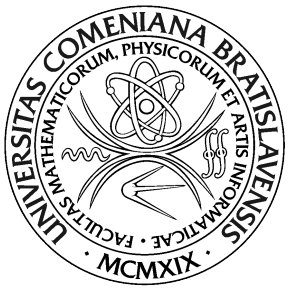
UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



LOKALIZAČNÝ SYSTÉM PRE MOBILNÉHO ROBOTA

Diplomová práca

2018 Bc. Dušan Matejka

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

LOKALIZAČNÝ SYSTÉM PRE MOBILNÉHO ROBOTA

Diplomová práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Mgr. Pavel Petrovič, PhD.

Bratislava, 2018 Bc. Dušan Matejka

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Dušan Matejka
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: aplikovaná informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Lokalizačný systém pre mobilného robota
Localization System for Mobile Robot

Anotácia: Základná úloha pre mobilného robota pohybujúceho sa v známom hoci dynamickom prostredí s cieľom plnenia zadanej úlohy je čo najpresnejšia lokalizácia. Vzhľadom na povahu dát zo senzorov je výhodné využiť pravdepodobnostné reprezentácie a algoritmy. V akademickom roku 2018/2019 sa robotická skupina na Katedre aplikovanej informatiky zúčastní robotickej súťaže SICK Robot Day. Cieľom práce je vyskúmať, navrhnúť a implementovať vhodný lokalizačný algoritmus využívajúci dáta z hĺbkového laserového senzora a navigáciu podľa naplánovanej stratégie pre mobilného robota do tejto súťaže. Výsledkom bude univerzálny lokalizačný systém, ktorý je použiteľný aj v iných podobných scenároch. Predpokladáme využitie výkonnej výpočtovej platformy GPU NVIDIA Jetson TX2.

Literatúra: S. Thrun, W. Burgard, D. Fox: Probabilistic Robotics, MIT Press, 2006.
N. Wilt: The CUDA HANDBOOK. A Comprehensive Guide to GPU Programming, Addison Wesley, 2013.

Vedúci: Mgr. Pavel Petrovič, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 03.10.2016

Dátum schválenia: 04.05.2018 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestne prehlasujem, že túto diplomovú prácu som vypracoval samostatne len s použitím uvedenej literatúry a za pomoci konzultácií u môjho školiteľa.

. . . . . .

Bratislava, 2018 Bc. Dušan Matejka

Poďakovanie

Rád by som sa poďakoval v prvom rade môjmu školiteľovi Mgr. Pavel Petrovič, PhD. za poskytnuté rady a inšpirácie, ktoré ma smerovali správnym smerom a posúvali napred. Ďalej sa chcem poďakovať mojej rodine, priateľom a v neposlednej rade priateľke za ich podporu, bez ktorej by som to určite nezvládol.

Abstrakt

Táto práca sa venuje problematike lokalizácie robota v známom prostredí. Súčasťou tejto práce bolo zúčastniť sa súťaže SICK robot day 2018, kde sme boli s naším robotom Mikešom súťažiť proti tímom z ďaľších univerzít. Na lokalizáciu boli použité údaje z odometrie, laserového senzoru tim571 a kompasu. Údaje z laserového senzora boli spracovávane Houghovou transformáciou pre zostrojenie stien a rohov z bodových lúčov laserového senzora. Ďalej za pomoci kompasu sa určilo na ktorej strane ihriska sa robot práve nachádzal, kvôli symetrii. Taktiež bolo potrebné vytvoriť navigačný a strategický systém pre súťaž. Počas súťaže SICK robot day 2018 výsledky z lokalizácie a navigácie nášho robota Mikeš dosahovali 100% spoľahlivosť.

Kľúčové slová: mobilny autonomny robot, Houghová transformácia, počítačové videnie

Abstract

This work deals with the problem of robot localization in a known environment. Part of this work was to participate in the SICK robot day 2018 competition, where we competed with our robot Mikeš against teams from other universities. Odometry, tim571 laser sensor, and compass data were used for localization. Laser sensor data were processed by the Hough transform to construct walls and corners from the laser beam spots. In addition, with the help of a compass, it was determined on which side of the field the robot was just for symmetry. It was also necessary to create a navigation and strategic system for competition. During the SICK robot day 2018 competition, the results of localization and navigation of our Mikeš robot reached 100% reliability.

Keywords: mobile autonomous robot, Hough transform, computer vision

Obsah

Kapitola 1

Úvod

Lokalizácia v robotike je jedna z najzložitejších a najdôležitejších procesov. Robot potrebuje vedieť, kde sa nachádza a čo vidí, aby sa vedel rozhodnúť, čo môže ďalej vykonať. Aby sa robot dostal z miesta A na miesto B, musí presne poznať svoju aktuálnu polohu. V prípade, že sa robot stratí a nepozná svoju polohu, riskuje kolíziu s okolitým prostredím, ktorá ho následne môže aj poškodiť. Preto je veľmi dôležité udržiavať a prepočítavať aktuálnu polohu a jej pravdepodobnosť. Keďže senzory sú dosť nepresné a hlavne často chybové, robot si musí udržiavať viacej možných pozícií na mape, aby sa v prípade chybného merania nestratil a vedel opraviť chybné meranie.

Pre sonarové senzory je veľmi dôležitý druh materiálu a uhol odrazu, vďaka ktorému sa vysielané zvukové vlny nemusia vrátiť, napriek tomu, že sa objekt nachádza priamo pred robotom. Naproti tomu, laserové senzory, vidia cez sklo a nevidia čiernu farbu, ktorá pohlcuje vyslané lúče.

Pomocou odometrie sa dá udržovať platná poloha, ale čím dlhšie sa ňou robot riadi, tým väčšia chyba sa naakumuluje. Zdrojom takejto chyby môže byť napríklad šmykľavý povrch, nerovnomerné otáčanie kolies a veľa iných faktorov, ako napríklad aj zásah človeka.

Z toho dôvodu je potrebné, aby robot zvážil každú nameranú hodnotu a rozhodol sa, ako ovplyvniť stav jeho aktuálnej lokalizácie.

Mojou motiváciou je vylepšenie existujúcich riešení a nájdenie efektívneho spôsobu pre určenie polohy v známom prostredí, ktorý sa bude naďalej využívať vo výučbe a vo výskume. Ďalej ma motivuje aj možnosť zúčastniť sa na súťaži a dosiahnúť porovnateľné výsledky ako iné európske tímy, ktoré sa tejto prestížnej súťaže zúćastňujú. Čakajú nás ťažkí súperi z celej Európy, s oveľa väčším počtom účastníkov v jednotlivých tímoch.

Nasledujúce kapitoly venujem použitým senzorom, ich vlastnostiam a existujúcim technológiam. Potom stručne spomeniem robota Mikeša a súťaž SICK robot day 2018. Následne uvediem stručný návrh riešenia. Potom sa presuniem k samotnému riešeniu a rozoberiem dosiahnuté výsledky.

Kapitola 2

Prehľad problematiky

V tejto kapitole sa oboznámime s problémami, ktoré je nutné riešiť pri návrhu a implementácii lokalizácie robota v priestore.

Problematiku môžeme rozdeliť do viacerých základných častí.

1. Problematika spracovania údajov senzorov
2. Problematika implementácie

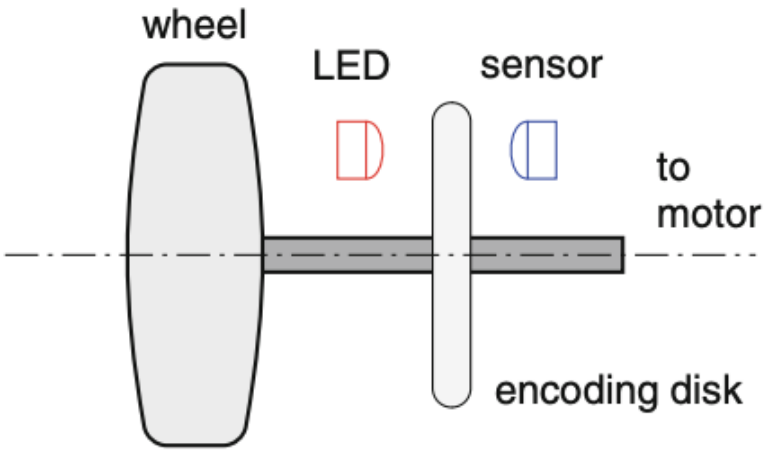
2.1 Problematika spracovania údajov senzorov

Aby sa robot vedel sám orientovať, potrebuje získavať údaje o prostredí, v ktorom sa nachádza, pomocou svojich senzorov. Poznáme veľké množstvo rôznych senzorov, ale my sa teraz zameriame na tieto základné senzory:

- Otáčkový senzor
- Sonarový senzor
- Laserový senzor
- Kompas

2.1.1 Otáčkový senzor

Otáčkový senzor vzhľadom na svoju jednoduchosť má pomerne veľkú presnosť. Každé koleso má svoj vlastný otáčkový senzor, pomocou ktorého sa počíta počet pootočení jednotlivého kolesa. Týmto vieme zistiť pomerne presne, ako ďaleko a akým smerom robot zmenil svoju polohu. Napríklad, ak každé koleso urobilo rovnaký počet otáčok, tak náš robot sa pohol dopredu o počet otáčok vynásobený obvodom kruhu kolesa.



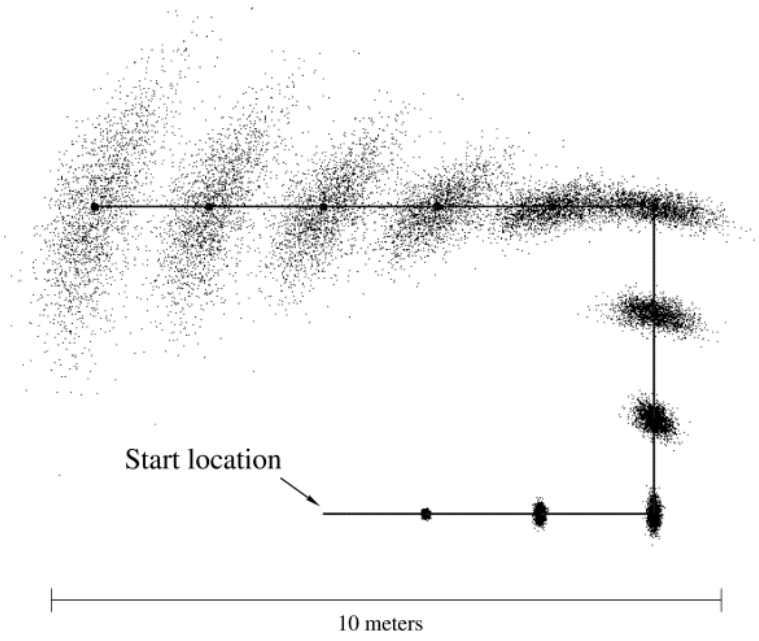
Obr. 2.1: Princíp otáčkového senzora [MBA17]

V prípade, že sa robot riadi len otáčkovými senzormi, prináša to viacej problémov. V prvom rade, potrebuje robot poznať svoju presnú počiatočnú polohu a nesmie byť presunutý počas svojho fungovania alebo nesmie byť ovplyvnený vonkajšími silami. Aj napriek tomu, ak by sme dokázali zabezpečiť, že nenastane ani jedna z uvedených situácií, čím dlhšie robot aktualizuje svoju polohu iba pomocou otáčkových senzorov, tým viac sa naakumuluje chybovosť merania. Napríklad, ak na jeden meter, prejdený rovno, môže nastať odchýlka jeden centimeter a jeden stupeň. Tak akú veľkú chybu vo svojej

polohe môže mať po prejdení tisíc metrov? V takomto prípade sa robot nachádza už na úplne inom mieste ako si myslí. Tieto chybné merania môžu vzniknúť na nerovnej ploche, na šmykľavom povrchu alebo pri mnohých iných faktoroch.

Parametrom otáčkového senzora je počet impulzov za jednu otáčku. Napríklad LEGO MINDSTORMS to má vyladené tak, že jeden impulz dostáva pri každom jednom stupni.

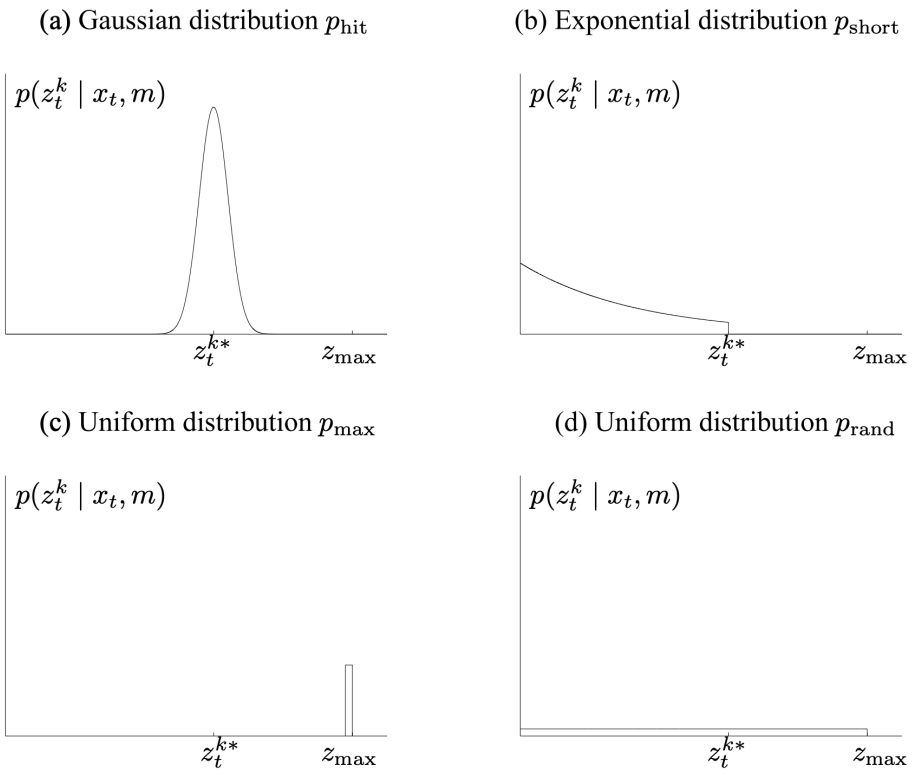
Poznáme dva druhy otáčkového senzora. Jeden senzor pozná iba výsledný počet nameraných impulzov. Pomocou dvoch fázovo posunutých senzorov dokáže určiť aj smer podľa poradia, v ktorom nastávajú jednotlivé impulzy krátko za sebou.



Obr. 2.2: Akumulácia chýb lokalizácie pomocou odometrie [ST03]

2.1.2 Sonarový senzor

Sonarový senzor používa techniku, pri ktorej vysiela zvukové vlny do priestoru a meria dĺžku pomocou doby, za ktorú sa odrazené vlny od predmetu vrátia naspäť do senzora. Takýto senzor má bohužiaľ dosť krátky dosah a jeho funkčnosť závisí od materiálu, uhlu odrazu a tvaru povrchu. Ak sa väčšina vĺn odrazí zlým smerom alebo sú absorbované prostredím, tak tento senzor nemusí spozorovať predmet, ktorý sa nachádza v jeho tesnej blízkosti. Okrem toho, sa mu môžu vrátiť poodrážané vlny z predchádzajúceho merania a tým pádom dokáže oznámiť prekážku, aj keď sa reálne v danej vzdialenosti vôbec nenachádza.

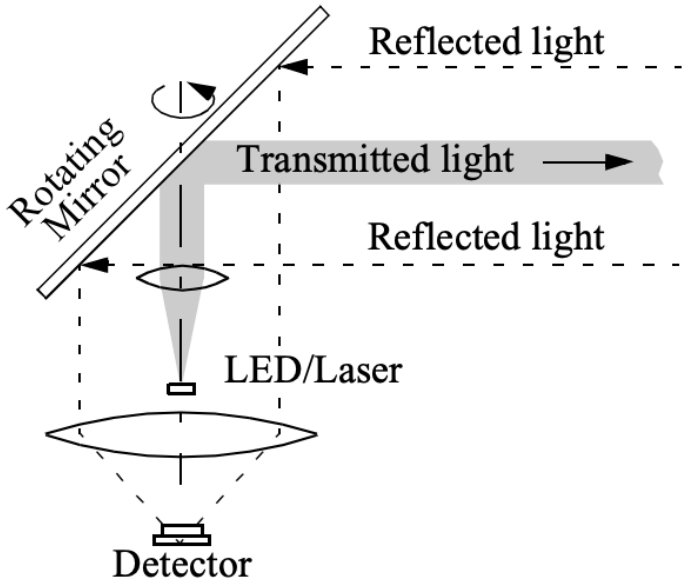


Obr. 2.3: Pravdepodobnostné rozdelenia [ST00]

Merania senzorov dosahujú gausové rozdelenie v prípade korektného merania, neočakávané objekty predstavujú krátke exponenciálne rozdelenie a chyby tvoria uniformné rozdelenie centrované v maxime. Pri náhodných meraniach sa prejavuje náhodná uniformná distribúcia.

2.1.3 Laserový senzor

Laserový senzor používa na určenie vzdialenosti od prekážky množstvo svetelných lúčov, ktoré vysiela rôznym smerom a meria čas, kým sa vyslané lúče vrátia naspäť k senzoru. Niektoré laserové senzory poskytujú aj informácie o intenzite odrazu a vďaka tomu môžu rozpoznávať aj tmavé plochy. Takéto senzory vedia poskytnúť okrem vzdialenosti aj doplňujúcu informáciu o odrazivosti zameraného povrchu. Z toho dôvodu má senzor pomerne dobrú presnosť a spoľahlivosť svojich meraní, ale keďže jeho lúče prechádzajú cez sklo, nevie spozorovať napríklad sklenené dvere a považuje ich za voľný priestor. Okrem toho jeho svetelné lúče sú pohlcované čiernou farbou, od ktorej sa neodrážajú.



Obr. 2.4: Princíp laserového senzora s otáčavým zrkadlom [RS04]

2.1.4 Kompas

Kompas sa dá použiť na meranie uhlu trajektórie, len ako doplnková informácia. Výsledky jeho merania sú ovplyvnené prostredím, kde sa nachádzajú objekty obsahujúce kov alebo magnetické vlastnosti. Merania z takéhoto zariadenia sú často veľmi skreslené a chybné, môžu sa mýliť o desiatky stupňov. Takýto zdroj je nevhodné použiť na určenie otočenia na mape, určite nie za takýchto podmienok.

Azimut voči severu sa určuje na základe merania magnetického poľa pomocou magnetometra v troch osiach. Jednotlivé siločiary magnetického poľa sú na rôznych miestach na Zemi odlišné a preto sa musí kompas zakaždým prekalibrovať. Pre kalibrovanie sa používa program, v ktorom sa postupne počas jednotlivých meraní hľadajú minima a maxima v jednotlivých osiach.

Následne sa nájde stredná hodnota, ktoré sa potom definujú ako posun súradnicovej sústavy. Tento posun je na jednotlivých miestach rôzny a preto sme tento úkon museli vykonať aj na mieste súťaži vo Waldkirchu.

2.2 Problematika implementácie

V tejto sekcii poukážem na základné problémy, na ktoré sa treba zamerať, vzhľadom na obmedzenú výpočtovú silu.

Treba sa zamerať na základné problémy:

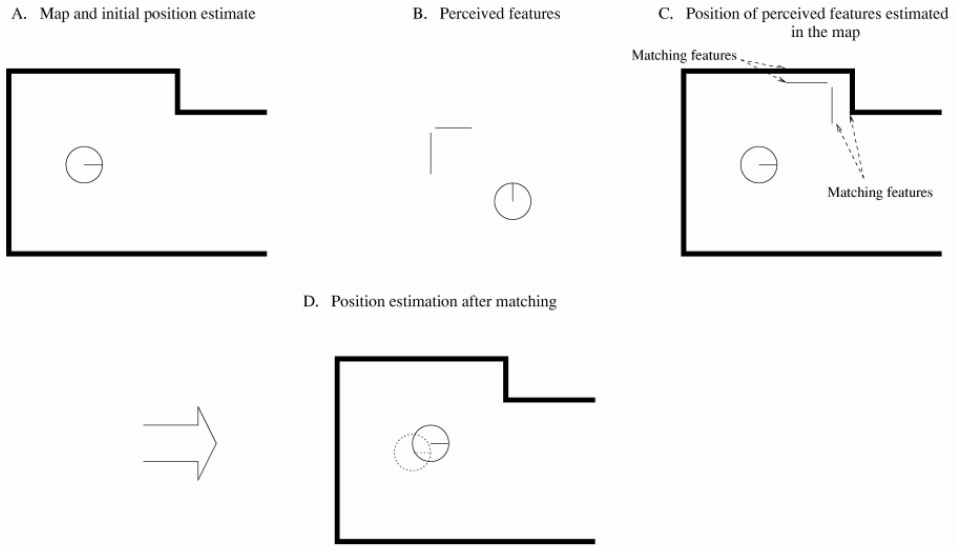
- spracovanie údajov zo senzorov
- napasovanie spracovaných údajov do mapy
- vyhodnotenie zmeny stavu robota

2.2.1 Spracovanie údajov zo senzorov

Základným problémom bude spracovanie údajov z laserového senzora, ktoré sú posielané cez sieť ako páry hodnôt vzdialenosti a intenzity odrazu. Následne z jednotlivých bodov bude treba rozumne efektívne zostrojiť obraz, ktorý dokázal nasnímať. Ďalej bude treba z bodov zostrojiť úsečky na viditeľnom povrchu objektov v prostredí, ktoré by sa dali využiť. Menej náročné bude spracovanie signálov zo sonarového senzora, ktorý len bude oznamovať, či vidí alebo nevidí pred sebou prekážku. V neposlednom rade sa zohľadní údaj z kompasu, aby sa náhodou nestalo, že si robot bude myslieť, že je otočený opačným smerom, v prípade symetrických priestorov.

2.2.2 Napasovanie spracovaných údajov do mapy

V prípade získaných a extrahovaných údajov bude potrebné z jednotlivých prekážok určiť, kde pravdepodobne sa robot aktuálne nachádza. Jednotlivé možnosti si bude treba uchovávať a aktualizovať ďalším prísunom nových dát zo senzorov. Táto akcia bude časovo zložitejšia, vzhľadom na množstvo dát, ale nebude ju potrebné vykonávať veľmi často, aby sa chybné hypotézy neposilňovali a správne náhodou nezanikali.



Obr. 2.5: Vyhodnotenie viditeľných prekážok [DF15]

Na obrázku 2.5 je vidno ako robot pomocou nameraných údajov zo senzorov dokázal priradiť správny roh na mape, ale v dôsledku chybovosti senzora jeho poloha je posunutá oproti skutočnosti.

2.2.3 Vyhodnotenie zmeny stavu robota

Následne po určení rôznych možných polôh, bude treba vybrať najvhodnejšieho kandidáta a rozhodnúť sa ako pokračovať ďalej vo svojej trase. Taktiež

bude treba brať do úvahy kritické situácie, keď senzory hlásia pred sebou neočakávanú prekážku, aby sa robot vyhol nebezpečným kolíziám s okolitým priestorom.

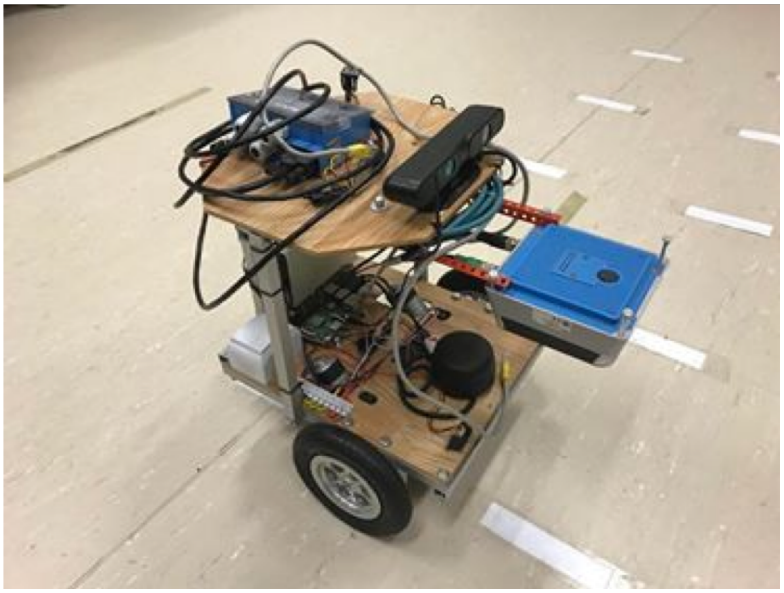
2.3 Robot Mikeš

Konštrukcia robota Mikeša je navrhnutá a skonštruovaná z dreva a hliníka. Robot sa skladá z dvoch drevených podlaží, ktoré sú navzájom spojené hliníkom. Na nich sú namontované jednotlivé komponenty, ktoré sa používali v predchádzajúcich prácach a súťažiach.

Na spodnom podlaží sa nachádzal pôvodný laserový senzor RPLidar A2, ktorý bol použitý v predchádzajúcej bakalárskej práci Integrovaný navigačný systém pre mobilného robota [Zem17], kde sa tento senzor používa pre ohodnotenie jednotlivých hypotéz Monte Carla. Podobný laserový senzor Hokuyo LiDAR bol použitý na robotovi Smelý Zajko v diplomovej práci Robotour with Laser Range Sensor [Du7], ktorý je využívaný na detekovanie prekážok. V tejto práci sa zameriavame na priamu výpočtovú lokalizáciu porovnávaním nameraných údajov a mapy a je pre nás veľmi dôležitá presnosť výslednej lokalizácie na centimeter.

V diplomovej práci Robot Poštár [Mad14] sa pre lokalizáciu použila metóda SURF, kde sa robot lokalizuje pomocou nájdených význačných bodov v obraze. Autor tu popisuje, že pre jednoznačné určenie polohy potrebuje dve dvojice bodov, keďže jedna dvojica vytvára kruhový výsek možných pozícii pre rovnaký uhol pod ktorým sa dajú jednotlivé body nasnímať. U nás použijeme podobný princíp prieniku dvoch kružníc pre určenie polohy, ale nám budú stačiť dva susedné rohy, keďže náš laserový senzor poskytuje informáciu o vzdialenosti.

Riadiaca jednotka robota Mikeša je Raspberry PI 3, ktorý disponuje štvorjadrovým ARM 64-bitovým procesorom o frekvencii 1.2 GHz. Obsahuje grafický čip VideoCore IV s taktom 300 MHz a operačnou pamäťou o veľkosti 1 GB. Výstupné porty tohto zariadenia sú štyri USB 2.0, HDMI a LAN. Rasberry PI 3 nemá zabudovaný harddisk, namiesto toho používa Mikro SD kartu, na ktorej je nainštalovaný operačný systém Raspbian GNU/Linux 8 (jessie) s verziou kernelu Linux 4.9.35-v7+.



Obr. 2.6: Robot mikeš pred súťažou

Dôležitou súčasťou Mikeša je Arduino Nano. Arduino je open-source platforma, ktorá obsahuje ľahko ovládateľný hardvér a softvér pre čítanie a zapisovanie z digitálnych aj analógových vstupov a výstupov. Aktuálne sa využíva pre riadenie podvozku a neskôr využijeme ďalšie Arduino Nano pre ovládanie riadenia seriev, sklápanie zbernej radlice a otváranie dvierok. Programy pre ovládanie tohto hardvéra sa píšu v jazyku C++ a následne sa pomocou prostredia Arduino IDE priamo nahrávajú do mikrokontroléru na doske

Arduina.

```
#define TRIG 4
#define ECHO 3

Servo s;
Servo d;

void setup()
{
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);

  s.attach(10);
  d.attach(9);
  Serial.begin(115200);
  s.write(0);
  d.write(100);
}
```

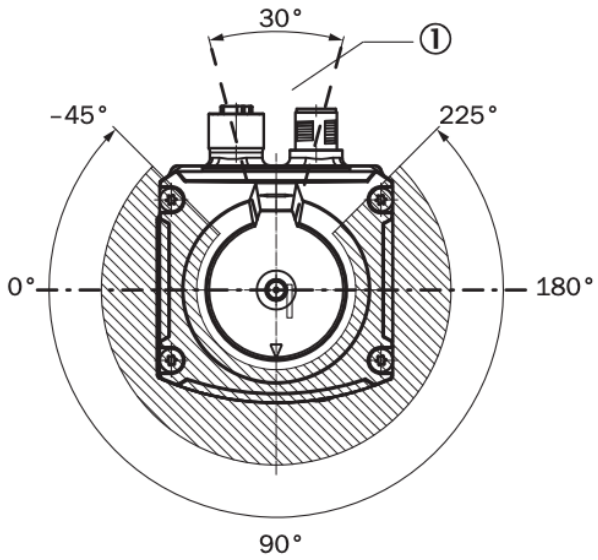
Obr. 2.7: Príklad inicializácie dvoch seriev pomocou Arduina

Každý program musí obsahovať dve funkcie. Prvou funkciou je setup(), ktorá sa spúšťa len raz pri inicializácii Arduina, druhou funkciou je loop(), ktorej volanie sa opakuje až do ukončenia programu.

2.4 Laserový senzor TiM 571

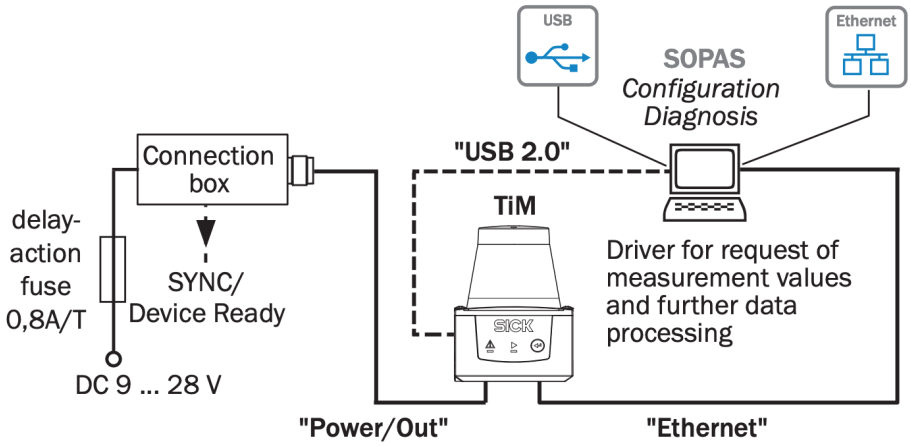
Tento senzor bol poskytnutý pre fakultu od spoločnosti SICK pre účasť v súťaži SICK robot day 2018. Tento senzor disponuje rozhľadom 270° s použitím technológie HDDM (High Definition Distance Measurement) pre meranie vzdialenosti. Technológia HDDM používa na vytvorenie jednej hodnoty prie-

mer viacerých lúčov. Konkrétne tento model, pre zostrojenie jednej výslednej hodnoty 0,3°, využíva priemer 84 vyslaných lúčov. Na určenie vzdialenosti jedného lúča používa klasický princíp merania času odrazeného lúča. Pre dosiahnutie viditeľnosti uhlového rozsahu používa rotovacie zrkadlo. Dosah tohto senzora je až 25 metrov pri odrazivosti materiálu aspoň 50% a pre tmavé materiály s väčšou odrazivosťou ako 10% 8 metrov.



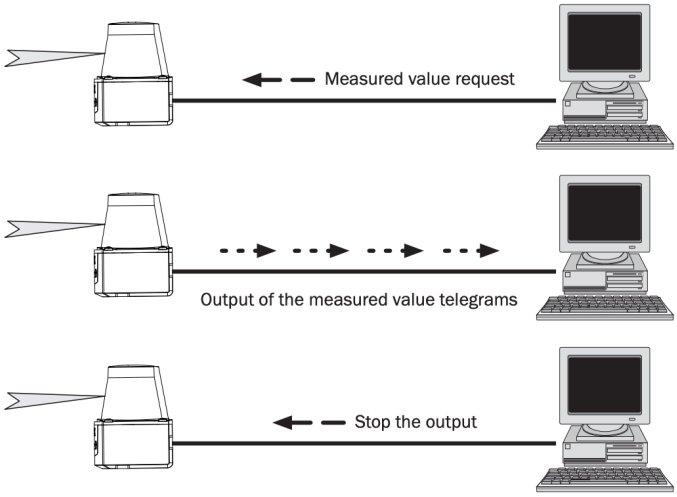
Obr. 2.8: 270° uhlový rozsah laserového senzoru TiM 571 [SIC16]

Senzor je dodávaný aj so softvérom a ovládačmi pre operačný systém Windows. Tento softvér okrem spracovania a zobrazovania výsledkov meraní zo senzora slúži aj pre konfiguráciu nastavení senzora. Tieto príkazy v dokumentácii nie sú popísané, ale budeme ich musieť nastaviť pomocou softvéru na operačnom systéme Windows. Takto vieme nastaviť senzoru, aby nám posielal aj jednotlivé hodnoty RSSI, okrem samotnej vzdialenosti.



Obr. 2.9: Prehľad rozhraní laserového senzora TiM 571 [SIC16]

Senzor neobsahuje knižnicu pre sieťovú komunikáciu pre jazyk C/C++ a preto túto komunikačnú vrstvu budeme musieť neskôr vytvoriť sami. Táto komunikácia funguje v dvoch režimoch. Prvý režim je taký, že si za každým vypýtame od senzora jednu správu s nameranými hodnotami. V takomto prípade vyšleme senzoru správu <STX>sRN LMDscandata<ETX>, kde nám senzor následne odpovie v tvare <STX>sRA LMDscandata (kontent správy) <ETX> a ďalej čaká na nový príkaz. Druhý režim, v ktorom vie fungovať je, že si vyžiadame opätovné posielanie správ vo frekvencii 15 Hz. Takúto komunikáciu začneme správou <STX>sEN LMDscandata 1<ETX>, na ktorú dostaneme potvrdenie vo forme správy <STX>sEA LMDscandata 1<ETX>. Následne dostávame opätovné správy vo forme <STX>sRA LMDscandata (kontent správy) <ETX>. Pre ukončenie tejto komunikácie treba poslať správu <STX>sEN LMDscandata 0<ETX>, na ktorú nám senzor odpovie <STX>sEA LMDscandata 0<ETX>.



Obr. 2.10: Priebeh komunikácie so senzorom TiM 571 [SIC16]

Konkrétny obsah správy je podrobne rozpísaný v dokumentácii. Na začiatku každej správy sa nachádzajú informácie o stave senzora a až za nimi sa nachádzajú jednotlivé údaje o vzdialenostiach a aj hodnoty RSSI v prípade, že je senzor tak nakonfigurovaný.

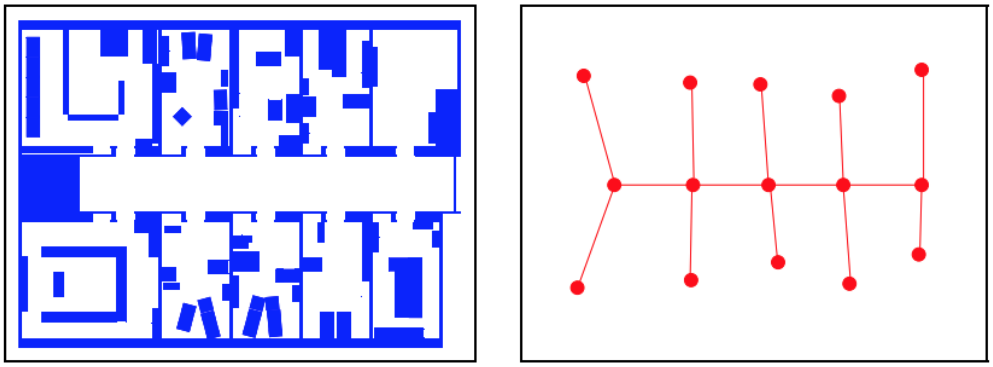
2.5 Lokalizácia

Lokalizácia je problém určenia polohy v rámci mapy daného prostredia. Tento proces sa dá nazvať tiež ako odhad polohy alebo sledovanie polohy. Lokalizácia je jeden zo všeobecných problémov v robotike, pretože skoro každá úloha vyžaduje poznať aktuálnu polohu robota a objektov, s ktorými potrebuje manipulovať.

Na lokalizáciu sa dá pozerať ako na problém transformácie koordinátorov. Mapy sú zobrazené v globálnom súradnicovom systéme a sú nezávislé od lokálnej pozície robota. Lokalizácia je následné vytvorenie zhody pomocou transformácie medzi lokálnym a globálnym súradnicovým systémom. Vytvo-

renie takejto úspešnej transformácie umožňuje robotovi poznať polohu jednotlivých objektov mapy vo svojej vlastnej lokálnej súradnicovej sústave.

Poznáme základné tri druhy mapy - metrické, topologické a hybridné. Metrická mapa sa dá reprezentovať ako mriežková mapa, ktorá sa často využíva v mnohých robotických systémoch. Každá bunka v takejto mape obsahuje informáciu o pravdepodobnosti obsadenosti. Veľkou nevýhodou takejto mapy je, že nedokáže reprezentovať entity, ako sú napríklad dvere. Topologická mapa je v podstate graf, v ktorom jednotlivé vrcholy predstavujú výnimočné vlastnosti a hrany v takomto grafe reprezentujú jednotlivé vzťahy medzi vrcholmi. Takáto reprezentácia zaberá oveľa menšiu kapacitu, na rozdiel od metrickej mapy.

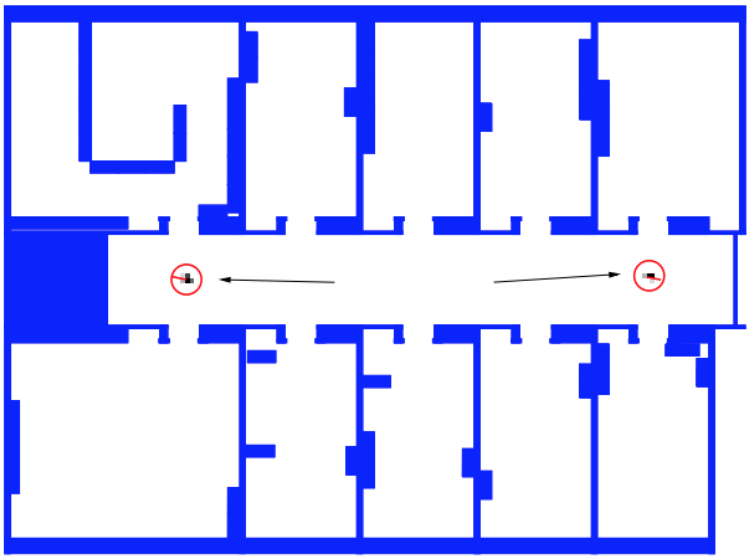


Obr. 2.11: Príklad zobrazenia mapy pomocou metrickej a topologickej reprezentácie [ST00]

Hybridná mapa je kombináciou oboch typov máp a využíva výhody oboch spôsobov. Kde vzhľadom na väčší celok sa lokalizuje pomocou topologickej reprezentácie a vzhľadom na jednotlivý vrchol sa reprezentuje pomocou metrickej reprezentácie.

Pre úspešnú lokalizáciu je potrebné riešiť mnoho rôznych problémov. Position tracking je metóda na určenie lokálnej polohy vzhľadom na zaznamе-

nané hodnoty z jednotlivých senzorov a predpokladá, že pôvodná poloha je známa. Global localization je zložitejší problém, ako position tracking, vzhľadom na to, že nepozná svoju pôvodnú polohu a zahŕňa v sebe aj Position tracking. Kidnapped robot problem je varianta problému Global localization, kde počas lokalizácie môže byť robot presunutý z jedného miesta na druhé. Dôležitou vlastnosťou dobrého lokalizačného systému je samozrejme obnovenie lokalizácie z chybového výsledku.



Obr. 2.12: Príklad zobrazenia, kedy sa robot vzhľadom na globálnu lokalizáciu nevie rozhodnúť, kde sa nachádza vzhľadom na symetrické lokálne prostredie. Robot sa musí presunúť do jednej z miestností, aby vedel určiť svoju polohu. [ST00]

Ďalšou dimenziou zložitosti pre lokalizáciu, je typ prostredia, ktoré môže byť statické alebo dynamické. Lokalizácia v dynamickom prostredí je samozrejme oveľa komplikovanejšia ako v statickom. Lokalizáciu rozdeľujeme medzi pasívnu a aktívnu, kde pri pasívnej robot len pozoruje svoje okolie a pri aktívnej rozhoduje aj o svojom pohybe. Aktívna lokalizácia dosahuje zvyčajne

lepšie výsledky ako pasívna. Ďalším problémom je prípadne lokalizovanie viacerých robotov súčasne.

2.6 Navigácia

V tejto sekcii sa oboznámime s niektorými navigačnými algoritmami podľa článku A Study of Distinctive Localization and Navigation Methods for Mobile Robots [SMUM15]. Jednotlivé názvy nebudeme prekladať.

- Active Beacons
- Landmark-based Navigation
- Line Navigation
- Map-based Navigation
- GPS Navigation System

Active Beacons sa dá využiť pri navigovaní napríklad pomocou hviezd s použitím princípu triangulácie a trilaterácie.

Landmark-based Navigation v prípade použitia prírodných orientačných bodov používa zvyčajne križovatky alebo rohy, za svoje význačné body, ktoré sa dajú jednoduchšie rozpoznať. V prípade využitia umelých význačných bodov je proces rozpoznávania jednoduchší a ľahšie implementovateľnejší.

Roboty využívajúce Line Navigation sa v praxi dlho používajú a sú nazývané Automatic Guided Vehicles (AGVs). Takýto robot sa musí neustále nachádzať v blízkosti svojej čiary, aby ju vedel sledovať.

Map-based Navigation je spôsob, pri ktorom sa pomocou získaných údajov zo senzorov vytvorí mapa okolitého prostredia a následne sa hľadá zhoda

medzi získanou a známou mapou. Ak sa takáto zhoda nájde, robot dokáže určiť svoju skutočnú polohu.

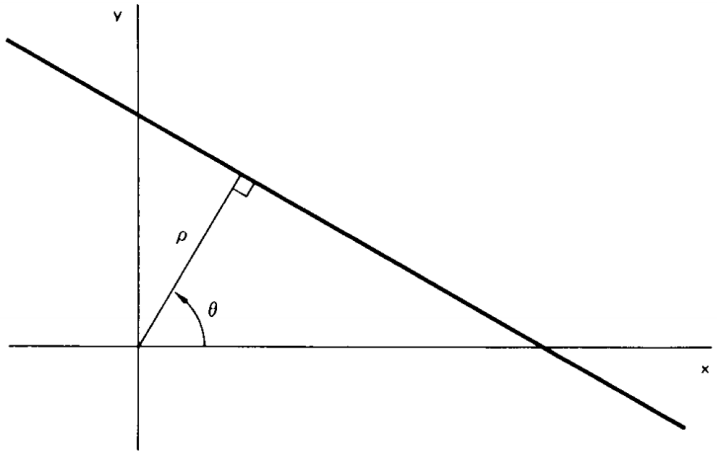
Navigovanie pomocou GPS Navigation System nie je vhodné pre mobilné roboty z dôvodu vysokej nepresnosti 15 metrov. Obzvlášť GPS vyžaduje dostupnosť signálu aspoň zo štyroch satelitov pre jednoznačné určenie polohy a je nevhodná pre navigovanie v uzavretom prostredí.

2.7 Houghova transformácia

Houghova transformácia je algoritmus, ktorý sa najčastejšie využíva v počítačovom videní pre extrakciu priamok z obrazu, ale taktiež kružníc. Najskôr je potrebné získať množinu bodov pomocou detekovania hrán v obraze, ktoré sa následne budú spracovávať.

Algoritmus si na začiatku vytvorí maticu všetkých možných kombinácii. Pôvodný Houghov návrh využíval reprezentovanie priamky pomocou smernicového vyjadrenia. Takéto definovanie bolo, ale nevhodné pre zvislé priamky. Tento nedostatok bol odstránený reprezentáciou priamky uhlom zvierajúcim medzi priamkou a osou x súradnicovej sústavy a vzdialenosťou priamky od počiatku.

Pre každý bod z obrazu získaný pomocou detekcie hrán sa následne zisťuje, či tento bod je súčasťou jednotlivých priamok, respektíve či sa jednotlivý bod nachádza v blízkosti jednotlivej priamky. Ak sa bod nachádza dostatočne blízko priamky, tak takáto priamka si pripočíta jeden hlas v matici za daný bod. Na koniec algorimu sa zozbierajú maximálne hodnoty z matice, ktoré reprezentujú jednotlivé vyhľadané priamky obrazu.

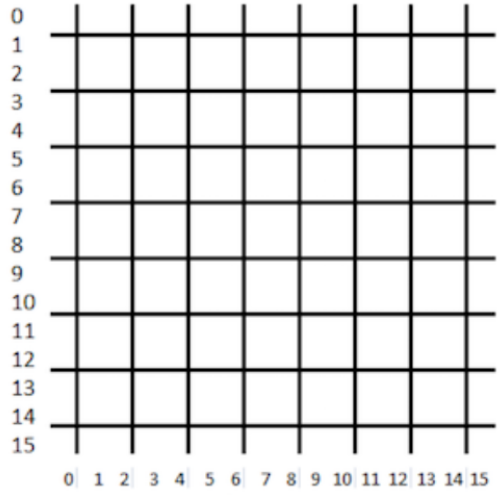


Obr. 2.13: Reprezentovanie priamky pomocou polárnej súradnice [DH72]

2.8 Prvá modelová štúdia

Cieľom tejto štúdie bolo zoznámiť sa s navigáciou robota, spracovaním senzorových údajov v priestore a následné mapovanie na jednoduchšom príklade.

Na tento projekt sme použili robota vybaveného s troma ultrazvukovými senzormi, ktorý sa vedel pohybovať sledovaním čiary po mriežkovej dráhe. Skutočná plocha obsahovala mriežku o veľkosti 8 x 8, kde každý štvorec mal veľkosť 25 centimetrov. Túto mriežku sme reprezentovali v systéme ako 16 x 16 a s veľkosťou jedného štvorca 12,5 centimetra. Každý ultrazvukový senzor má svoj vlastný lokálny koordinačný systém, ktorý je potrebné transformovať do globálneho koordinačného systému a následne do samotnej konkrétnej mriežky.



Obr. 2.14: Reprezentácia mapy pomocou mriežky o veľkosti 16 x 16

Na reprezentáciu mriežky sme použili dve dvojrozmerné polia. Po každom vykonanom pohybe sme pomocou nameraných hodnôt z jednotlivých ultrazvukových senzorov do jedného poľa zaznamenali, na ktorých mriežkach pozoroval robot prekážku a do druhého poľa pre všetky mriežky medzi prekážkou a robotom zaznamenali voľné miesto. Následne pravdepodobnosť prekážky v danej mriežke sme vyhodnotili ako podiel medzi počtom úspešných pozorovaní prekážky a celkovým počtom meraní danej mriežky.

Model ultrazvukového senzora sme definovali ako kužeľ, ktorý je tvorený dvoma priamkami. Funkcie týchto dvoch priamok sme definovali pomocou pozorovania vlastností tohto senzora a jeho umiestnenia. Rovnica ľavej priamky kužeľa predného ultrazvukového senzoru

$$-\frac{100}{15.2+101} * x + (-1 + \frac{100}{15.2+101}) * y + 1 = 0$$

a rovnica pravej priamky kužeľa predného ultrazvukového senzoru

$$\frac{100}{15.2-101} * x + (-1 - \frac{100}{15.2-101}) * y + 1 = 0$$

Ak aspoň jeden roh bunky mriežky sa nachádza medzi týmito priamkami alebo ľavý predný roh sa nachádza vľavo od ľavej priamky a pravý predný roh sa nachádza napravo od pravej priamky, tak takúto bunku považujeme za pozorovanú.

```
-----see obstacle----------------
 0   0   0   1   1   1   0   0   0   0   0   0   0   0   0   0
 0   0   0  20  15   1   0   0   0   0   0   2   2   5  14   0
 0   0   1  54  57  11  13   0   0   1   1   1   0   3  12   0
 0   0   1  52  46   0   0   0   0  25   0   0   1   4  12   0
 0   0   0  28  14   0  24   2   0   8   0  14   1   2  12   0
 0   0  11   9   0   0   1   3   0   0   0   9   1   1   0   0
 0   0  11   9   0   0   0  21  83  83  47   3   0   0   0   0
 0   0  18  16   0   0   0  35  81  81  34  14   1   0   2  11
 0   0  18  16   0   0   0   1   4   5   5   1   1   0   2  11
 0   0  18  18   7   7   0   0   1   2   1   1   1   0  18  13
 0   0   7  54   3   3  11  10  10   0   0   1   1  18  24   4
 0   0   7  75  39  39  24   1   1   0   0   0   0  18  22   3
 0   0   0  49  57  24  Rv  13  13   0   0   0   0  18   6  54
 0   0   0   7  46   1   1   1   1   1   0   2   2   2   2  55
 0   0   0   8 106   1   0   0   0   0   0   3   3   3   3 115
 0   0   0   2  61   2   1   0   0   0   0   1   1   1   1  60
```

Obr. 2.15: Namerané prekážky v mriežke

Robot sa pohyboval po mriežke pomocou príkazov operátora, ktoré dostával prostredníctvom Bluetooth komunikácie. Pôvodne ultrazvukové senzory snímali situáciu aj počas pohybu smerom rovno, čo ale viedlo k chybným údajom, pretože robot sa počas sledovania čiary pohyboval po kúskoch sprava doľava, aby sa udržal na svojej ceste. Takýmto spôsobom snímal predmety, ktoré sa v skutočnosti nenachádzali v danom smere. Preto sa takéto meranie vykonávalo po každom ukončenom pohybe a tiež po zaslaní príkazu.

```
-----see nothing-----------------
 0   0   0  11  11  11   0   0   0  18  18  39  65  65  47   0
33   3   3  15  34  34  22   0   0  33  33  54  80  77  21   0
33  33   5  19  62 108  69  84  51  69  55  55  82  79  23   4
65  33   5  30  73 119  69  84  51  76  62  62  87  83   4   4
35  35   3  27  69  83   9  15  15  34  34  25  54  53   4  13
41  32   0  11  61  61  34  40  15  24  43  34  84  83  13  13
 9   9   0  11 104 104  45  54  30  21  40  28  53  53  13  13
31  31  22  18  71  71  36  65 109  84  41  31  95  95  70  38
22  22  22  40 103 103  83  83 155 171 242  63 105 105 112  80
41  41  41  55  85  85  72  72 163 167  87  82 124 124 106  72
19  19  19  51 119 119 122  60 233 252  19  40  97  79  55  50
19  19  19  31  57  57  75  36 126 252  19  19  77  59  37  13
 0   0   0   7  77  87  Rv  42  80  79  22  43  74  93  87  12
 0   0   0  33  40  86 101  61  79 101  22  41  72  72  72  19
 0   0   0  32  39  91 109  61  20  42  22  38  75 255 255  16
 0   0   0  30  30 111 108  61  20  42  22  37  74  74  74  15
```

Obr. 2.16: Namerané voľné miesta v mriežke

Po ukončení fázy mapovania sa robotu zadal cieľ v mriežke, kde sa následne pomocou algoritmu A* našla cesta ku cieľu a robot sa autonómne dostavil do svojho cieľa pomocou zmapovaných údajov o prekážkach.

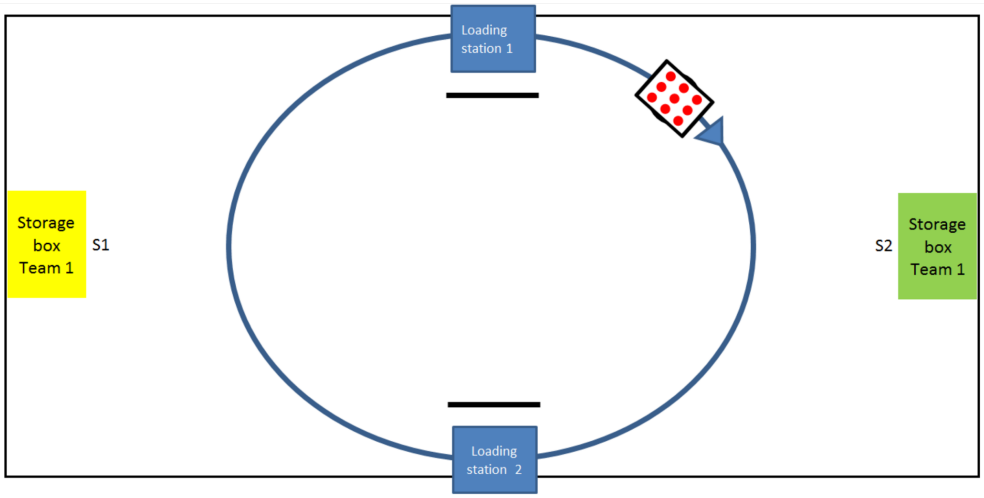
Kapitola 3

SICK robot day 2018

Lokalizácia sa uplatnila na súťaži SICK robot day 2018 v októbri v meste Waldkirch, ktoré sa nachádza na juhozápade Nemecka. Na tejto súťaži sa zúčastnilo dvanásť škôl z Európy.

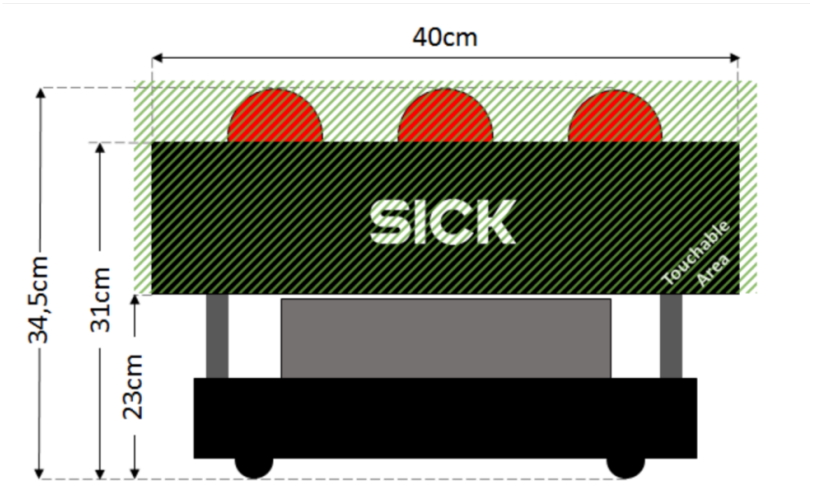
3.1 Ihrisko

Ihrisko je o veľkosti 13 x 7 metrov a okrem nášho robota sa na ňom nachádza transportér, ktorý nesie náklad ťažkých kovových loptičiek a pohybuje sa po elipse okolo stredu ihriska a prechádza cez dve stanice, na ktorých sú následne odobrané loptičky dopĺňané. Ďalej sa tam nachádzajú dve základne, na ktoré je potrebné presunúť kovové loptičky z transportéra. Samozrejme v neposlednej rade súčasťou je aj súper, ktorý sa môže voľne pohybovať po ihrisku.



Obr. 3.1: Nákres ihriska pre súťaž SICK robot day 2018

Našou úlohou je pozbierať, čo najviac loptičiek z transportéra, tak aby sme ho zastavili na čo najkratší čas a následne ich presunuli do našej základne.



Obr. 3.2: Transportér s nákladom

3.2 Pravidlá

Transportér naraz prenáša deväť kovových loptičiek. Tento transportér sa zastaví, akonáhle sa mu nejaký robot stane prekážkou na jeho trase. Robot sa smie dotýkať transportéra len vo vrchnej časti nákladného priestoru. Za každých tridsať sekúnd blokovania sa robotovi odpočítava jeden bod a za každú úspešne odnesenú loptičku získava jeden bod. Robot je diskvalifikovaný v prípade, že súvisle blokuje transportér dlhšie ako dve minúty alebo sa dostane do kolízie so svojím súperom a zároveň nedá prednosť sprava.

Kapitola 4

Návrh riešenia

V tejto kapitole sa mienim venovať návrhu riešenia a objasním jednotlivé rozhodnutia. Kvôli účasti v súťaži "SICK robot day 2018"treba navrhnúť riešenie, ktoré bude dosť efektívne a presné pre obdĺžnikovú miestnosť, v ktorej sa nachádzajú neznáme pohyblivé predmety. Budeme sa spoliehať hlavne na presnosť našej lokalizácie z laserového senzora a za pomoci odometrie sa presúvať medzi jednotlivými bodmi nášho plánu.

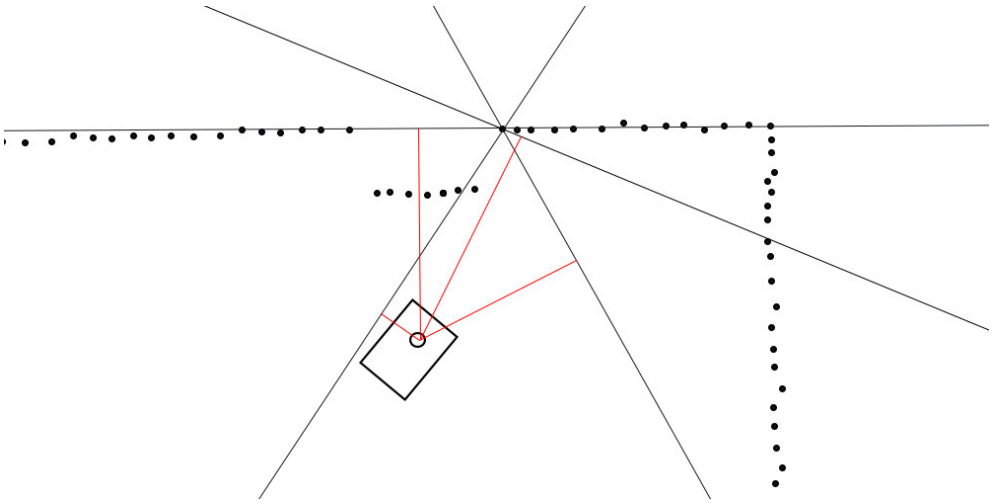
4.1 Modifikácia Houghovej transformácie

Algoritmus Houghovej transformácie si upravíme pre vlastné potreby. Extrakciu hrán môžeme samozrejme preskočiť, keďže jednotlivé lúče laserového senzora pre nás predstavujú jednotlivé hrany získané z obrazu.

Klasický algoritmus porovnáva každý bod hrany s každou potencionálnou priamkou. Veľkosť množiny všetkých potencionálnych priamok označíme N a množiny všetkých získaných bodov M. Zložitosť takéhoto algoritmu je $N*M$.

V našej modifikácii namiesto toho pre každý získaný bod vygenerujeme priamky. Takéto priamky budeme generovať spôsobom, že priamku prechá-

dzajúcu týmto bodom budeme postupne rotovať až o 180°. Každý bod vygeneruje množinu priamok o veľkosti K. Index v matici pre hlasovanie k takýmto priamkam vieme určiť v konštantnom čase. Zložitosť takéhoto algoritmu je potom $M * K$.



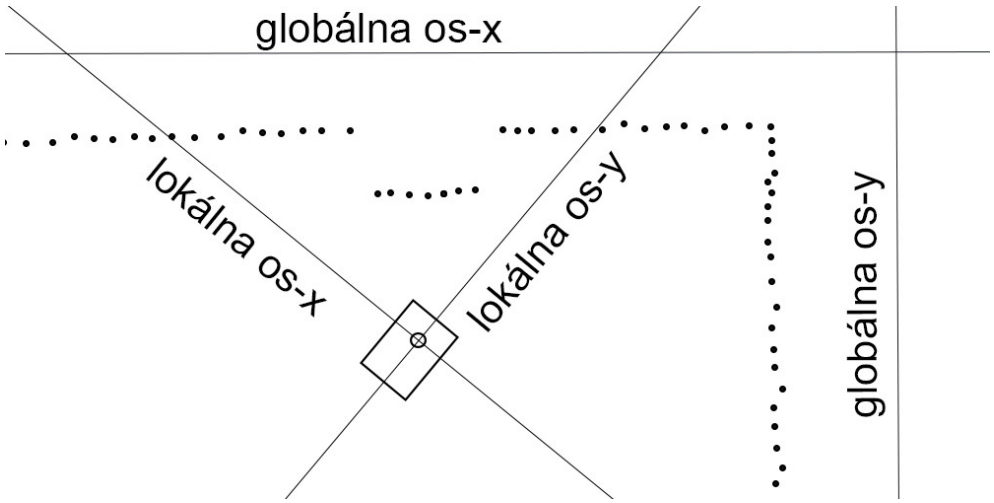
Obr. 4.1: Príklad generovania priamok z bodu získaného pomocou laserového senzoru

Súčasne platí $N > K$. V našom prípade M je presne 811. Ak zoberieme presnosť 15 mm a 1° na vzdialenosť 15 metrov N by bolo rovné 360 000 a K rovné 180. V oboch prípadoch sa použije rovnako veľká matica potencionálnych priamok ale v našom prípade vykonáme $811 * 180$ výpočtov, namiesto $360000 * 811$. Takýto výpočtový rozdiel je pre nás dôležitý.

4.2 Lokalizácia

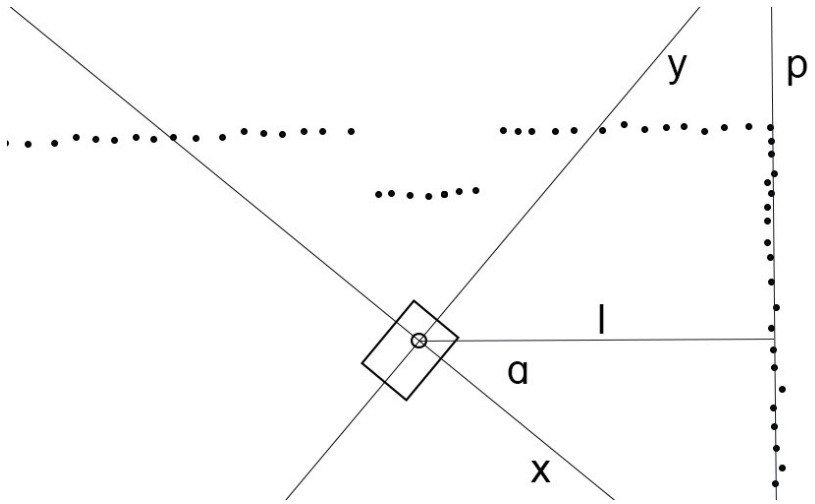
Z dôvodu spoľahlivosti bude jednoznačne hlavným indikátorom polohy laserový senzor. Jeho hodnoty sú často veľmi presné a správne. Robot sa bude na základe jeho údajov rozhodovať, kde sa v priestore aktuálne nachádza. Sen-

zor produkuje 3 hodnoty pre každý stupeň v rozmedzí 270 stupňov. Každý takýto bodový údaj obsahuje informáciu o vzdialenosti a intenzite odrazu.



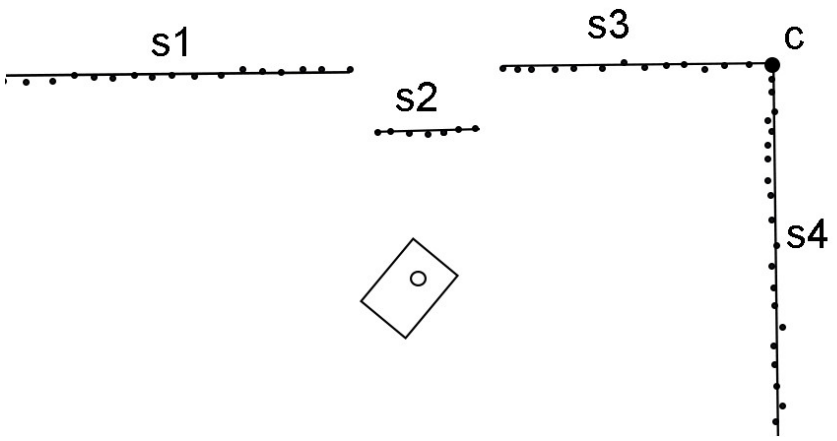
Obr. 4.2: Zobrazenie nasnímaných bodov z laserového senzora vzhľadom na lokálnu a globálnu súradnicovú sústavu

Pomocou Houghovej transformácie zostrojíme priamky.



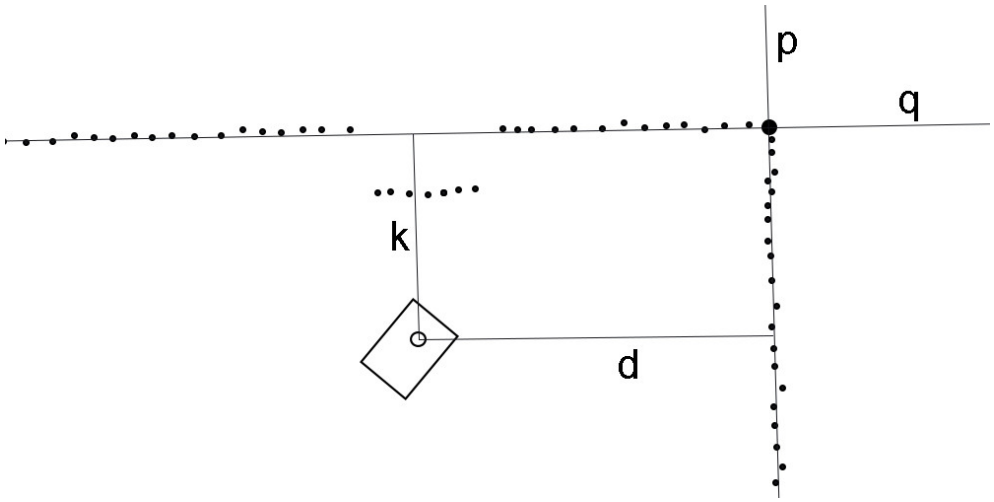
Obr. 4.3: Vzdialenosť a uhol od priamky získané pomocou Houghovej transformácie

Z nájdených priamok a nameraných hodnôt z laserového senzora následne vytvoríme spojité úsečky, medzi ktorými budeme hľadať rohy. V prípade, že budeme mať úspešne získané dve susedné steny a tým pádom roh, ktorý tieto dve steny zovierajú, budeme vedieť pomocou kompasu určiť, o ktorý konkrétny roh sa jedná.



Obr. 4.4: Nájdené úsečky a roh z laserového senzoru

Vďaka tejto informácii a vzdialenosti od jednotlivých segmentov, respektíve stien, budeme vedieť určiť presnú polohu. Otočenie robota určíme taktiež z týchto údajov, pretože kompas je veľmi nepresný a nespoľahlivý. Z toho dôvodu si budeme uchovávať jednotlivé hodnoty získané z Houghovej transformácie vzhľadom na lokálnu sústavu robota. Teda každú potenciálnu priamku budeme mať uloženú ako vzdialenosť od priamky a uhol k najbližšiemu bodu ležiacemu na potenciálnej priamke. Takýmto spôsobom budeme vedieť rýchlo a efektívne určiť presnú vzdialenosť aj otočenie od jednotlivých úsečiek.



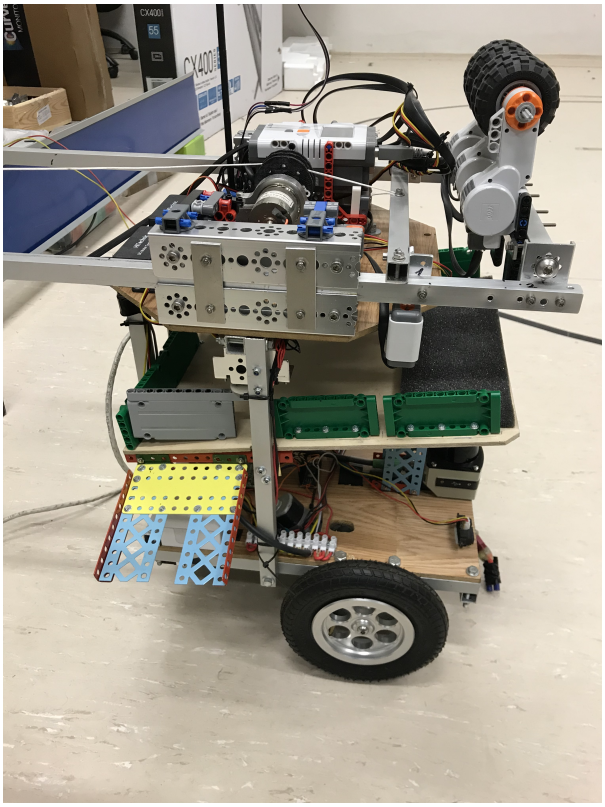
Obr. 4.5: Pomocou pôvodných priamok, ktoré boli použité pri zostrojení úsečiek s3 a s4 vieme určiť presnú polohu a otočenie robota

4.3 Vyhýbanie

Okrem získaných informácii o jednotlivých vzdialených segmentoch, treba z jednotlivých hodnôt z laserového senzora určiť, či sa náhodou pred robotom nenachádza nejaké neznáme teleso a teda neprekáža nášmu robotovi v pohybe. Pre takéto určenie použijeme namerané údaje z laserového senzora a v prípade zvýšeného výskytu zastavených lúčov v blízkosti pred robotom, vyhodnotíme túto situáciu ako neznámu prekážku. Okrem iného použijeme aj údaje zo sonarového senzora, ktoré nám povedia informáciu, či vidí alebo nevidí prekážku v danom rozmedzí. Napriek tomu, že sonarový senzor je veľmi nespoľahlivý, je pre nás nesmierne dôležité neriskovať zrážku s neznámym telesom a preto budeme aj tento údaj považovať za kritický aj v prípade, že laserový senzor nebude nič pozorovať pred sebou. V prípade dlhého blokovania sa skúsime cúvaním vyhnúť danému telesu.

4.4 Nakladanie

Jedným zo zložitých úkonov bude samotné preloženie jednotlivých loptičiek z transportéra na nášho robota Mikeša. Kvôli tejto úlohe je nutné vymyslieť efektívny nakladací systém. Keďže kvalitný a silný magnet je drahý a zároveň by zvýšil komplexitu tohto úkonu, rozhodli sme sa takúto myšlienku zamietnuť. Ďalším nápadom bolo loptičky jednoducho zhrnúť z transportéra do nášho robota, no bohužiaľ kvôli hlbokému vloženiu lôpt do otvorov na transportéri to nebolo možné vykonať. Nakoniec sme sa rozhodli pre systém podobný zhŕňaniu, ale vylepšený o otáčajúce sa gumové kolieska, ktoré nám pomôžu vybrať loptičky zo zapadnutých otvorov.



Obr. 4.6: Mikeš po úpravach

Pre zhrnuté loptičky bolo potrebné vytvoriť úložný priestor a taktiež aj systém pre vyloženie nákladu do základne. Pre zabezpečenie týchto požiadaviek sme sa rozhodli pre šikmú plochu, ktorú umiestnime do stredu robota a následne pomocou vyklápacích dvierok sa loptičky vypustia do našej základne.

4.5 Navigácia

Vzhľadom na očakávanú komplexitu všetkých prvkov dohromady, navigácia bude určovať, kedy sa bude poloha aktualizovať, pretože práve navigácia rozpozná, kedy je robot v pokojovom stave a môže počkať, kým sa údaje obnovia, aby boli aktuálne k danej polohe, čím sa zvýši presnosť merania. Počas pohybu sa bude robot riadiť pomocou odometrie, ktorá je pri krátkych vzdialenostiach na rovnom teréne pomerne presná. Teda pred každým a súčasne po každom vykonanom pohybe sa robot nanovo zlokalizuje. Samozrejme v prípade, že lokalizácia skončí viackrát opakovane neúspešne, robot bude pokračovať s poslednou známou polohou dosiahnutou pomocou odometrie.

4.6 Stratégia

Stratégiu nášho autonómneho robota definujeme ako stavový automat. Robot sa bude postupne presúvať medzi jednotlivými bodmi po ihrisku. Prvou úlohou bude dostať nášho robota na cestu, po ktorej sa bude presúvať transportér. Tu následne náš robot bude čakať, pokiaľ sa transportér nepriblíži na takú vzdialenosť, aby sa zastavil. Následne sa náš robot priblíži k transportéru tak blízko, aby bol schopný naložiť čo najviac loptičiek. Akonáhle sa dokončí proces nakladania loptičiek, alebo vyprší maximálny limit ich

nakladania, použijeme rovnaký vyhýbací manéver, ako pri vyhýbaní sa neznámemu telesu. V prípade úspešného vyhodnotenia naloženia loptičiek sa vrátime s robotom späť na štartovnú polohu, kde náklad vyložíme do našej základne a stavový automat sa presunie na začiatok. V prípade neúspešného naloženia, ktoré môže vzniknúť napríklad aj príliš dlhým čakaním sa vrátime späť na trasu a následne opätovne budeme čakať na príchod transportéra. Vo všetkých stavoch automatu bude zapnutý program na vyhýbanie, okrem stavov, počas ktorých čaká a vykladá z transportéra loptičky, kedy sa očakáva, že vo veľkej blízkosti sa bude nachádzať neznáme teleso, ktoré budeme vyhodnocovať ako transportér. V ostatných prípadoch má najväčšiu kontrolu nad pohybom modul pre vyhýbanie sa neznámym predmetom.

Kapitola 5

Riešenie

V tejto kapitole postupne popíšem architektúru softvéru a jej jednotlivé moduly, ktoré vznikli počas implementácie. Projekt je napísaný v jazyku C z dôvodu jeho rýchlosti. Celý pôvodný projekt sa najmä kvôli udržateľnosti rozdelil do dvoch repozitárov:

- mikes-common
- mikes-generic

Repozitár mikes-common obsahuje všetky spoločné moduly a základnú štruktúru pre manipuláciu s pohybom, získavania jednotlivých údajov zo senzorov a prácu s logovacím systémom. Naopak repozitár mikes-generic obsahuje už jednotlivé projekty, ktoré sú postavené pomocou jednotlivých modulov z repozitára mikes-common. Repozitár mikes-common sa považuje za jadro Mikeša.

5.1 Framework

Pred začiatkom samotnej práce sme sa najskôr rozhodli pre upravenie štruktúry pôvodného kódu. Jednotlivé predchádzajúce projekty a ich súbory boli spoločne uložené v jednej zložke. Samozrejme takto vzniklo veľa súborov, nad ktorými sa začala postupne strácať kontrola. Samozrejme bolo pre mňa zložité sa naučiť v tomto orientovať.

Základom každého nového projektu je konfigurácia modulov. Každý projekt má svoju vlastnú konfiguračnú triedu, ktorá v podstate dedí od matičného projektu. Takýmto spôsobom sa následne dajú vypínať a zapínať jednotlivé moduly. Tieto moduly sa rozdelili na live a passive. Každý modul má svoju dátovú štruktúru. Live moduly sú také, ktoré majú vlastné vlákno v ktorom prebiehajú ich výpočty. Svoje posledné údaje vedia kedykoľvek poskytnúť inému modulu. Alternatívnym spôsobom získavania výsledkov z live modulu je pomocou návrhového vzoru Observer, kedy sa nové údaje posielajú pomocou callbackov v okamih ich dostupnosti. Passive moduly neobsahujú žiadne vlastné vlákno a teda sú používané z iných modulov. Väčšina týchto modulov sú najmä GUI moduly. Súbežne sa pre väčšinu modulov vytvorili aj testy. Takáto prerábka bola zložitá a hlavne časovo náročná. Postupne sme museli upraviť všetky súbory z viacerých projektov a spojiť ich do funkčného a modulárneho celku.

5.2 Lokalizácia

Mnohé moduly lokalizácie boli doplnené do repozitára mikes-common, vzhľadom na ich možnosť opätovného použitia. Každý takýto modul má svojich subscriberov, ktorí čakajú na jednotlivé vypočítané hodnoty daného modulu. Súčasne tieto moduly obsahujú vlastný thread, v ktorom bežia jednotlivé vý-

počty. Ku každému takémuto modulu vznikla aj zvlášť abstraktná časť pre výpočty, tak aby sa odstránila závislosť od konkrétneho laserového senzora.

Následne teda vznikli moduly:

- tim571
- tim_hough_transform
- line_filter
- tim_segment
- tim_corner
- sick_localization

Všetky moduly okrem sick_localization modulu boli pridané do repozitára mikes-common. K viacerým vznikli aj už spomínané pomocné časti pre opakované použitie pre iný senzor ako tim571:

- math_2d
- hough
- filter
- segment
- corner

K jednotlivým pomocným modulom sme pridali aj testy, aby boli spoľahlivejšie a dlhodobo udržateľnejšie:

- test_math_2d

- test_hough
- hough_tests
- test_filter

5.2.1 Modul math_2d

Vzhľadom na časté matematické operácie v dvojrozmernom priestore sme vytvorili tento modul pre uľahčenie bežných výpočtov.

```
void angle_to_vector(double *angle, vector_2d *v);

void get_normal_vector(vector_2d *v, vector_2d *normal);

void vector_and_distance_to_point(vector_2d *v, double *distance, point_2d *p);

void get_line_data_from_distance_and_angle(double *dist, double *ang,
  vector_2d *v, vector_2d *line_vector, point_2d *line_point);

void rotate_vector_by_angle(vector_2d *line, int *angle, vector_2d *result);

double get_vector_length(vector_2d *v);

void normalize_vector(vector_2d *v);

double get_offset_from_vector_and_point(vector_2d *normal_v, point_2d *p);

double angle_between_vectors(vector_2d *v1, vector_2d *v2);

double angle_from_axis_x(vector_2d *v);

void vector_from_two_points(point_2d *p1, point_2d *p2, vector_2d *v);

void find_cross_of_two_lines(vector_2d *normal_v1, point_2d *p1,
  vector_2d *normal_v2, point_2d *p2, point_2d *result);

void find_distance_and_angle_between_point_and_line(point_2d *point,
  vector_2d *line_directional, point_2d *line_p, double *distance, double *angle);
```

Obr. 5.1: Rozhranie math_2d

Takisto sme vytvorili štruktúry pre definovanie priamky, ako uhol a vzdialenosť k najbližšiemu bodu ležiacemu na priamke. Takáto definícia priamky

sa neskôr zíde pre jednoduchšie a efektívnejšie určenie polohy. Tento jednoduchý modul obsahuje aj sadu testov pre jeho spoľahlivosť a opakované použitie. Tento modul sa na ďalej bude využívať aj v nasledujúcich projektoch a takisto dopĺňať o nové funkcionality.

5.2.2 Modul tim571

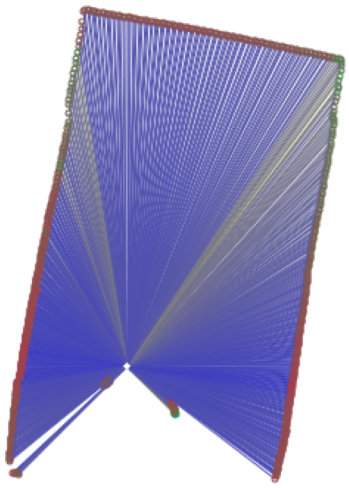
Modul tim571 slúži na spracovanie údajov z laserového senzora a následne si ich ukladá do dátovej vlastnej štruktúry a poskytuje tieto údaje svojim subscriberom. Laserový senzor poskytuje frekvenciu 15 Hz, čo je viac než postačujúce. Na začiatku každého cyklu obsahuje senzor okrem údajov o vzdialenosti a sile jednotlivých lúčov, tiež rôzne iné dôležité informácie, napríklad informácie o verzii softvéru. So senzorom sa komunikuje obojsmerne pomocou sieťovej vrstvy.

```
// status information received from the sensor (typical values in comments)
typedef struct tim571_status_struct {
  uint16_t firmware_version;     // 1
  uint16_t sopas_device_id;      // configurable in SOPAS
  uint32_t serial_number;
  uint8_t error;                 // 0 = device is ready
  uint32_t scanning_frequency;   // 1500  in 1/100 Hz
  float multiplier;              // 1
  int32_t starting_angle;        // -450000 (in 1/10000 of degree)
  uint32_t angular_step;         // 3333    (in 1/10000 of degree)
  uint16_t data_count;           // number of distance/rssi data
  uint8_t rssi_available;        // 0 = not receiving rssi data
  char name[17];                 // name configurable in SOPAS or "\0"
} tim571_status_data;

typedef void (*tim571_receive_data_callback)(uint16_t *, uint8_t *, tim571_status_data *status_data);
```

Obr. 5.2: Dátové štruktúry modulu tim571

Tim571 vieme spustiť tak, aby opakovane posielal namerané údaje vo svojej frekvencii 15 Hz alebo si vieme vyžiadať len jednu vzorku. Rozhodli sme sa pre zasielanie údajov v opakovanej frekvencii. Toto rozhodnutie sme urobili pre možnosť zapojenia aj iných modulov, ktoré sa vedia tiež subscribnuť

a počúvať, no majú inú funkčnú logiku. Taktiež je možné získať poslednú prečítanú vzorku aj bez subscribnutia.



Obr. 5.3: Body získané pomocou modulu tim571. Zelené body a sivé lúče zodpovedajú nižšej intenzite a týkajú sa tmavších stien

5.2.3 Modul hough

Tento modul slúži na rozpoznanie jednotlivých potencionálnych priamok z jednotlivých lúčov ktoré dostáva z modulu tim571. Tento modul obsahuje vlastnú konfiguráciu ktorá vstupuje ako parameter do výpočtovej funkcie. Tým sa dá dosiahnuť, že viac rôznych modulov vie využívať výpočet súčasne so svojimi vlastnými konfiguračnými parametrami. V tejto konfigurácii sa dá nastaviť maximálna vzdialenosť, presnosť pre vzdialenosť a uhol, minimálny počet hlasov a minimálne hodnoty pre vstupnú vzdialenosť a RSSI jednotlivých lúčov z laserového senzora.

```
typedef struct line_struct {
  angle_line_2d line;
  int votes;
} line_data;

typedef struct lines_struct {
  line_data lines[LINE_MAX_DATA_COUNT];
  int count;
} lines_data;

typedef struct hough_config_struct {
  int distance_max; // 15000
  int distance_step; // 15

  int angle_step; // 5

  int votes_min; // 10

  int bad_distance; // 0
  int bad_rssi; // 0
} hough_config;
```

Obr. 5.4: Datové štruktúry Houghovej transformácie

Podľa vstupnej konfigurácie vytvorí pole pre každú potenciálnu priamku po maximálnu vzdialenosť. Celková maximálna vzdialenosť sa vydelí veľkosťou zrnitosti vzdialenosti a následne sa vynásobí množstvom zrnitosti uhla. Tým vznikne veľkosť jednorosmerného poľa všetkých možných priamok podľa zadanej konfigurácie.

```
int get_votes_array_size(hough_config *config)
{
  return get_distance_step_count(config) * get_angle_step_count(config);
}
```

Obr. 5.5: Množstvo potencionálnych priamok

Následne pre každý lúč senzora sa skontroluje, či je vhodným kandidátom vzhľadom na jeho nameranú vzdialenosť a hodnotu RSSI. Následne určíme vektor prechádzajúci týmto bodom a súčasne kolmý na senzor. Po tom, ako máme našu prvú potenciálnu priamku prechádzajúcu týmto lúčom, budeme

ju postupne rotovať až o 180°, keďže po tejto hodnote sa jedná už o rovnaké vektory aké sme už použili, len smerujúce opačným smerom a takéto duplicitné hodnoty nás nezaujímajú. Následne z nových potencionálnych priamok, ktoré sú tvorené bodom a vektorom, získame vzdialenosti a uhly od senzora. Pomocou týchto hodnôt zistíme prislúchajúci index v poli a za jednotlivé hodnoty zahlasujeme.

```
for (int index = 0; index < status_data->data_count; index++) {
  if (is_good_value(config, distance[index], rssi[index])) {
    double dist = distance[index];
    double ang = get_angle(status_data, index);
    vector_2d v, line_vector;
    point_2d line_point;

    get_line_data_from_distance_and_angle(&dist, &ang, &v, &line_vector, &line_point);

    double distRes;
    double angRes;
    vector_2d result_vector;

    for (int angle = 0; angle < HALF_ANGLE; angle++) {
      rotate_vector_by_angle(&line_vector, &angle, &result_vector);
      find_distance_and_angle_between_point_and_line(&entry, &result_vector, &line_point, &distRes, &angRes);

      int voteIndex = get_index_from_distance_and_angle(config, distRes, angRes);
      if (voteIndex >= 0) {
        votes[voteIndex]++;
      }
    }
  }
}
```

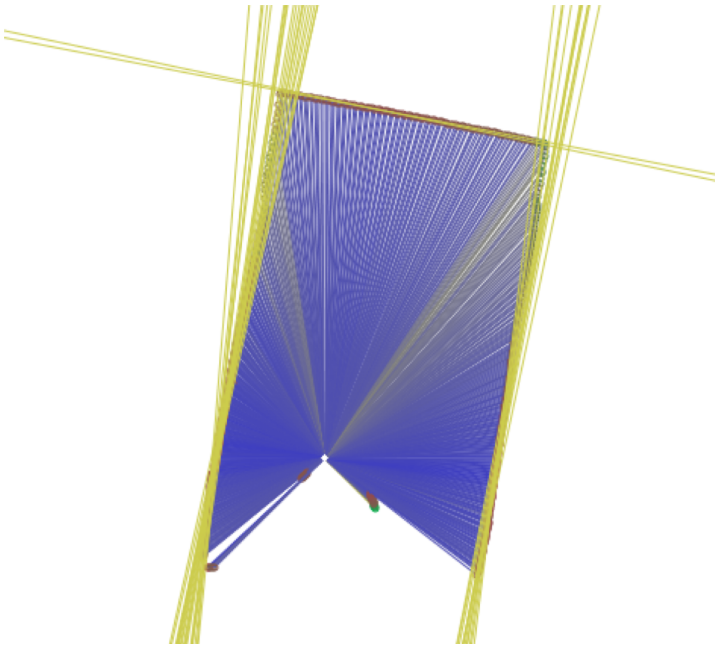
Obr. 5.6: Hlasovanie za jednotlivé potencionálne priamky

Následne zozbierame všetkých vhodných kandidátov, ktorí spĺňajú kritériá vstupnej konfigurácie. Tieto údaje zotriedime podľa počtu hlasov a následne vyberieme najlepšie výsledky aj s ich počtom ohodnotení.

```
// Take all good votes
line_data vote_results[size];
int result_count = 0;

for (int vote_index = 0; vote_index < size; vote_index++) {
  if (is_good_candidat(config, votes[vote_index])) {
    line_data line;

    get_line_data_from_index(config, vote_index, &line);
    line.votes = votes[vote_index];

    vote_results[result_count++] = line;
  }
}

// Sort MAXIMUM_POSSIBLES votes
qsort(vote_results, result_count, sizeof(line_data), line_comparator);

// Take first best fit LINE_MAX_DATA_COUNT from MAXIMUM_POSSIBLES
for (int index = 0; index < LINE_MAX_DATA_COUNT && index < result_count; index++) {
  data->lines[index] = vote_results[index];
}
data->count = result_count < LINE_MAX_DATA_COUNT ? result_count : LINE_MAX_DATA_COUNT;
```

Obr. 5.7: Vyberanie najlepších výsledkov priamok

Takýto výsledok z Houghovej transformácie ešte nie je dobre použiteľný. Napríklad, ak náš robot vidí tri steny naraz, neznamená to, že prvé tri výsledky z Houghovej transformácie budú práve tieto tri steny. V prípade, ak jedna stena je oveľa dlhšia ako zvyšné dve steny, tak takáto stena môže vyprodukovať viac priamok, ktoré budú ohodnotené silnejšie ako najlepšie ohodnotené priamky zvyšných dvoch stien. Okrem toho, najlepšia priamka aj najdlhšej steny nemusí byť úplne presná v súlade s reálnou stenou. Takýto jav vzniká najmä kvôli veľkosti jednotlivých krokov vzdialenosti a uhlu zo vstupnej konfigurácie.



Obr. 5.8: Množina priamok z Houghovej transformácie

Takáto veľká množina je, ale aj dôsledkom nerovného povrchu. V skutočnosti sa nejedná o tri steny, ale o ihrisko poskladané z rôznych predmetov v tvare obdĺžnika, v ktorom sa robot práve nachádza.

5.2.4 Modul filter

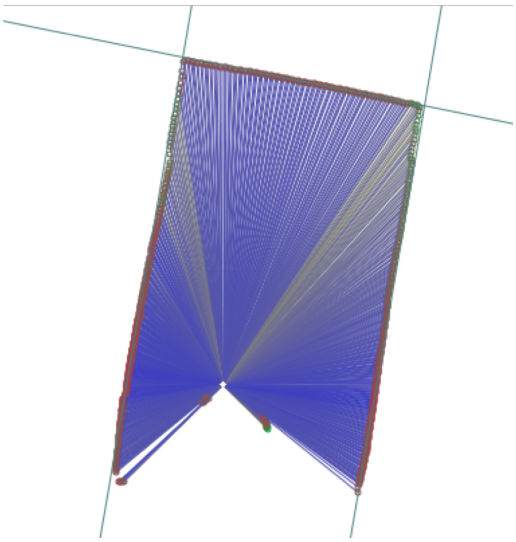
Kvôli veľkému množstvu podobných údajov z Houghovej transformácie vznikol modul Filter, ktorý podobné priamky spojí a vytvorí z nich jednu priamku, ktorá sa nemusí nachádzať ani v množine pôvodných priamok z Houghovej transformácie.

Jednotlivé priamky najskôr usporiadame podľa vzdialeností a následne každej priamke pridelíme vlastnú skupinu. Potom jednotlivé skupiny postupne zlučujeme, v prípade, že ich vzdialenosť aj uhol sú dostatočne podobné.

```
for (int i = 0; i < lines_count; i++) {
  int maxdist = original_lines->lines[i].line.distance
   + LINES_SAME_CLUSTER_MAX_DISTANCE_DIFFERENCE;
  short alpha = original_lines->lines[i].line.angle;

  for (int j = i + 1; (j < lines_count) &&
      (original_lines->lines[j].line.distance <= maxdist); j++) {
    short beta = original_lines->lines[j].line.angle;
    if (abs(angle_difference(alpha, beta))
      < LINES_SAME_CLUSTER_MAX_ANGLE_DIFFERENCE) {
      if (clusters[i] != clusters[j]) {
        if (clusters[i] < clusters[j]) {
          for (int k = 0; k < lines_count; k++) {
            if (clusters[k] == clusters[j]) clusters[k] = clusters[i];
          }
        } else {
          for (int k = 0; k < lines_count; k++) {
            if (clusters[k] == clusters[i]) clusters[k] = clusters[j];
          }
        }
      }
    }
  }
}
```

Obr. 5.9: Vytváranie jednotlivých spoločných skupín

Následne potrebujeme vytvoriť priemerné hodnoty pre jednotlivé skupiny pôvodných priamok z Houghovej transformácie. Každá priamka z jednotlivej skupiny zahlasuje svojou váhou podľa počtu z Houghovej transformácie svojou dĺžkou a uhlom. Takýmto spôsobom vznikne vážený priemer dĺžky a uhla jednotlivých skupín.



Obr. 5.10: Nové spriemerované priamky z Houghovej transformácie

5.2.5 Modul segment

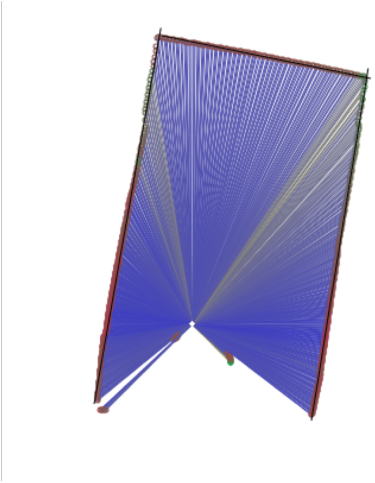
V tomto module sa jedná o zostrojenie jednotlivých ucelených celkov. Na jednej priamka sa samozrejme môže nachádzať viacero segmentov, čo môže vzniknúť napríklad vďaka predmetom, ktoré sa môžu nachádzať pred danou priamkou.

Pre každú získanú priamku z modulu filter a pôvodných lúčov z laserového senzora tim571 postupne získavame ucelené celky s určitou toleranciou vzdialenosti jednotlivých lúčov od priamky. Zároveň predpokladáme, že niektoré lúče môžu byť chybné alebo malý predmet môže zakrývať výhľad a tak sa môžu nachádzať príliš ďaleko od priamky. Preto je možné nastaviť tolerancie jednotlivých faktorov, ktoré môžu ovplyvniť výsledok segmentácie.

```
typedef struct segment_config_struct {
  double max_distance_error; // 10.0
  int min_points_segment; // 5
  int max_points_skip; // 5
  uint8_t bad_rssi; // 0
} segment_config;
```

Obr. 5.11: Dátová štruktúra pre konfiguráciu výpočtu segmentov

Tieto hodnoty sú veľmi dôležité ak vieme že sa nachádzame v skoro prázdnej miestnosti s dlhými stenami je rozumné nastaviť max_points_skip na vysokú hodnotu a tým pádom vieme v podstate vidieť cez prekážky, ktoré nám tento celok rozdeľuje na viac časti. Samozrejme treba byť súčasne opatrný.

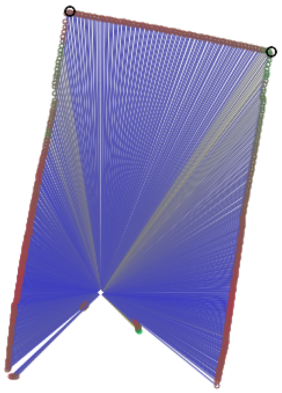


Obr. 5.12: Vytvorené segmenty pomocou priamok a bodov sú znázornené čiernými čiarami

5.2.6 Modul corner

Postupne od najväčších úsečiek hľadáme rohy. Z úsečiek si zostrojíme opätovne priamky. Následne zistíme uhol, ktoré tieto dve priamky zovierajú a ich priesečník. Skontrolujeme vypočítaný uhol, či je pre nás dostatočne pravouhlý

a súbežne skontrolujeme vzdialenosť priesečníka od jednotlivých úsečiek.



Obr. 5.13: Nájdené rohy z jednotlivých segmentov

5.2.7 Modul sick_localization

V tomto module využijeme výpočet z modulu corner, ktorý obsahuje jednotlivé nájdené rohy a súčasne segmenty, ktorými sú tvorené. Okrem toho použijeme hodnotu kompasu z base modulu. Vzhľadom na nepresnosť kompasu jeho informáciu použijeme len na definovanie, o ktorý konkrétny roh sa jedná. Následne pomocou jednotlivých segmentov vieme určiť vzdialenosť od konkrétnych stien na mape, čím dostávame aktuálnu pozíciu na mape. Otočenie robota získame následne podľa už známej polohy a známeho uhla ku konkrétnemu rohu.

```
int get_pose_base_on_corners_and_heading(corners_data *corners,
  base_data_type *base_data, pose_type *result_pose)
{
  for (int c_index = 0; c_index < corners->count; c_index++) {
    corner_data *corner = &corners->corners[c_index];
    angle_line_2d line_left, line_right;
    double difference_x, difference_y;

    int corner_index = get_corner_index(corner, base_data->heading);
    get_left_and_right_line_from_corner(corner, &line_left, &line_right);
    get_difference_x_and_y(&corner_index, &line_left, &line_right, &difference_x, &difference_y);
    get_map_x_and_y(&corner_index, &difference_x, &difference_y, &result_pose->x, &result_pose->y);
    get_heading(corner, &corner_index, &result_pose->x, &result_pose->y, &result_pose->heading);
    if (result_pose->x > 0 && result_pose->x < SICK_MAP_WITH_IN_MM &&
      result_pose->y > 0  && result_pose->y < SICK_MAP_HEIGHT_IN_MM) {
      return SICK_LOCALIZATION_SUCCESS;
    }
  }
  return SICK_LOCALIZATION_FAIL;
}
```

Obr. 5.14: Získanie lokalizácie pomocou rohov a base modulu

Takáto lokalizácia samozrejme môže skončiť aj neúspešne kvôli rôznym procesom, ktoré sa vykonávajú v predošlých moduloch alebo jednoducho nemusí mať výhľad na ani jeden roh v miestnosti. Takáto situácia vie nastať, napríklad ak sa robot pozerá priamo do rohu a pred ním sa súčasne nachádza prekážka, respektíve súper. S takouto situáciou sa samozrejme musí vedieť vysporiadať nasledujúci modul, ktorý spracúva výsledky z lokalizácie.

5.3 Stratégia

Stratégia obsahuje len jeden modul, ktorý zabezpečuje celé spracovanie z jednotlivých modulov a rozhoduje, čo sa má následne vykonať.

5.3.1 Modul sick_strategy

Tento modul je hlavným modulom celej aplikácie. Všetky dôležité rozhodnutia sa vykonávajú v jeho stavovom automate. Do niektorých stavov sa vie

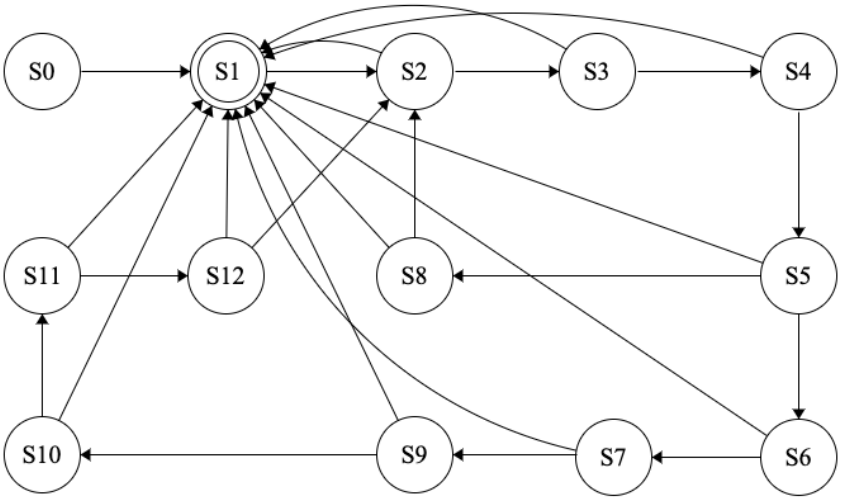
dostať z viacerých rôznych stavov, ale po vykonaní jednotlivého stavu sa rozchádzajú, a preto okrem aktuálneho stavu si robot udržiava aj predchádzajúci stav.

```
#define SICK_STRATEGY_STATE_NONE 0
#define SICK_STRATEGY_STATE_STANDBY 1
#define SICK_STRATEGY_STATE_MOVING1_TO_CART 2
#define SICK_STRATEGY_STATE_MOVING2_TO_CART 3
#define SICK_STRATEGY_STATE_WAITING_CART 4
#define SICK_STRATEGY_STATE_ALIGN 5
#define SICK_STRATEGY_STATE_GRABBING 6
#define SICK_STRATEGY_STATE_LOADED_ESCAPE 7
#define SICK_STRATEGY_STATE_NOT_LOADED_ESCAPE 8
#define SICK_STRATEGY_STATE_RETURNING1 9
#define SICK_STRATEGY_STATE_RETURNING2 10
#define SICK_STRATEGY_STATE_RETURNING3 11
#define SICK_STRATEGY_STATE_RELEASING 12
```

Obr. 5.15: Jednotlivé stavy strategického modulu

Robot je v stave STANDBY, pokiaľ sa nespustí hra. Po spustení hry sa zapne modul na vyhýbanie a z tohoto stavu sa prepne do stavu, kedy sa začne presúvať pomocou navigačného modulu k svojej prvej destinácii. Pohyb smerom k bodu na okruhu, kde budeme čakať na transportér je rozdelený na dve časti pre lepšie a presnejšie nasmerovanie priamo k transportéru, keďže počas pohybu sa lokalizuje pomocou odometrie, ktorá je menej presná ako lokalizačný modul pre súťaž SICK. Následne po dorazení do svojej destinácie sa prepne do stavu, kedy čaká na príchod transportéra a po jeho zastavení sa priblíži na takú vzdialenosť, aby sme vedeli z neho zozbierať, čo najväčšie množstvo loptičiek. V prípade neúspešného priblíženia sa robot prepne do stavu, aby sa vyhol transportéru a neblokoval ho, za čo by sme mohli dostať mínusové body. Potom sa vráti do pôvodného stavu, aby znovu počkal na príchod transportéra. Naopak v prípade úspešného naštelovania k trasportéru prejde do stavu naloženia nákladu, kde sa postupne pokúsi zozbierať všetky tri stĺpce loptičiek. Po ukončení nakladania sa robot prepne do stavu

vyhýbania a následne na tri kroky stavového automatu sa dostane k svojej odkladacej krabici. Potom Mikeš vytrasie loptičky pomocou motorov, pretože sklon zásobníka nie je dostatočne veľký, kvôli obmedzenému priestoru. Po ukončení vykladania sa vráti do pôvodného stavu ako pri spustení hry a pokračuje ďalej v hre. Modul pre vyhýbanie neznámym predmetom je obmedzený počas nakladania, vykladania a čakania na príchod transportéra, kedy sa v blízkej vzdialenosti očakáva neznámy predmet. Samozrejme z každého stavu sa vie dostať do stavu STANDBY z dôvodu vypršania herného času.



Obr. 5.16: Stavový automat stratégie

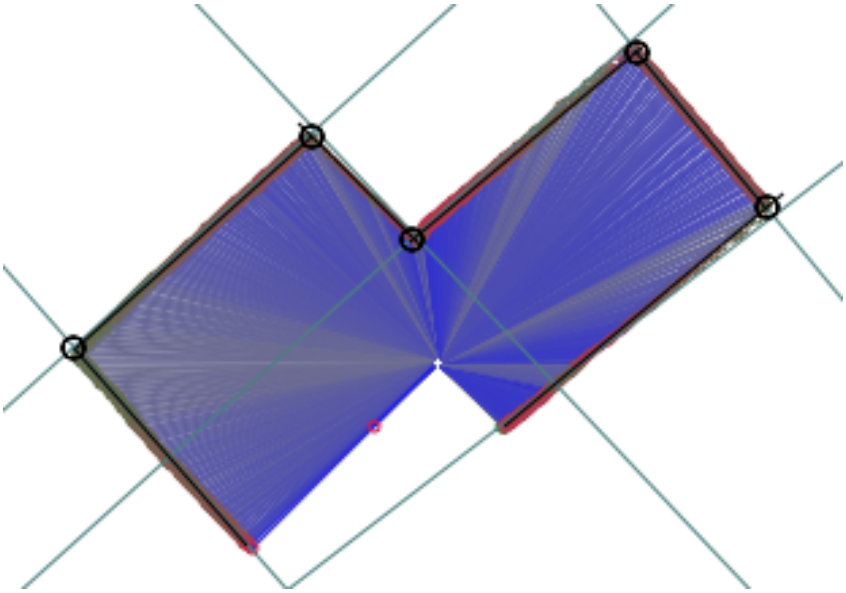
5.4 Vylepšenia lokalizácie

Ako prvé bolo riešenie pre súťaž SICK robot day 2018 upravené tak, aby sa odstránili jednotlivé prepojenia medzi modulmi sick a lokalizáciou v obdĺžnikovej mape. Pomocou modifikácie modulu sick_localization vznikol modul rect_localization. Tento modul rieši lokalizáciu obdobne, ako modul sick_localization, ale nie je už nepriamo ovládaný pomocou navigačného

modulu, ktorý pomocou modulu tim571 vypína a zapína údaje z laserového senzora, potrebné pre jeho činnosť. Týmto vznikol univerzálny modul pre lokalizáciu v obdĺžniku.

5.4.1 Lokalizácia v polygóne

Pre určenie polohy v polygóne bolo potrebné navrhnúť nový univerzálnejší spôsob lokalizácie. Keďže tu nevieme jednoznačne určiť pomocou kompasu, ktorý konkrétny roh náš senzor spozoroval. Pre tento účel vznikol nový modul pol_localization.



Obr. 5.17: Zosnímanie miestnosti v tvare L

Museli sme upraviť najskôr citlivosť jednotlivých modulov, ktoré boli optimalizované pre hľadanie obdĺžnika s dlhými rovnými stenami, tak aby dokázal rozpoznať aj kratšie úseky. Takáto citlivosť spôsobila, že sa občas vyskytne priamka ktorá prechádza, kadiaľ reálne skoro nič nevedie. Tieto chyby sa ale odfiltrujú ďalšími krokmi, získavaním jednotlivých úsečiek a rohov a

preto nerobia starosti.

Po získaní údajov z modulu segment a corner sa vyberú všetky úsečky, ktoré sú z oboch strán uzatvorené rohmi. Takéto úsečky následne predstavujú zatiaľ neznáme steny nachádzajúce sa v polygóne načítanej mapy. Obe reprezentácie úsečiek, namerané a načítané, sa udržujú v rovnakom smere, tak aby sa vedeli navzájom namapovať.

Z úsečiek laserového senzora sa následne vytvoria spojené ucelené celky. Takéto udržanie celkov po kope zaručí presnejší výsledok mapovania.

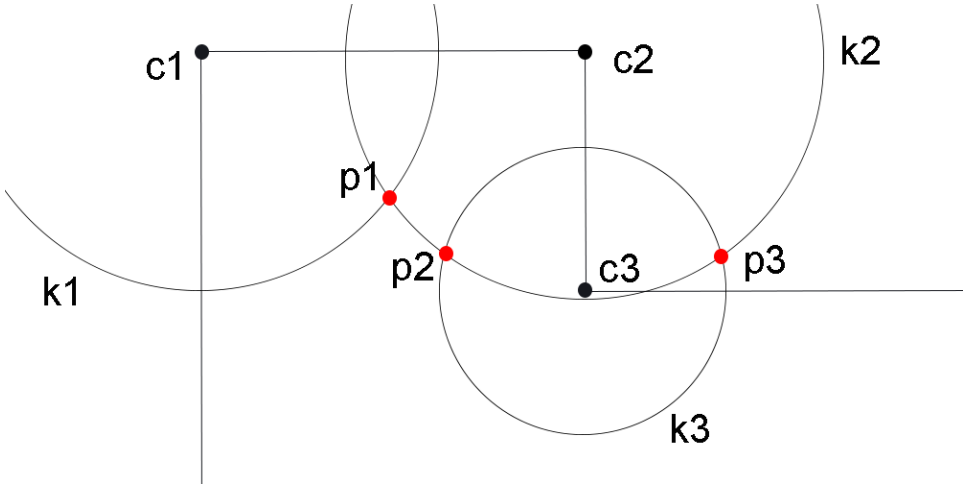
Proces mapovania prebieha tak, že sa vygenerujú postupne všetky možnosti s dodržaním poradia. Tento problém riešime ako kombináciu vkladania voľných miest pred jednotlivé nájdené ucelené celky z laserového senzora.

```
int best_combination_i = -1;
int best_start = -1;
double best_combination_difference = INFINITY;

for (int index = 0; index < numberOfCombinations; index++) {
  for (int start = 0; start < numberOfItems; start++) {
    double difference = get_difference_of_combination(&found_segments,
      combined_segments, combinations[index], numberOfCombinedSegments, start);
    if (best_combination_difference > difference) {
      best_combination_difference = difference;
      best_combination_i = index;
      best_start = start;
    }
  }
}
```

Obr. 5.18: Hľadanie najlepšej kombinácie namapovania segmentov

Ako najlepšiu kombináciu vyhodnotíme kombináciu s najmenším súčtom rozdielov vzdialeností medzi jednotlivými porovnaniami segmentu a steny polygónu danej kombinácie. Potom, ako sú popárované steny a segmenty, pomocou údajov zo senzora, máme konkrétne vzdialenosti k daným rohom. Keďže máme pre každú priradenú stenu, už aj známe vzdialenosti od ro-

hov, vytvoríme kružnice okolo vrcholov a nájdeme priesečníky týchto kružníc. Každá stena nám vyprodukuje dve možné pozície. Potom odstránime všetky body, ktoré ležia mimo nášho polygónu. Z jednoznačne určených bodov vytvoríme spriemerovanú pozíciu, pomocou ktorej sa rozhodneme medzi zvyšnými dvojicami. Všetky vybrané hodnoty použijeme na určenie finálnej polohy.



Obr. 5.19: Príklad úspešného rozpoznania a namapovania troch rohov

Na obrázku 5.19 je vidno úspešne rozpoznanie a priradenie troch vrcholov do mapy. Za pomoci nameraných vzdialeností sa vytvoria kružnice $k1$, $k2$ a $k3$ z vrcholov $c1$, $c2$ a $c3$. Priesečník $p3$ bude odstránený, pretože sa nenachádza v polygóne a výsledná poloha bude určená medzi priesečníkmi $p1$ a $p2$.

Kapitola 6

Výsledky

6.1 Obdĺžniková lokalizácia

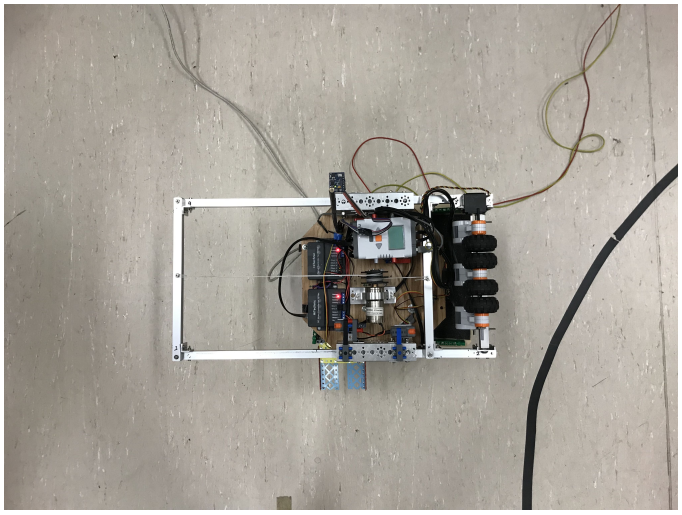
Vytvorili sme si provizórne ihrisko o šírke 5,8 metra a dĺžke 3,2 metra, na ktorom sme vykonávali testy. Umiestnili sme robota do ihriska a spustili sme program na lokalizáciu.

```
difference_x: 4271.0791 difference_y: 2293.5096 heading: 87.429
difference_x: 4274.2391 difference_y: 2293.3090 heading: 87.439
difference_x: 4280.0000 difference_y: 2293.5476 heading: 87.478
difference_x: 4273.8537 difference_y: 2293.7019 heading: 87.454
difference_x: 4273.9891 difference_y: 2293.1591 heading: 87.454
difference_x: 4271.0000 difference_y: 2292.7360 heading: 87.427
difference_x: 4274.0217 difference_y: 2293.1333 heading: 87.449
difference_x: 4274.0426 difference_y: 2293.5437 heading: 87.449
difference_x: 4274.5455 difference_y: 2293.7815 heading: 87.457
difference_x: 4280.0000 difference_y: 2293.3333 heading: 87.395
```

Obr. 6.1: Výstup lokalizácie z desiatich meraní

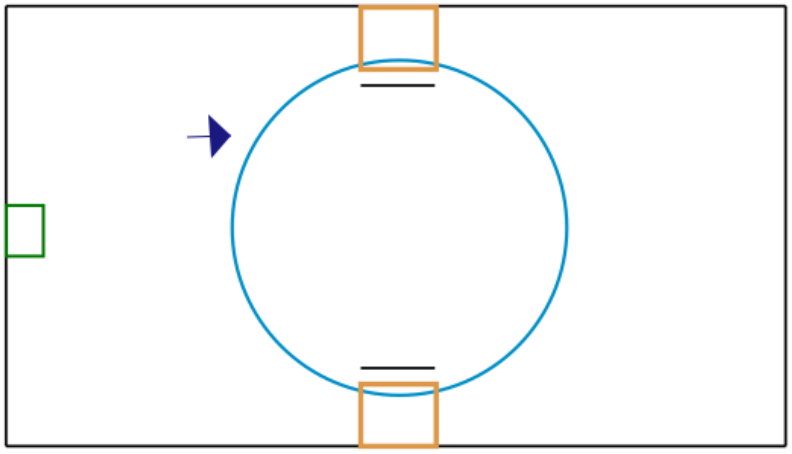
Následne sme jednotlivé hodnoty odmerali pomocou meracieho pásma. Namerané hodnoty boli veľmi podobné. Vzdialenosť od steny v smere x-ovej osi bola 4 279 milimetrov a v smere y-novej osi 2 293 milimetrov. Odchýlka pri vzdialenosti 4 metrov bola približne 10 milimetrov. Takýto výsledok je

výborný vzhľadom na štatistickú chybu laserového senzora tim571, ktorej hodnota je uvedená 20 milimetrov.



Obr. 6.2: Skutočná pozícia robota v miestnosti

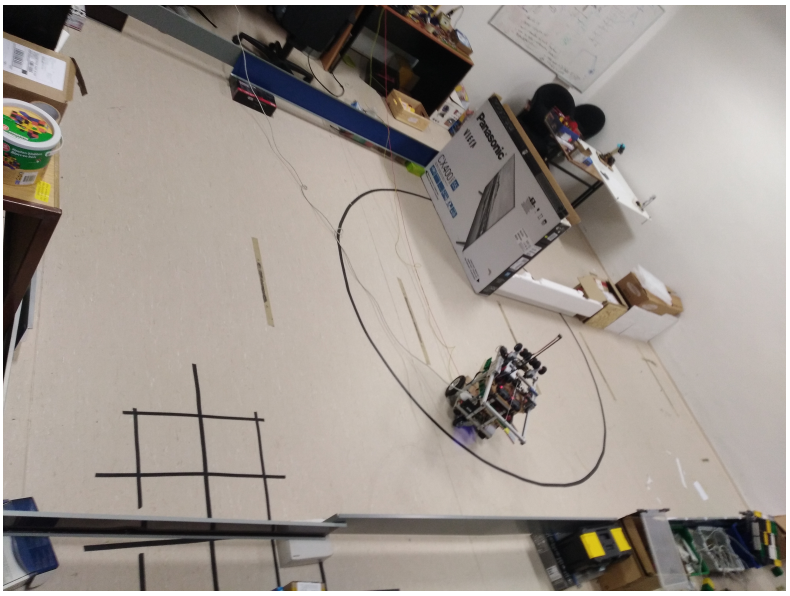
Skutočná pozícia robota je následne aj správne rozpoznaná za pomoci kompasu a zaznamenaná do mapy.



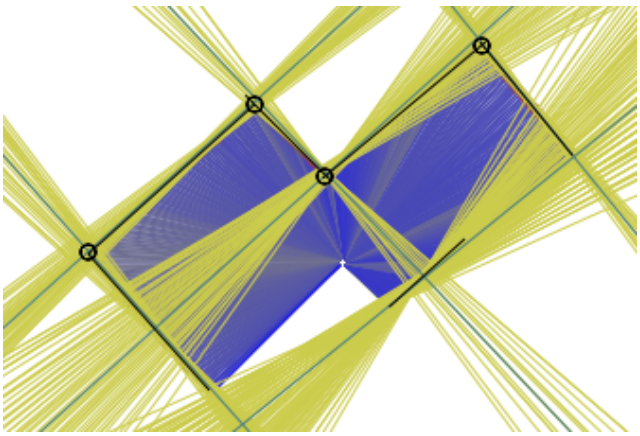
Obr. 6.3: Zobrazenie pozície robota do mapy

6.2 Lokalizácia v polygóne

Podobne, ako pri predchádzajúcom prípade sme vytvorili z dostupných predmetov ohraničenú plochu v tvare L, na ktorej sme testovali lokalizáciu v rámci polygónu.

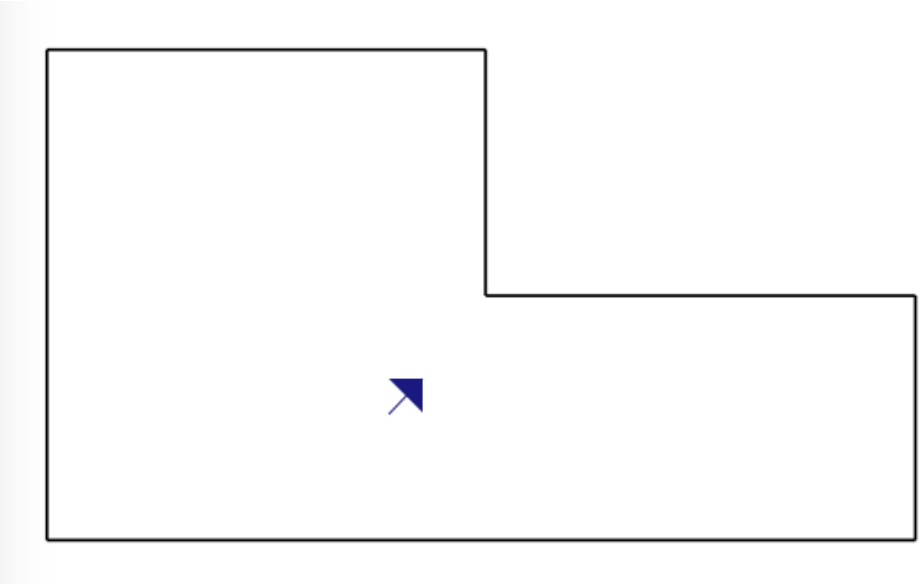


Obr. 6.4: Skutočná poloha robota v polygóne



Obr. 6.5: Zobrazenie spracovaných údajov polygónu z laserového senzora

Na spodnej časti obrázku 6.5 je možné spozorovať segment, ktorý nebol úspešne rozpoznaný v plnej šírke. Bolo to spôsobené zlým povrchom a nerovnosťou použitých materiálov na zostrojenie miestnosti. Rovnako ako pri predchádzajúcom riešení sa robot úspešne lokalizoval.



Obr. 6.6: Zobrazenie pozície robota do mapy polygónu

6.3 SICK robot day 2018

Z celkového počtu dvanástich tímov sme sa umiestnili na štvrtom mieste. Hoci sa nám nepodarilo zozbierať ani jednu loptičku, ale hlavne sme nezískali žiadne mínusové body. Náš robot mal v porovnaní s ostatnými súpermi skvelú lokalizáciu a techniku na nakladanie a vykladanie loptičiek. Náš zberný systém vedel pozbierať naraz bez menších problémov až šesť loptičiek. Výsuvné rameno bohužiaľ nebolo dostatočne dlhé na pozbieranie loptičiek z posledného radu, kvôli limitu rozmerov na 60 centimetrov. Tento limit sme striktne dodržali, na rozdiel od niektorých súperov, aj keď sme to veľmi po-

trebovali. Lokalizačný aj navigačný systém fungoval veľmi účinne a preto sa náš robot Mikeš vždy dostal, takmer na centimeter presne, kam potreboval.

Bohužiaľ transportér nebol zostrojený presne podľa špecifikácie. V úrovni výšky nášho laserového senzora mal transportér namiesto pevného tela iba štyri kovové okrúhle tyčky, od ktorých sa lúče skoro vôbec neodrážali. Takáto situácia nám pred tým počas testovania nenastala a neboli sme vôbec na to pripravení. Z tohto dôvodu modul pre priblíženie skončil zakaždým neúspešne a nemohli sme zozbierať ani jednu loptičku. Takýto výsledok malo za následok aj to, že čas na testovanie počas súťaže bol veľmi krátky a na ihrisku sa nachádzali všetky tímy naraz. Nedostali sme sa ani raz k pokusu o zarovnanie k transportéru a tak sme nedokázali túto chybu odladiť v čas.

Od organizátorov sme boli pochválení za výbornú lokalizáciu a navigáciu, pretože náš robot sa niekoľkokrát presunul po ihrisku a prišiel presne na tie isté miesta. Bolo evidentne zrejmé, že robot vždy presne vedel, kde sa práve nachádza.

Kapitola 7

Záver

Cieľom práce bolo vytvoriť systém pre lokalizáciu, navigáciu a stratégiu do súťaže SICK robot day 2018. Jednotlivé moduly sme úspešne vytvorili a dajú sa využiť v podobných súťažiach, okrem strategického modulu. Odsledovaním robota sme odhalili kritické nedostatky, ktoré sa týkali hlavne strategickej časti.

Výsledok lokalizačného systému je takmer 100%. Jednotlivé moduly sa vedeli naň spoľahnúť, pretože výpočty skončili vždy úspešne a ich výsledok zodpovedal skutočnosti. Počas testov a aj počas súťaže sa nám ani raz nestalo, aby sa robot nevedel úspešne zlokalizovať pomocou laserového senzora a každý výsledok bol takmer úplne presný.

Celá štruktúra projektu sa upravila a vylepšila. Pre lepšiu modularitu sú jednotlivé implementácie teraz aj dlhodobo ľahšie udržateľné.

Námetom na ďalšiu prácu by mohlo byť nadviazanie na výsledky z jednotlivých lokalizačných modulov a pridať k nim pravdepodobnostný model. V prípade opätovnej účasti na súťaži, treba jednoznačne vylepšiť stratégiu v oblasti zarovnávania k svojmu cieľu.

Vytvorený softvér je open-source a je dostupný na adresách
https://github.com/Robotics-DAI-FMFI-UK/mikes-common/
a https://github.com/Robotics-DAI-FMFI-UK/mikes-generic/

Literatúra

[DF15] Jean-Arcady Meyer David Filliat. Map-based navigation in mobile robots: I. a review of localization strategies. 2015.

[DH72] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. 1972.

[Du7] Bc. Jozef Duc. Robotour with laser range sensor. Master's thesis, Fakulta matematiky, fyziky a informatiky univerzita Komenského v Bratislave, 2017.

[Hon10] Honeywell Solid State Electronics Center. *3-Axis Digital Compass IC*, 2010.

[Mad14] Bc. Marína Madovác. Robot poštár. Master's thesis, Fakulta matematiky, fyziky a informatiky univerzita Komenského v Bratislave, 2014.

[MBA17] Francesco Mondada Mordechai Ben-Ari. *Elements of Robotics*. Springer, Cham, 2017.

[RS04] Illah R. Nourbakhsh Roland Siegwart. *Introduction to Autonomous Mobile Robots*. Massachusetts Institute of Technology, 2004.

[SIC16] SICK AG. *TiM55x/TiM56x/TiM57x RANGING LASER SCANNER*, 2016.

[SMUM15] Muhammad Waqas Khan Taimoor Zahid Syed Muhammad Umer, Aftab Ahmed Chandio and Fayyaz Muhammad. A study of distinctive localization and navigation methods for mobile robots. 2015.

[ST00] Wolfram Burgard Sebastian Thrun, Dieter Fox. *PROBABILISTIC ROBOTICS*. Massachusetts Institute of Technology, 2000.

[ST03] Wolfram Burgard Frank Dellaert Sebastian Thrun, Dieter Fox. Robust monte carlo localization for mobile robots. 2003.

[Zem17] Radoslav Zeman. Integrovaný navigačný systém pre mobilného robota, 2017.

Zoznam obrázkov