/vscs.pdf

[8] Cummins, M.; Newman, P.: *Fab-map: Probabilistic localization and mapping in the space of appearance.* International Journal of Robotics Research, 2008, 647-665 s., [Online; navštívené 6.1.2017].
URL http://www.robots.ox.ac.uk/~mobile/Papers/IJRR_2008_FabMap.pdf

[9] Cummins, M.; Newman, P.: *Highly scalable appearance-only slam fab-map 2.0.* In Robotics Science and Systems RSS, 2009, [Online; navštívené 6.1.2017].
URL http://www.roboticsproceedings.org/rss05/p39.pdf

[10] Dubé, R.; Dugas, D.; Stumm, E.; aj.: *SegMatch: Segment based loop-closure for 3D point clouds*. 2016.
URL https://arxiv.org/pdf/1609.07720v1.pdf

[11] Fraundorfer, F.; Engels, C.; Nister, D.: *Topological mapping, localization and navigation using image collections*. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, [Online; navštívené 6.1.2017].
URL http://cvg-pub.inf.ethz.ch/WebBIB/papers/2007/topnav_final.pdf

[12] Geiger, A.; Lenz, P.; Urtasun, R.: *Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite*. Conference on Computer Vision and Pattern Recognition (CVPR), 2012, [Online; navštívené 30.12.2016].
URL http://www.cvlibs.net/publications/Geiger2012CVPR.pdf

[13] Hanan, S.: *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006, ISBN 978-0-12-369446-1.

[14] Hartley, R. I.; Zisserman, A.: *Multiple View Geometry in Computer Vision*. Cambridge University Press, druhé vydanie, 2004, ISBN 0521540518.

[15] He, L.; Wang, X.; Zhang, H.: *M2DP: A novel 3D point cloud descriptor and its application in loop closure detection*. Oct 2016, 231-237 s., [Online; navštívené 4.3.2018].
URL https://ieeexplore.ieee.org/document/7759060/

[16] Ho, K. L.; Newman, P.: *Detecting Loop Closure with Scene Sequences*. International Journal of Computer Vision, 2007, 261–286 s., [Online; navštívené 4.1.2017].
URL http://dx.doi.org/10.1007/s11263-006-0020-1

[17] Jakab, M.; Benesova, W.; Racev, M.: *3D object recognition based on local descriptors*. Intelligent Robots and Computer Vision XXXII: Algorithms and Techniques, 2015, [Online; navštívené 6.3.2017].
URL http://dx.doi.org/10.1117/12.2083104

[18] Jurišica, L.; Duchoň, F.: *Klasické metódy lokalizácie mobilného robota v prostredí*. AT&P journal, 2010, [Online; navštívené 30.12.2016].
URL
http://www.atpjournal.sk/buxus/docs/casopisy/atp_2010/pdf/online102.pdf

[19] Makarenko, A. A.; Williams, S. B.; Bourgault, F.; aj.: *An experiment in integrated exploration*. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002, 534-539 vol.1 s., [Online; navštívené 28.12.2016].
URL http://ieeexplore.ieee.org/document/1041445/

[20] Mantia, M. L.: *A Virtual Journey through 2D and 3D Elaborations Recorded with Range-Based and Image-Based Method: The Experience of Vitelleschi Palace in Tarquinia*. In Handbook of Research on Emerging Digital Tools for Architectural Surveying, Modeling, and Representation. IGI Global, 2015, 607–653 s., doi:10.4018/978-1-4666-8379-2.ch021, [Online; navštívené 5.1.2017].
URL
http://www.igi-global.com/chapter/a-virtual-journey-through-2d-and-3d-elaborations-recorded-with-range-based-and-image-based-method/133429

[21] Nister, D.; Stewenius, H.: *Scalable Recognition with a Vocabulary Tree*. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, 2161-2168 s., [Online; navštívené 5.1.2017].
URL http://ieeexplore.ieee.org/document/1641018/

[22] Point Cloud Library authors: *How to use a KdTree to search*. [Online; navštívené 3.1.2017].
URL http://pointclouds.org/documentation/tutorials/kdtree_search.php

[23] Scaramuzza, D.; Fraundorfer, F.: *Visual Odometry: Part I - The First 30 Years and Fundamentals*, ročník 18. IEEE Robotics and Automation Magazine, 2011, [Online; navštívené 3.1.2017].
URL https://docs.google.com/viewer?url=
http%3A%2F%2Frpg.ifi.uzh.ch%2Fdocs%2FVO_Part_I_Scaramuzza.pdf

[24] Se, S.; Lowe, D.; Little, J.: *Local and global localization for mobile robots using visual landmarks*. 2001 IEEE/RSJ International Conference on, 2001, 414–420 s., [Online; navštívené 28.12.2016].
URL http://www.cs.ubc.ca/~lowe/papers/iros01.pdf

[25] Siegwart, R.; Nourbakhsh, I. R.; Scaramuzza, D.: *Introduction to Autonomous Mobile Robots*. The MIT Press, druhé vydanie, 2004, ISBN 978-0-262-01535-6.

[26] Sivic, J.; Zisserman, A.: *Video google: A text retrieval approach to object matching in videos*. 2003 In Nineth IEEE International Conference on Computer Vision and Pattern Recognition, 2003, 1470-1477 s., [Online; navštívené 21.12.2016].
URL http://www.robots.ox.ac.uk/~vgg/publications/papers/sivic03.pdf

[27] Steder, B.; Ruhnke, M.; Grzonka, S.; aj.: *Place recognition in 3D scans using a combination of bag of words and point feature based relative pose estimation*. Sept 2011, 1249-1255 s., [Online; navštívené 7.3.2018].
URL https://ieeexplore.ieee.org/abstract/document/6094638/

[28] Wettergreen, D. S.; Barfoot, T. D.: *Field and Service Robotics*. Springer International Publishing, 2016, ISBN 978-3-319-27702-8.

[29] Zhang, H.: *BoRF: Loop-closure detection with scale invariant visual features*. 2011 IEEE International Conference on Robotics and Automation, 2011, 3125-3130 s., [Online; navštívené 20.12.2016].
URL http://ieeexplore.ieee.org/document/5980273/

[30] Zhang, H.; Liu, Y.; Tan, J.: *Loop Closing Detection in RGB-D SLAM Combining Appearance and Geometric Constraints*. Sensors, 2015, 14639–14660 s., [Online; navštívené 3.1.2017].
URL http://doi.org/10.3390/s150614639

Prílohy

Príloha A

Obsah CD

Technická správa

Technická správa vo formáte `PDF` s názvom `TechnickaSprava` sa nachádza na priloženom CD v adresári `text`. Zdrojové kódy pre LaTeXsa nachádzajú v adresári text/src, ktoré je možné preložiť pomocou príkazu `make`.

Plagát

Vytvorený plagát, ktorý prezentuje obsah práce a dosiahnuté výsledky sa nachádza v adresári `poster` s názvom `poster` a je vo formáte `PDF`.

Návod k spusteniu aplikácie

V adresári `docuemntation` sa nachádza návod k nastaveniu a spusteniu aplikácie vo formáte `txt` pre `system` s názvom `spustenieSystemu` a pre pomocný nástroj `visualizer` s názvom `spustenieVisualizeru`.

Zdrojové kódy

Zdrojové kódy implementovaného systému sa nachádzajú v adresári `src/system`, pomocného visualizéru v adresári `src/visualizer` a skriptov v `Matlabe` v adresári `src/system/bin/matlab_scripts`. Preklad systému je možný vykonať správnou konfiguráciou ciest k použitým knižniciam v adresári `src/system` pomocou `cmake-gui` a následne zavolaním príkazu `make` v adresári `src/system/bin`. Obdobné nastavenie je potrebné vykonať aj pre spustenie visualizéru s tým rozdielom, že visualizér sa nachádza v adresári `src/visualizer`. Príklad príkazov k spusteniu preložených programov ako aj podrobnejší popis prekladu sa nachádza v spomínaných návodoch k spusteniu aplikácie v adresári `docuemntation`.
Pre správne fungovanie systému je potrebné pridať do zložiek `src/system/kitti_data/poses` a `src/system/kitti_data/sequences` príslušné `KITTI` dáta.

Aplikácia

V adresári `application` je možné nájst preloženú aplikáciu hlavného systému.

Príloha B

Plagát