Razvoj fizikalnog modela mobilnog robota u Gazebo simulatoru

Brnadić, Domagoj

Sveučilište u Zagrebu
Fakultet strojarstva i brodogradnje

DIPLOMSKI RAD

Domagoj Brnadić

Zagreb, 2021.

Sveučilište u Zagrebu
Fakultet strojarstva i brodogradnje

DIPLOMSKI RAD

RAZVOJ FIZIKALNOG MODELA MOBILNOG ROBOTA U GAZEBO SIMULATORU

Mentor:
doc. dr. sc. Marko Švaco

Student:
Domagoj Brnadić

Zagreb, 2021.

Zahvaljujem se svome mentoru doc.dr.sc. Marku Švaci te dr. sc. Josipu Vidakoviću mag.ing. koji su iskazali povjerenje te svojim znanstvenim i stručnim savjetima oblikovali temeljnu ideju ovog rada i pomogli mi u njegovoj realizaciji. Želim se također zahvaliti i svim ostalim djelatnicima i asistentima na Katedri za autonomne sustave i računalnu inteligenciju na suradnji i ugodnom boravku prilikom diplomskog studija. Na kraju bih se posebno zahvalio svojoj obitelji, svim kolegama i prijateljima te djevojci na iskazanom povjerenju, strpljenju te podršci koju su mi ukazali tijekom studija.

Izjava

Izjavljujem da sam ovaj rad radio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zagreb, srpanj 2021. Domagoj Brnadić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske radove studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment,
inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa: 602-04/21-6/1	
Ur. broj: 15-1703-21	

DIPLOMSKI ZADATAK

Student: **DOMAGOJ BRNADIĆ** Mat. br.: 0035198055

Naslov rada na hrvatskom jeziku: **Razvoj fizikalnog modela mobilnog robota u Gazebo simulatoru**

Naslov rada na engleskom jeziku: **Development of a physical mobile robot model in the Gazebo simulator**

Opis zadatka:

Razvoj koncepta mobilnog robota počinje prijedlogom njegovog pogonskog, a potom i senzorskog sustava. Simulacija koncepta i testiranje u simulacijskom okruženju pri tome predstavlja vrlo vrijednu mogućnost. ROS (eng. Robot Operating System) okruženje putem Gazebo fizikalnog simulatora omogućuje implementaciju proizvoljne robotske strukture zajedno sa senzorskim elementima i testiranje različitih upravljačkih algoritama u realnim uvjetima bez prisustva pravog robota.
U sklopu rada potrebno je u Gazebo okruženju implementirati eksperimentalnog mobilnog robota dostupnog u laboratoriju. Koncept mobilnog robota sastoji se od diferencijalnog pogona s jednom dodatnom servo osi za kutno pozicioniranje tereta, podiznog sustava za podizanje tereta i senzorskog dijela koji se sastoji od dva lidar senzora, dubinske kamere i enkodera. U simulatoru je potrebno implementirati sve bitne mehaničke, pogonske i senzorske elemente za vjernu simulaciju mobilnog robota.
Na simuliranom robotu potrebno je implementirati i testirati sljedeće algoritme:
- algoritam za izgradnju karte prostora iz dvodimenzionalnih podataka dobivenih s lidar senzora,
- algoritam za automatsku lokalizaciju robota u mapi,
- algoritam za globalno planiranje kretanja u mapi,
- algoritam za lokalno planiranje kretanja.

Algoritmi za globalno i lokalno planiranje kretanja trebaju biti razvijeni za kretanje robota s teretom čiji je centar mase na visini od 900 mm i čija je masa 320 kg. Algoritme treba verificirati u različitim konfiguracijama mape za različite početne i krajnje pozicije putanje. Kao mjeru za usporedbu kvalitete kretanja treba uzeti mogućnost pronalaska rješenja, izbjegavanja prepreka, te akceleraciju robota kroz vrijeme. U okviru ovih algoritama potrebno je implementirati i prikladne strategije za ponovno uspostavljanje gibanja (eng. recovery strategy) s obzirom na karakteristike tereta.
U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:	Rok predaje rada:	Predviđeni datum obrane:
6. svibnja 2021.	8. srpnja 2021.	12. srpnja do 16. srpnja 2021.

Zadatak zadao: Predsjednica Povjerenstva:

doc. dr. sc. Marko Švaco prof. dr. sc. Biserka Runje

Sadržaj

Sadržaj vi

Popis slika viii

Popis tablica xi

Popis oznaka xii

Sažetak xii

Summary xiii

1. Uvod 1

1.1. RONNA - Robotic Neuronavigation 1

1.2. Mobilni roboti 3

1.2.1. Roboti na kotačima 3

1.2.2. Vrste pogona 6

1.3. Senzorika 6

1.3.1. Odometrija 8

1.3.2. Inercijske mjerne jedinice 9

1.3.3. LiDAR 10

1.4. Robot Operating System 11

1.4.1. Navigation stack 14

2. Teorija **17**
2.1. Kinematika . . . 17
2.1.1. Kinematika mobilnog robota s diferencijalnim pogonom [1] . . . 19
2.2. Navigacija . . . 21
2.2.1. Monte Carlo algoritam [2] . . . 21
2.2.2. SLAM . . . 24

3. Razrada **27**
3.1. Model robota . . . 27
3.1.1. URDF . . . 27
3.1.2. Launch datoteke . . . 32
3.2. Izrada karte . . . 35
3.3. Autonomna navigacija . . . 35
3.3.1. DWA . . . 38
3.3.2. TEB . . . 38
3.4. Prikupljanje brzina i akceleracija . . . 40
3.4.1. Čvor brzina, vremena i prijeđenog puta . . . 42
3.4.2. Čvor kutnih i linearnih akceleracija . . . 43

4. Rezultati **44**
4.0.1. DWA . . . 45
4.0.2. TEB . . . 45
4.0.3. Analiza rezultata . . . 47

5. Zaključak **52**

A. Prvi prilog **53**

B. Drugi prilog **55**

Literatura **58**

Popis slika

1.1 Prikaz sustava RONNA G3 s komponentama: (1) glavni robot, (2) pomoćni robot, (3) univerzalna mobilna platforma, (4) optički sustav za praćenje i (5) softversko sučelje za upravljanje i planiranje. [3] 2
1.2 Spot, četveronožni komercijalni robot tvrtke Boston Dynamics [4] 3
1.3 Četiri osnovne vrste kotača. (a) Standardni kotač: dva stupnja slobode; rotacija oko (motorizirane) osovine kotača i kontaktne točke. (b) Kotačić: dva stupnja slobode; rotacija oko pomaknutog zglobnog zgloba. (c) Švedski kotač: tri stupnja slobode; rotacija oko (motorizirane) osovine kotača, oko valjaka i oko točke kontakta. (d) Lopta ili sferni kotač: tehnički je teško ostvariti. [1] 4
1.4 Osnovna konfiguracija "Omni-ball-a" [5] 5
1.5 Mogućnosti kretanja diferencijalnog pogona. (A) Ravna staza, (B) zakrivljena staza, (C) kružna staza, (D) manevriranje bez prepreka za prelazak iz početne u završnu pozu i (E) manevriranje za prelazak iz početne u završnu pozu dok izbjegava preprek. [6] 7
1.6 Kinematika diferencijalnog pogona [7] 10
1.7 IMU blok dijagram 11
1.8 Hokuyo UST-10LX [8] 11

1.9 Pregled mehanizma ROS tema. Kada su izdavač (*eng. Publisher*) (a) i pretplatnik (*eng. subscriber*) (b) registrirani na istu temu, pretplatnik dobiva mrežnu adresu i priključke (*eng. port*) svih izdavača (c). Pretplatnik nastavlja izravnim kontaktiranjem svakog izdavača koji zauzvrat započinje slanje podataka izravno pretplatniku (d). Mnogi čvorovi mogu objaviti i pretplatiti se na istu temu što rezultira N/M relacijom (e). Na mrežnom sloju postoje $N \times M$ veze, po jedna za svaki (izdavač, pretplatnik) skup. Čvorovi se mogu distribuirati na bilo koji broj hostova unutar ROS mreže.[9] 12
1.10 Pregled tipičnog sustava koji pokreće Navigation stack[10] 15

2.1 Globalni referentni okvir i lokalni referentni okvir robota [1] 18
2.2 Mobilni robot poravnat s globalnom osi [1] 19
2.3 Opća shema za lokalizaciju mobilnog robota. [1] 21
2.4 MCL, ili Monte Carlo Localization, lokalizacijski algoritam zasnovan na filtrima čestica [2] 22
2.5 Adaptivna varijanta MCL-a koja dodaje slučajne uzorke. Broj slučajnih uzoraka određuje se usporedbom kratkoročne i dugoročne vjerojatnosti mjerenja senzora.[2] 23
2.6 Lokalizacija Monte Carlo sa slučajnim česticama. Svaka slika prikazuje skup čestica koji predstavlja procjenu položaja robota (male crte označavaju orijentaciju čestica). Veliki krug prikazuje srednju vrijednost čestica, a pravi položaj robota označava mali, bijeli krug. Slike ilustriraju globalnu lokalizaciju (a)-(d) i relokalizaciju (e)-(h).[2] 25

3.1 Stvarni prikaz robota 28
3.2 3D funkcionalni model robota 28
3.3 Drugi pogled na 3D funkcionalni model robota 29
3.4 URDF model s pripadajućim komponentama unutar Gazebo okruženja . 32
3.5 Hijerarhija direktorija unutar ROS okruženja 34
3.6 3D prikaz okruženja unutar kojeg će se nalaziti robot 36
3.7 3D prikaz okruženja unutar kojeg će se nalaziti robot iz druge perspektive 36
3.8 Pogled na okruženje iz ptičje perspektive 37
3.9 Karta stvorena pomoću navigacijskog paketa 37

3.10 Prikaz svih učitanih parametara unutar rviz okruženja 40
3.11 Prikaz stvorene karte unutar rviz okruženja 41
3.12 Prikaz procesa lokalizacije pomoću Monte Carlo algoritma 42

4.1 Usporedba vidljive prepreke u Gazebo simulatoru te ostvarene prilagodbe putanje unutar rviz-a. 45
4.2 Prikaz putanje DWA algoritma za lokalno planiranje 46
4.3 Prikaz putanje TEB algoritma za lokalno planiranje 48
4.4 Kutna akceleracija DWA algoritma između točaka 2 i 3 49
4.5 Kutna akceleracija TEB algoritma između točaka 2 i 3 49
4.6 Linearna akceleracija DWA algoritma između točaka 8 i 9 50
4.7 Linearna akceleracija TEB algoritma između točaka 8 i 9 50
4.8 Usporedba prijeđenog puta po putanjama 51
4.9 Usporedba potrebnog vremena za sve putanje 51

Popis tablica

3.1 Parametri za podešavanje lokalnog planera 39

Sažetak

Svrha robotske neuronavigacije (u daljnjem tekstu RONNA) istraživanje je i razvitak inovativnih i konkurentnih robotskih sustava za neurokirurške primjene. Roboti u upotrebi su teški i stoga nisu idealni za preciznu navigaciju. Glavni cilj ovog rada je uspostaviti okvir koji će omogućiti nastavak istraživanja mobilnosti RONNA-e. U sklopu ovog rada razvijena je i implementirana platforma za autonomnu navigaciju mobilnog robota s diferencijalnim pogonom. Implementacija algoritama za lokalizaciju te za globalno i lokalno planiranje putanje izvršena je pomoću Robotskog Operativnog Sustava (ROS).

Kao dio rada napravljena je i analiza te verifikacija dva različita algoritma za lokalno planiranje putanje. Izvedba algoritama DWA (*eng. Dynamic Window Approach*) i TEB (*eng. Timed Elastic Band*) procijenjena je kroz niz različitih konfiguracija u okviru tlocrta Regionalnog centra izvrsnosti za robotske tehnologije (CRTA) Sveučilišta u Zagrebu. Uz ove procjene, razvijena su i dva vanjska programa koji prikupljaju i uspoređuju podatke o ubrzanjima robota. Algoritmi su morali zadovoljiti uvjete sigurnosti, reaktivnosti te optimalnosti putanje.

U ovim testovima TEB algoritam ocijenjen je kao najprikladniji za potrebe ovog specifičnog robotskog sustava za neurokirurške primjene te za postojeći hardver.

Ključne riječi: ROS, Autonomna navigacija, Mobilni roboti, Algoritmi za lokalno planiranje putanje, RONNA

Summary

Robotic Neuronavigation's (hereinafter RONNA) aim is to explore and develop an innovative and competitive robotic system for neurosurgical applications. The robots in use are heavy and therefore not ideally suited for precise navigation. The main goal of this thesis is to establish a framework that will enable the continuation of RONNA mobility research. Within this thesis, a platform for autonomous navigation of a mobile robot with a differential drive was developed and implemented. The implementation of algorithms for localization and for global and local path planning was performed using the Robot Operating System (ROS).

As a part of this thesis, the verification of two different algorithms for local path planning was made and analyzed. DWA (Dynamic Window Approach) and TEB (Timed Elastic Band) algorithms performance was evaluated through a number of different configurations within the layout of the Regional Centre of Excellence for Robotics at the University of Zagreb. In addition to these evaluations, two external programs have also been developed that collect and compare acceleration data. Path planning algorithms had to meet the criteria of safety, reactivity and trajectory optimality.

In this empiric test, the TEB algorithm was evaluated as the best fit for the needs of this specific robotic system for neurosurgical applications and for the existing hardware.

Keywords: ROS, Autonomous navigation, Mobile robots, Algorithms for local path planning, RONNA

1 Uvod

1.1. RONNA - Robotic Neuronavigation

RONNA G4 robotski je sustav za neuronavigaciju zasnovan na zglobnim robotskim rukama koji je namijenjen minimalno invazivnim stereotaktičkim postupcima kao što su biopsije, stereoelektroencefalografije, operacije epilepsije, duboke stimulacije mozga i resekcije tumora. RONNA se može konfigurirati kao sustav s jednom ili dvije ruke: sustav s jednom rukom namijenjen je stereotaktičkoj neuronavigaciji i služi kao navigacijski pomoćnik kirurgu, dok konfiguracija s dvije ruke obavlja autonomne invazivne zadatke poput bušenja kostiju, umetanje sonde itd. RONNA-u karakterizira potpuno automatizirani postupak registracije pacijenta, planiranje položaja robota, precizno vođenje instrumenta i autonomno bušenje kostiju. U sklopu projekta, razvijena je i nova metoda lokalizacije koja kombinira strojni vid i matematičku procjenu, kao i novi algoritam korespondencije točkastih uparivanja i funkciju višeobjektnih funkcija troškova za optimizaciju postavljanja robota. RONNA pruža pozicioniranje kirurškog alata unutar intrakranijalnog prostora pacijenta te robotsku pomoć u operacijama koje zahtijevaju izvanrednu preciznost. Klinička primjena RONNA-e u stereotaktičkim neurokirurškim zahvatima skraćuje vrijeme operacije, smanjuje invazivnost zahvata, omogućuje brži oporavak pacijenta i bolju upotrebu bolničkih operativnih resursa. Od 2016. godine RONNA prolazi klinička ispitivanja u KBC Dubrava.[3] Mobilni robot koji je opisan i simuliran u ovom radu zamišljen je kao platforma i prijevozno sredstvo RONNA-e u bolničkom okruženju. Imajući na umu vanjske dimenzije RONNA-e, njenu težinu te osjetljivost na udarce tijekom prijevoza, od važnosti je da navigacijski algoritam bude sofisticiran i precizan.

Slika 1.1: Prikaz sustava RONNA G3 s komponentama: (1) glavni robot, (2) pomoćni robot, (3) univerzalna mobilna platforma, (4) optički sustav za praćenje i (5) softversko sučelje za upravljanje i planiranje. [3]

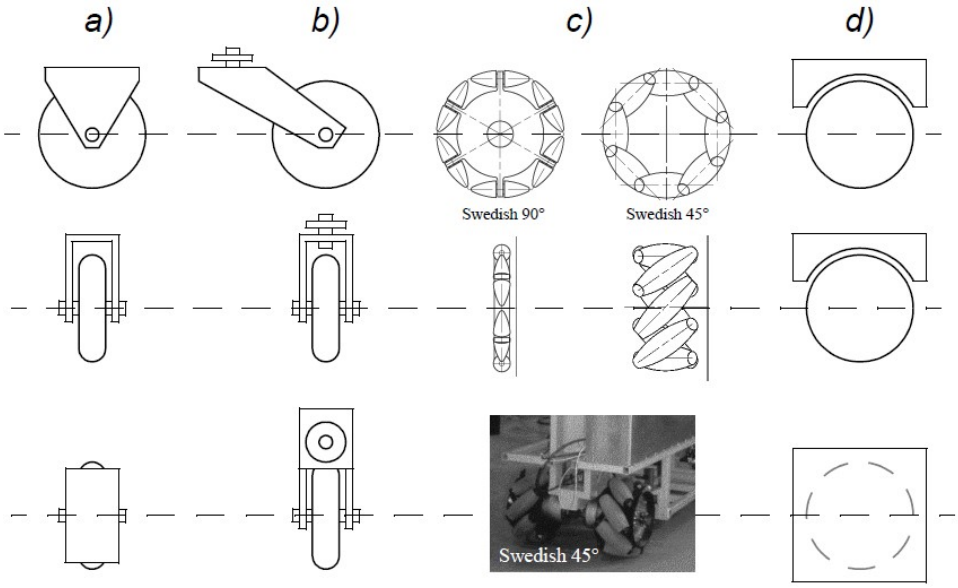
Slika 1.2: Spot, četveronožni komercijalni robot tvrtke Boston Dynamics [4]

1.2. Mobilni roboti

Mobilni roboti oni su roboti koji se mogu samostalno pomicati s jednog položaja na drugi, odnosno bez pomoći vanjskih ljudskih operatera. U usporedbi s većinom industrijskih robota koji imaju mogućnost kretanja samo u zadanom radnom prostoru, mobilni roboti imaju posebnost slobodnog kretanja unutar unaprijed definiranog radnog prostora kako bi postigli zadane željene ciljeve. Ova sposobnost mobilnosti čini ih pogodnima za veliki spektar aplikacija u strukturiranim i nestrukturiranim okruženjima. Mobilne robote dijelimo na prizemne, leteće te podvodna vozila. Prizemne potom dijelimo na mobilne robote na kotačima te mobilne robote s nogama. Mobilni roboti na kotačima vrlo su popularni jer su prikladni za tipične primjene s relativno malom mehaničkom složenošću i potrošnjom energije. Roboti s nogama, kakvog vidimo na slici 1.2 prikladni su za zadatke u nestandardnim okruženjima, stepenicama, hrpama ruševina i slično. Tipično su sustavi s dvije, tri, četiri ili šest nogu, no postoje i mnoge druge izvedbe. Ovaj će se rad usredotočiti na mobilne robote na kotačima. [6]

1.2.1. Roboti na kotačima

Kako bi postigli kretanje robota, mobilni roboti na kotačima u širokoj su primjeni kako u mobilnoj robotici tako i općenito u svim vrstama vozila. Roboti na kotačima troše manje energije i kreću se brže od ostalih mehanizama za kretanje (npr. robota

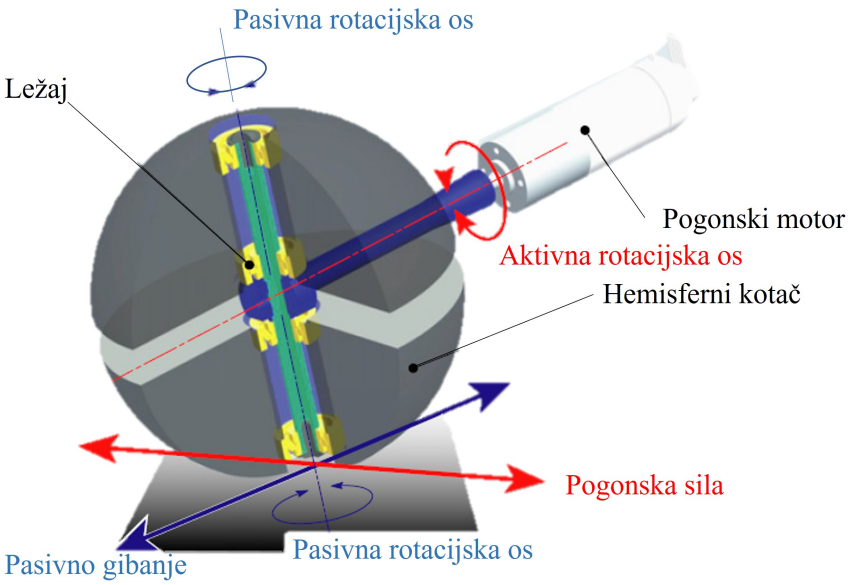


Slika 1.3: Četiri osnovne vrste kotača. (a) Standardni kotač: dva stupnja slobode; rotacija oko (motorizirane) osovine kotača i kontaktne točke. (b) Kotačić: dva stupnja slobode; rotacija oko pomaknutog zglobnog zgloba. (c) Švedski kotač: tri stupnja slobode; rotacija oko (motorizirane) osovine kotača, oko valjaka i oko točke kontakta. (d) Lopta ili sferni kotač: tehnički je teško ostvariti. [1]

s nogama ili gusjeničnih vozila). Sa stajališta upravljanja potrebno je manje napora u upravljanju, zahvaljujući njihovim jednostavnim mehanizmima i smanjenim problemima stabilnosti. [11] Tri kotača dovoljna su za postizanje stabilne ravnoteže, iako roboti s dva kotača, uz žiroskopska rješenja, također mogu biti stabilni. Kada se koristi više od tri kotača, potreban je sustav ovjesa koji omogućava svim kotačima da održavaju kontakt s tlom kada robot naiđe na neravne terene. [6] Iako je teško prevladati neravne terene ili neravne uvjete na terenu, mobilni roboti na kotačima prikladni su za veliku klasu ciljanih okruženja u praktičnoj primjeni.

Dizajn kotača

Postoje četiri glavne klase kotača, kao što je prikazano na slici 1.3.
Sve klase kotača široko se razlikuju po pripadajućim kinematikama, stoga izbor vrste



Slika 1.4: Osnovna konfiguracija "Omni-ball-a" [5]

kotača ima velik utjecaj na ukupnu kinematiku mobilnog robota. Standardni kotač i kotačić imaju primarnu os rotacije i stoga su visoko usmjerljivi. Kada kotačem želimo upravljati u drugom smjeru njime se mora upravljati duž okomite osi. Ključna razlika između ova dva kotača u tome je što standardni kotač može postići ovo kretanje upravljača bez nuspojava jer središte rotacije prolazi kroz kontaktnu plohu s tlom, dok se kotačić okreće oko pomaknute osi, što uzrokuje predaju sile šasiji robota tijekom upravljanja. Švedski kotač i sferni kotač dizajni su koji su manje ograničeni usmjerenošću od uobičajenih standardnih kotača. Švedski kotač funkcionira kao normalan kotač, ali pruža nizak otpor i u drugom smjeru, ponekad okomit na konvencionalni smjer (kao u švedskom 90), a ponekad pod srednjim kutom (kao u švedskom 45). Sferni kotač uistinu je svesmjerni kotač, često dizajniran tako da se može aktivno pokretati da se vrti u bilo kojem smjeru. Unaprijeđeno rješenje trenutnog dizajna robota sadržavat će upravo sferni kotač napravljen po ugledu na "omni-ball" razvijen na Sveučilištu u Osaki koji možemo vidjeti na 1.4. Trenutno rješenje ima standardni kotač. [1] [5]

1.2.2. Vrste pogona

Kod mobilnih robota razlikujemo različite tipove upravljanja. Najčešći su:

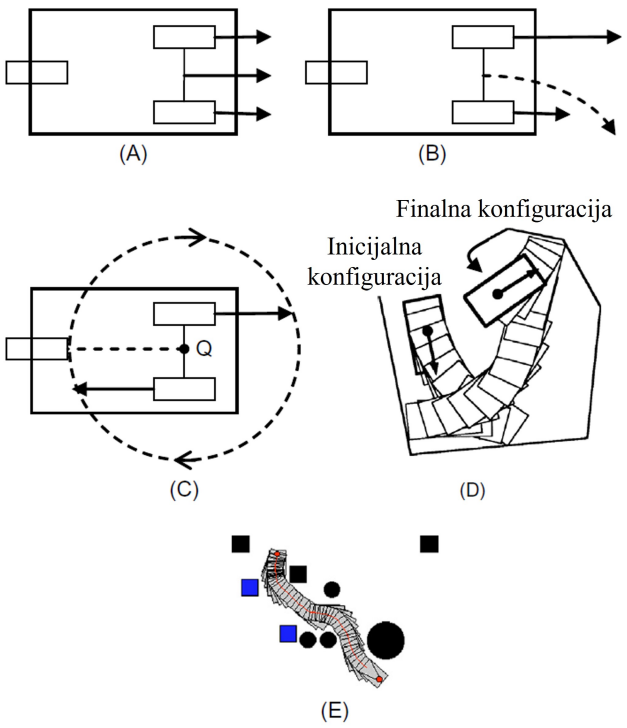
- Diferencijalni pogon
- Tricikl
- Svesmjerno
- Sinkroni pogon
- Ackermanovo upravljanje
- Klizno upravljanje

Diferencijalni pogon

Robot kojega ovaj rad opisuje i proučava pogonjen je diferencijalnim pogonom koji se sastoji od dva fiksna pogonska kotača postavljena na lijevoj i desnoj strani robotske platforme. Dva kotača neovisno su vođena. Četiri pasivna kotača koriste se za ravnotežu i stabilnost. Diferencijalni pogon najjednostavniji je mehanički pogon jer mu nije potrebna rotacija pogonske osi. Ako se kotači okreću istom brzinom, robot se kreće ravno naprijed ili natrag. Ako jedan kotač radi brže od drugog, robot slijedi zakrivljeni put duž luka trenutne kružnice. Ako se oba kotača okreću istom brzinom u suprotnim smjerovima, robot se okreće oko središnje točke dva pogonska kotača. Gore navedeni načini kretanja prikazani su na slici 1.5. [6]

1.3. Senzorika

Jedan od najvažnijih zadataka autonomnog sustava bilo koje vrste stjecanje je znanja o svom okruženju. To se postiže mjerenjem pomoću različitih senzora koji iz tih mjerenja izdvajaju značajne informacije. Senzori dizajnirani za robote nalikuju senzorima ljudskog osjetilnog sustava (npr. vida, sluha, kinestetičkih osjeta) koji pružaju ulazne signale u mozak za obradu, korištenje i djelovanje. Korištenje senzora u robotici od iznimne je važnosti za zatvaranje upravljačkih petlji s povratnim informacijama koje osiguravaju učinkovit i automatiziran/autonoman rad robota u stvarnim aplikacijama.



Slika 1.5: Mogućnosti kretanja diferencijalnog pogona. (A) Ravna staza, (B) zakrivljena staza, (C) kružna staza, (D) manevriranje bez prepreka za prelazak iz početne u završnu pozu i (E) manevriranje za prelazak iz početne u završnu pozu dok izbjegava preprek. [6]

Metode osjetljivosti pružaju robotima višu razinu i inteligencijske sposobnosti koje idu daleko dalje od "unaprijed programiranog" stila rada ponavljajućim izvršavanjem niza programiranih zadataka. [6]
U mobilnim robotima koristi se široka paleta senzora. Neki se senzori koriste za mjerenje jednostavnih vrijednosti poput unutarnje temperature elektronike robota ili brzine vrtnje motora. Ostali, sofisticiraniji, senzori mogu se koristiti za prikupljanje informacija o okolini robota ili za izravno mjerenje globalnog položaja robota. Budući da se mobilni robot kreće često će se susresti s nepredviđenim okolišnim karakteristikama pa je stoga takvo otkrivanje posebno kritično. [1] Robot kojega ovaj rad opisuje sastoji se od sljedećih senzora:

- Inercijske mjerne jedinice (Inertial Measurement Units – IMUs)
- Dva enkodera
- Dva LiDARa
- Dubinska stereo kamera

1.3.1. Odometrija

U kontekstu autonomnih vozila odometrija se obično odnosi na upotrebu podataka iz aktuatora za procjenu ukupnog kretanja vozila. Osnovni je koncept razviti matematički model načina zapovijedane kretanjem kotača, potaknuti kretanje samog vozila, a zatim integrirati te zapovjeđene pokrete s vremenom kako bi se razvio model poze vozila u funkciji vremena. Korištenje podataka o odometriji za procjenu poze vozila u ovisnosti o vremenu poznato je kao računska navigacija (eng. *Dead Reckoning*) ili deduktivno računanje.[7] Robot kojeg opisuje ovaj rad diferencijalno je pogonsko vozilo, ima dva pogonska kotača kojima se neovisno može upravljati i koji su postavljeni duž zajedničke osi. Mjesto kotača fiksirano je na vozilu, a kotači su u stalnom kontaktu sa zemljom, dva kotača moraju opisivati lukove u ravnini tako da se vozilo okreće oko točke (poznate kao ICC - trenutno središte zakrivljenosti (eng. *instantaneous center of curvature*)) koji leži na zajedničkoj osi kotača kao što je vidljivo na slici 1.10. Ako su brzine dodira lijevog i desnog kotača s tlom v_l i v_r, a kotači su odvojeni udaljenost $2d$, tada vrijedi:

$$\omega(R + d) = v_r \tag{1.1}$$

$$\omega(R-d) = v_l \tag{1.2}$$

Možemo ove dvije jednadžbe preurediti za rješavanje ω, brzina rotacije oko ICC i R, udaljenosti od središta robota do ICC.

$$\omega = \frac{(v_r - v_l)}{2d} \tag{1.3}$$

$$R = \frac{d(v_r + v_l)}{(v_r - v_l)} \tag{1.4}$$

Trenutna brzina točke na sredini između kotača robota definirana je kao $V = \omega R$. Sada, kako su v_l i v_r funkcije vremena, možemo generirati jednadžbu gibanja za robota s diferencijalnim pogonom. Koristeći točku na sredini između kotača kao ishodište robota i zapisujući Θ kao orijentaciju robota s obzirom na x-os globalnog kartezijanskog koordinatnog sustava dobivamo:

$$x(t) = \int V(t)\cos[\Theta(t)]dt \tag{1.5}$$

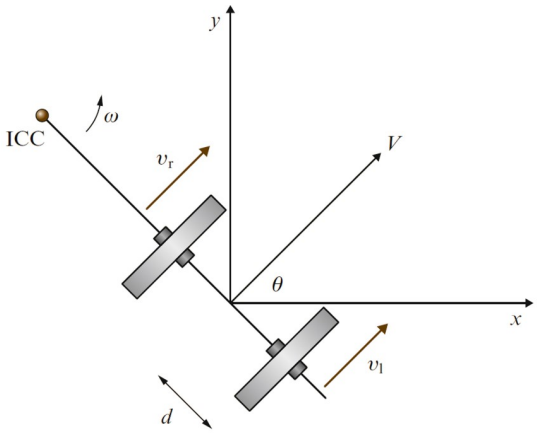
$$y(t) = \int V(t)\sin[\Theta(t)]dt \tag{1.6}$$

$$\Theta(t) = \int \omega(t)dt \tag{1.7}$$

Ovo je rješenje za odometriju diferencijalnog pogonskog vozila. S obzirom na kontrolne ulaze (v_l i v_r) i neke početne procjene stanja, možemo procijeniti gdje će idealizirani robot koji koristi ovaj model kretanja biti u bilo kojem trenutku t. [7] Nažalost, ovo radi samo u teoriji te je zbog raznih grešaka koje se zbrajaju potrebno ispravljati ove pogreške te one zahtijevaju integraciju procjene računske navigacije s dobivenim procjenama ostalih senzorskih sustava

1.3.2. Inercijske mjerne jedinice

Inercijska mjerna jedinica (IMU) uređaj je koji pomoću žiroskopa i akcelerometara procjenjuje relativni položaj, brzinu i ubrzanje vozila u pokretu. IMU je poznat i kao inercijski navigacijski sustav (INS). IMU procjenjuje pozu vozila od šest stupnjeva slobode: položaj (x, y, z) i orijentacija (α, β, γ). Uz pozu vozila, IMU-ovi često imaju

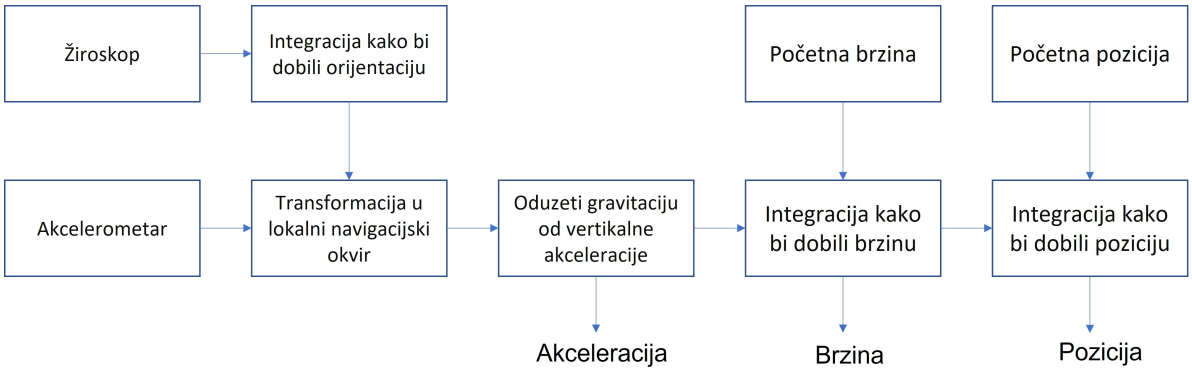


Slika 1.6: Kinematika diferencijalnog pogona [7]

mogućnost procjene brzine i ubrzanja. Kako bi se procijenila trenutna brzina treba znati početnu brzinu vozila. IMU ovog sustava ima tri pravokutna akcelerometra i tri ortogonalna žiroskopa. Podaci o žiroskopu integrirani su za procjenu orijentacije vozila, dok se tri akcelerometra koriste za procjenu trenutnog ubrzanja vozila. Ubrzanje se zatim transformira u lokalni navigacijski okvir pomoću trenutne procjene orijentacije vozila u odnosu na gravitaciju. U ovom se trenutku iz mjerenja može oduzeti gravitacijski vektor. Rezultirajuće ubrzanje zatim se integrira da se dobije brzina, a zatim ponovno integrira da se dobije položaj, pod uvjetom da su i početna brzina i položaj od prije poznati. Da bi se prevladala potreba za poznavanjem početne brzine, integracija se obično započinje u mirovanju (tj. brzina jednaka nuli). Blok dijagram ovog postupka prikazan je na slici 1.7. [1]

1.3.3. LiDAR

Laserski senzori blizine čine poseban slučaj optičkih senzora koji imaju raspon od nekoliko centimetara do nekoliko metara, također se nazivaju i laserskim radarima (ili lidar = senzorima smjera i dometa svjetla). Energija se emitira impulsima, a udaljenost se izračunava iz vremena leta, a laserski se senzori također mogu koristiti kao laserski visinomjeri za izbjegavanje prepreka ili za otkrivanje vozila na autocestama. Laserski radar koji koristimo u sklopu ovog rada prikazan je na slici 1.8. Rotacijski laseri temelje se na rotaciji vala 360° pri više od 1 ili 2 okretaja/min i zrcala pri 45°. Time se prevladava



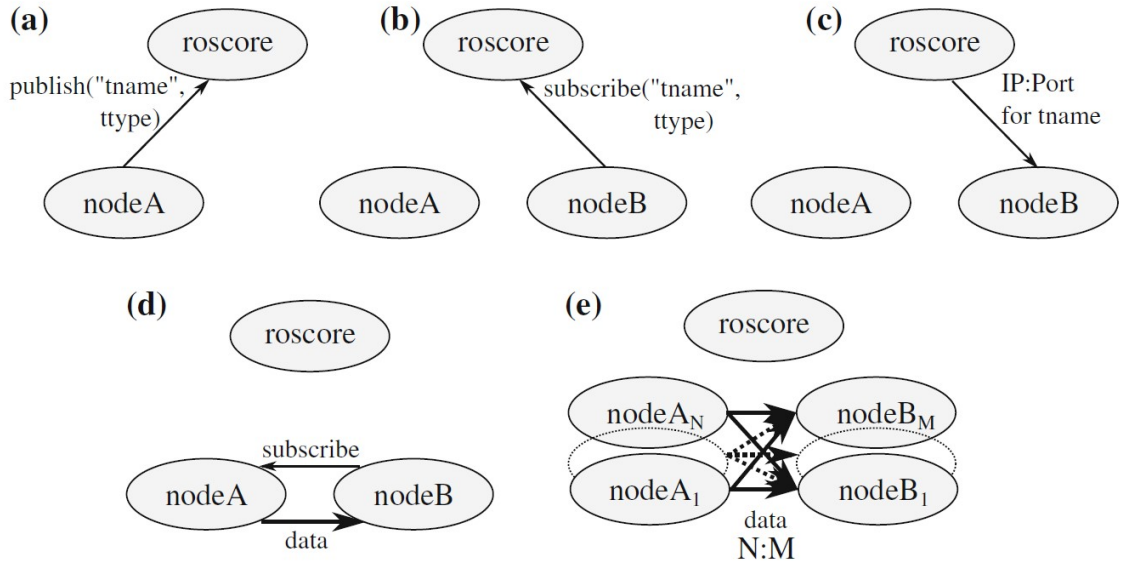
Slika 1.7: IMU blok dijagram

Slika 1.8: Hokuyo UST-10LX [8]

problem skrivenih područja. [6] Zbog velike brzine i kontinuiranog prikupljanja podataka koje pruža LIDAR, podaci obično nemaju pravokutni otisak kao drugi oblici rasterskih podataka. Skupovi podataka LIDAR-a obično se nazivaju oblaci točaka jer su podaci tok (x, y, z) lokacija, gdje je z udaljenost od lasera do otkrivenog objekta, a vrijednosti (x, y) su projicirano mjesto objekta izračunato iz mjesta senzora. [12]

1.4. Robot Operating System

Robot Operating System (ROS) okvir je za pisanje robotskog softvera. Može ga se opisati kao "middleware", kao zbirku alata, biblioteka te standarda kojima je cilj pojednostaviti zadatak stvaranja složenog i robusnog ponašanja robota na širokom spektru robotskih platformi.



Slika 1.9: Pregled mehanizma ROS tema. Kada su izdavač (*eng. Publisher*) (a) i pretplatnik (*eng. subscriber*) (b) registrirani na istu temu, pretplatnik dobiva mrežnu adresu i priključke (*eng. port*) svih izdavača (c). Pretplatnik nastavlja izravnim kontaktiranjem svakog izdavača koji zauzvrat započinje slanje podataka izravno pretplatniku (d). Mnogi čvorovi mogu objaviti i pretplatiti se na istu temu što rezultira N/M relacijom (e). Na mrežnom sloju postoje $N \times M$ veze, po jedna za svaki (izdavač, pretplatnik) skup. Čvorovi se mogu distribuirati na bilo koji broj hostova unutar ROS mreže.[9]

ROS je veliki projekt koji ima mnogo predaka i suradnika. Potrebu za otvorenim okvirom suradnje osjećali su mnogi ljudi u zajednici istraživača robotike. Razni projekti na Sveučilištu Stanford sredinom 2000.-ih koji uključuju integrativni, utjelovljeni AI, poput STanford AI Robot (STAIR) i programa Personal Robots (PR). Napor su potaknuli nebrojeni istraživači koji su svoje vrijeme i stručnost dali jezgri ROS-a i njegovim temeljnim softverskim paketima. Tijekom cijelog vremena softver se razvijao otvoreno, koristeći permisivnu BSD licencu otvorenog koda i postupno je postao široko korišten u zajednici istraživača robotike.

ROS teme implementacija su mehanizma objavljivanje-pretplata, u kojem ROS Master

služi kao ulazna točka za imenovanje i registraciju. Svaki ROS čvor oglašava teme koje objavljuje ili na koje se pretplaćuje na ROS Master. Ako objavljivanje i pretplata postoje za istu temu, stvara se izravna veza između izdavačkog i pretplatničkog čvora, kao što je prikazano na slici 1.9. Lista 1.4. objašnjava kratki pregled ROS terminologije:

- "Master": Jedinstvena, (ROS_MASTER_URI varijabla okoline) ulazna točka za imenovanje i registraciju. Naziva se i roscore.
- (Grafički resurs) "Name": Ime resursa (čvor, tema, usluga ili parametar) unutar ROS grafikona izračuna. Shema imenovanja hijerarhijska je i ima mnogo poveznica s UNIX-ovim datotečnim sustavom, npr. mogu biti apsolutni ili relativni.
- "Host": Računalo unutar ROS mreže, identificirano njegovom IP adresom (varijabla okruženja ROS_IP).
- "Node": Bilo koji proces koji koristi API ROS klijenta.
- "Topic": Jednosmjerni, asinkroni, imenovani komunikacijski kanal koji se koristi u mehanizmu objavljivanje-pretplata
- "Message": Specifična struktura podataka, temeljena na skupu ugrađenih tipova.
- "Connection": Veza između skupa (izdavača, pretplatnika) koji nosi podatke određene teme, koju identificira drugi skup (čvor izdavač, tema, pretplatnički čvor).
- "Service": Sinkroni daljinski poziv procedure.
- "Action": Mehanizam više razine izgrađen na vrh tema i usluga za dugotrajne zadatke ili zadatke koji se mogu unaprijed pripremiti, s posrednom povratnom informacijom pozivatelja.
- "Parametri": „Dijeljeni rječnik s više varijabli koji je dostupan putem mrežnih API-ja.“ Njegova je namjena za sporo mijenjanje podataka, poput argumenata inicijalizacije.
- "roslaunch": Alat naredbenog retka i XML format za koherentno pokretanje niza čvorova, uključujući ponovno mapiranje imena i postavljanje parametara.

Teme su vrlo pogodne za konstantno strujanje podataka, gdje bi svaki pretplatnik trebao dobiti onoliko podataka koliko može obraditi u određenom vremenskom intervalu, a mreža ih može isporučiti. Mrežna latencija relativno je niska jer veze između izdavača i pretplatnika ostaju otvorene. Međutim, važno je imati na umu da poruke automatski nestaju ako se pretplatnički red popuni. Suprotno tome, usluge uspostavljaju novu vezu po pozivu i ne mogu nestati bez obavijesti na strani pozivatelja. Treći mehanizam izgrađen na vrhu tema i usluga su akcije. Akcija koristi usluge za pokretanje i finaliziranje (potencijalno) dugotrajnog zadatka, pružajući tako određene povratne informacije. Između početka i kraja radnje pružaju se kontinuirane povratne informacije putem mehanizma tema.[9]

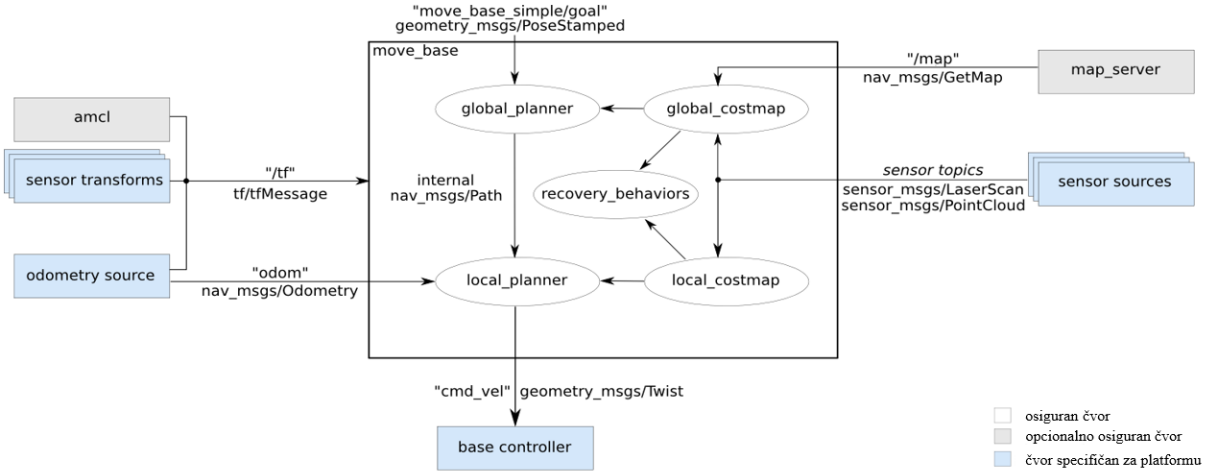
1.4.1. Navigation stack

ROS ima skup korisnih resursa koji pomažu robotu navigirati kroz poznato, djelomično poznato ili nepoznato okruženje, drugim riječima, robot je sposoban planirati i pratiti put dok odstupa od prepreka koje se pojavljuju na tom putu. Ti se resursi nalaze u paketu pod nazivom "Navigation Stack". Na slici 2.3 možemo vidjeti da postoje tri vrste čvorova: osigurani čvorovi, opcionalno osigurani čvorovi te čvorovi specifični za platformu.

- Čvorovi unutar okvira - osigurani čvorovi - jezgra su navigacijskog stacka i uglavnom su odgovorni za upravljanje kartama troškova te funkcionalnostima planiranja puta.
- Opcionalni osigurani čvorovi - amcl i map_server - povezani su s funkcijama statičke karte, što će biti objašnjeno kasnije, a budući da je upotreba statičke karte neobavezna nije obavezna ni upotreba ovih čvorova.
- Čvorovi specifični za platformu su čvorovi povezani s robotom, kao što su čvorovi za očitavanje senzora i čvorovi osnovnog kontrolera

Amcl i Map_server

Amcl i map_server dva su paketa koja su odgovorna za upotrebu statičke karte. Map_server sadrži dva čvora: map_server i map_saver. Prvi je ROS čvor koji pruža



Slika 1.10: Pregled tipičnog sustava koji pokreće Navigation stack[10]

statičke podatke karte kao ROS Service, dok drugi, map_saver, sprema dinamički generiranu kartu u datoteku. Amcl ne upravlja kartama, on je zapravo sustav lokalizacije koji radi na poznatoj karti. Za rad koristi transformaciju base_footprint ili base_link na kartu, stoga mu je potrebna statična karta i ona će raditi tek nakon što je karta izrađena. Ovaj se sustav lokalizacije temelji na pristupu lokalizacije Monte Carlo: on nasumično raspoređuje čestice na poznatoj karti, predstavljajući moguće lokacije robota, a zatim pomoću filtra čestica određuje stvarnu pozu robota. O ovome više u poglavlju 2.2.1.[9]

Gmapping

Gmapping, kao i amcl, sustav je lokalizacije, ali za razliku od amcl, radi u nepoznatom okruženju, izvršavajući simultanu lokalizaciju i mapiranje (SLAM). Stvara 2D kartu zauzetosti mreže koristeći pozu robota i laserske podatke. Radi preko transformacija odom - map, stoga mu nije potrebna karta ni IMU podaci, potrebna je samo odometrija.[9] [13]

Local Costmaps i Global Costmaps

Lokalni i globalni 2D costmapov-a teme su koje sadrže informacije koje predstavljaju projekciju prepreka u 2D ravnini, kao i radijus sigurnosne inflacije, područje oko prepreka

koje jamče da se robot neće sudariti s bilo koji predmetom, bez obzira na njegovu orijentaciju. Te su projekcije povezane s "troškom", a cilj robota je postići navigacijski cilj stvaranjem puta s najmanjim mogućim brojem "troškova". Dok globalna karta troškova predstavlja cjelokupno okruženje (ili njegov veliki dio), lokalna karta troškova, općenito je pomični prozor koji se kreće u globalnoj karti u odnosu na trenutnu poziciju robota. [9]

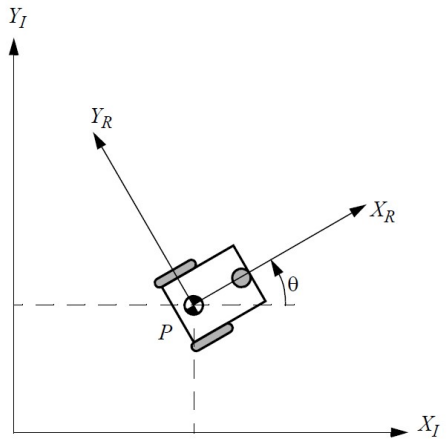
Lokalni i Globalni planeri

Globalni planer zauzima trenutnu poziciju i cilj robota te prati putanju nižih troškova u odnosu na globalnu kartu troškova. Lokalni planer djeluje preko lokalne mape troškova, a budući da je lokalna mapa troškova manja, obično ima više definicije i stoga je u stanju otkriti više prepreka od globalne mape troškova. Prema tome, lokalni je planer odgovoran za stvaranje i uvođenje putanje preko globalne putanje, koji se može vratiti na izvornu putanju s manje troškova, dok odstupa od novo umetnutih prepreka ili prepreka koje definicija globalne mape troškova nije uspjela otkriti. move_base je paket koji sadrži lokalne i globalne planere i odgovoran je za njihovo povezivanje radi postizanja navigacijskog cilja.[9]

2 Teorija

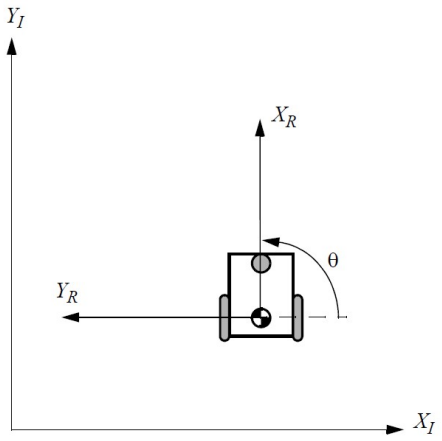
2.1. Kinematika

Kinematika je jedna od osnovnih grana mehanike koja se bavi promatranjem ponašanja mehaničkih sustava. Pri promatranju kinematike mobilnih robota, radni prostor robota predstavlja presudan faktor jer definira raspon mogućih položaja koji se može postići njegovim krajnjim efektorom u odnosu na njegov fiksiran položaj u okolišu. Mogućnost upravljanja mobilnim robotom definira moguće putove i putanje u njegovom radnom prostoru.[14] Dinamika robota postavlja dodatna ograničenja na radni prostor i putanju s obzirom na masu i silu, na primjer, visoko težište ograničava praktični radijus okretanja brzog robota sličnog automobilu zbog opasnosti od prevrtanja. Mobilni je robot samostalni automat koji se u potpunosti može kretati s obzirom na svoje okruženje. Ne postoji izravni način za trenutno mjerenje položaja mobilnog robota. Umjesto toga, čovjek mora integrirati kretanje robota s vremenom. Kada se tome doda netočnost procjene kretanja zbog proklizavanja, jasno je da je precizno mjerenje položaja mobilnog robota izuzetno zahtjevan zadatak. Proces razumijevanja kretanja robota započinje postupkom opisivanja doprinosa svakog kotača u kretanju. Svaki kotač ima ulogu u omogućavanju kretanja cijelog robota. Prema istom principu, svaki kotač također nameće ograničenja na kretanje robota. Kako bi omogućili izražavanje kretnje robota potrebno je uvesti zapis koji će izraziti kretnju robota u globalnom referentnom okviru, kao i lokalni referentni okvir robota. Zatim, potrebno je i demonstrirati konstrukciju jednostavnih kinematičkih modela kretanja prema naprijed, opisujući kako se robot u cjelini kreće u funkciji njegove geometrije i ponašanja pojedinačnog kotača. Dalje, moramo opisati kinematička ograničenja pojedinih kotača, a zatim kombinirati ta kinematička ograničenja kako bi



Slika 2.1: Globalni referentni okvir i lokalni referentni okvir robota [1]

se izrazila kinematička ograničenja cijelog robota. [1]
Izvođenje modela za kretanje cijelog robota postupak je koji se gradi od baze prema gore. Svaki pojedinačni kotačić doprinosi kretanju robota i, istovremeno, nameće ograničenja kretanju robota. Kotači su povezani zajedno na temelju geometrije šasije robota pa se njihova ograničenja kombiniraju i tvore ograničenja na cjelokupno kretanje šasije robota. No sile i ograničenja svakog kotača moraju se izraziti s obzirom na jasan i dosljedan referentni okvir. To je posebno važno za mobilnu robotiku zbog njezine samostalne i mobilne prirode; potrebno je jasno mapiranje između globalnog i lokalnog referentnog okvira. [15] Za mobilnog robota kojeg opisuje ovaj rad, model te veze između modela napravljen je u URDF formatu koji će biti opisan u kasnijim poglavljima.
Kako bismo odredili položaj robota u ravnini, uspostavljamo odnos između globalnog referentnog okvira ravnine i lokalnog referentnog okvira robota, kao što je prikazano na slici 2.1. Osi X_I i Y_I definiraju proizvoljnu osnovu na ravnini kao globalni referentni okvir iz nekog ishodišta $O : X_I, Y_I$. Kako bi se odredio položaj robota, odabire se točka P na šasiji robota kao referentna točka položaja. Osnova X_R, Y_R definira dvije osi u odnosu na P na šasiji robota i prema tome je lokalni referentni okvir robota. Položaj P u globalnom referentnom okviru određen je koordinatama x i y, a kutna razlika između globalnog i lokalnog referentnog okvira zadana je kao Θ. Pozu robota možemo opisati kao vektor s ova tri elementa. [1]
Kako bismo opisali kretanje robota u smislu kretanja komponenata, potrebno je mapirati



Slika 2.2: Mobilni robot poravnat s globalnom osi [1]

kretanje duž osi globalnog referentnog okvira u kretanje duž osi lokalnog referentnog okvira robota. Mapiranje je funkcija trenutne poze robota. Ovo mapiranje postiže se pomoću pravokutne matrice rotacije 2.1:

$$R(\Theta) = \begin{bmatrix} \cos\Theta & \sin\Theta & 0 \\ -\sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

Ova se matrica može koristiti za mapiranje kretanja u globalnom referentnom okviru X_I, Y_I u gibanje lokalnog referentnog okvira X_R, Y_R. Ova je operacija označena s $R(\Theta)\dot{\xi}_I$ jer izračunavanje ove operacije ovisi o vrijednosti Θ [1]:

$$\dot{\xi}_R = R(\frac{\pi}{2})\dot{\xi}_I \tag{2.2}$$

2.1.1. Kinematika mobilnog robota s diferencijalnim pogonom [1]

Ovaj robot diferencijalnog pogona ima dva kotača, svaki s promjerom r. S obzirom na točku P koja se nalazi na središtu između dva pogonska kotača, svaki je kotač udaljen za udaljenost l od točke P. S obzirom na r, l, Θ i brzinu rotacije svakog kotača $\dot{\varphi}_1$ i $\dot{\varphi}_2$, kinematički model predviđa ukupnu brzinu robota u globalnom referentnom okviru:

$$\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\Theta} \end{bmatrix} = f(l, r, \Theta, \dot{\varphi}_1, \dot{\varphi}_2) \tag{2.3}$$

Iz jednadžbe 2.2 znamo da možemo izračunati kretanje robota u globalnom referentnom okviru iz kretanja u njegovom lokalnom referentnom okviru: $\dot{\xi}_I = R(\Theta)^{-1}\dot{\xi}_R$. Postupak nalaže kako je prvo potrebno izračunati doprinos svakog od dva kotača u lokalnom referentnom okviru, $\dot{\xi}_R$.

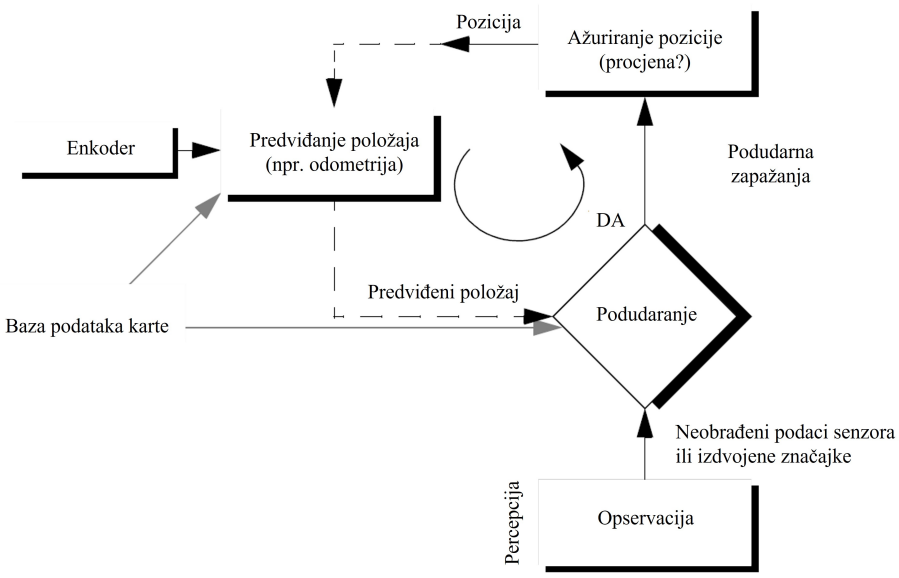
Pretpostavlja se da je lokalni referentni okvir robota poravnat tako da se robot kreće naprijed uzduž $+X_R$, kao što je prikazano na slici 2.2. Za početak, razmatra se doprinos brzine vrtnje svakog kotača brzini translacije u P u smjeru $+X_R$. Ako se jedan kotač okreće, dok drugi kotač ne doprinosi ničemu i miruje, budući da je P na pola puta između dva kotača, instantno će se krenuti kretati s pola brzine: $\dot{x_{r1}} = \frac{1}{2}r\dot{\varphi}_1$ i $\dot{x_{r2}} = \frac{1}{2}r\dot{\varphi}_2$. Kada se računa mobilni robot s diferencijalnim pogonom, kakvog opisuje ovaj rad, tada se dva doprinosa mogu jednostavno dodati za izračun $\dot{x_R}$ komponente $\dot{\xi}_R$. Kada uzmemo u razmatranje diferencijalnog robota kojem se svaki kotač vrti jednakom brzinom, ali u suprotnom smjeru, kao rezultat dobijemo nepokretnog robota koji se vrti u mjestu. Očekivano, $\dot{x_R}$ će u ovom slučaju biti nula. Vrijednost $\dot{y_R}$ još je jednostavnije izračunati. Nijedan kotačić ne može pridonijeti bočnom kretanju u referentnom okviru robota, stoga je $\dot{y_R}$ uvijek nula. Potrebno je izračunati i rotacijsku komponentu. Još jednom, doprinosi svakog kotačića mogu se izračunati neovisno i samo dodati. Razmotrimo desni kotač. Okretanje ovog kotača prema naprijed rezultira rotacijom u smjeru suprotnom od kazaljke na satu u točki P. Ako se jedan kotač okreće sam, robot se okreće oko drugog kotača. Brzina rotacije ω_1 u točki P može se izračunati jer se kotač kreće duž kružnog luka polumjera $2l$:

$$\omega_1 = \frac{r\dot{\varphi}_1}{2l} \tag{2.4}$$

Isti se izračun odnosi i na lijevi kotač, s iznimkom da okretanje prema naprijed rezultira rotacijom u smjeru kazaljke na satu u točki P:

$$\omega_2 = \frac{-r\dot{\varphi}_2}{2l} \tag{2.5}$$

Kombinacijom ovih pojedinačnih formula dobiva se kinematički model robota s diferencijalnim pogonom:



Slika 2.3: Opća shema za lokalizaciju mobilnog robota. [1]

$$\dot{\xi}_I = R(\Theta)^{-1} \begin{bmatrix} \frac{r\dot{\varphi}_1}{2} + \frac{r\dot{\varphi}_2}{2} \\ 0 \\ \frac{r\dot{\varphi}_1}{2l} + \frac{-r\dot{\varphi}_2}{2l} \end{bmatrix} \tag{2.6}$$

2.2. Navigacija

Navigacija je jedan od najizazovnijih zadataka koji se može zadati mobilnom robotu. Uspjeh u navigaciji zahtijeva uspjeh u četiri gradivna bloka navigacije: percepciji (robot mora interpretirati izlazne podatke svojih senzora kako bi izvukao relevantne informacije); lokalizaciji (robot mora odrediti svoj položaj u okolini, slika 2.3); spoznaji (robot mora odlučiti kako će postupiti da bi postigao svoje ciljeve); i kontroli kretanja (robot mora modulirati svoje izlaze motora da bi postigao željenu putanju).[1]

2.2.1. Monte Carlo algoritam [2]

Filter čestica alternativna je neparametrijska provedba Bayesova filter. Filteri za čestice aproksimiraju posteriornu vjerojatnost konačnim brojem parametara, međutim, razlikuju se po načinu na koji se ti parametri generiraju i po načinu na koji naselja-

1: **Algorithm MCL**($\mathcal{X}_{t-1}, u_t, z_t, m$)**:**
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: $x_t^{[m]} =$ **sample_motion_model**$(u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} =$ **measurement_model**$(z_t, x_t^{[m]}, m)$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

Slika 2.4: MCL, ili Monte Carlo Localization, lokalizacijski algoritam zasnovan na filtrima čestica [2]

vaju prostor stanja. Ključna ideja filtera za čestice jest predstavljati posterior $bel(xt)$ nizom nasumičnih uzoraka stanja izvučenih iz spomenutog posteriora. Umjesto da se distribuciju prikazuje parametarskim oblikom (eksponencijalna funkcija koja definira gustoću normalne raspodjele), filtri za čestice predstavljaju raspodjelu nizom uzoraka izvučenih iz ove raspodjele. Takav je prikaz približan, ali je neparametarski i stoga može predstavljati mnogo širi prostor distribucija od, primjerice, Gaussa.

Paket koji simulira navigaciju robota kojeg opisuje ovaj rad koristi upravo algoritam filtera čestica, točnije Adaptivnu Monte Carlo lokalizaciju. Slika 2.4 prikazuje osnovni MCL algoritam. Osnovni MCL algoritam predstavlja "uvjerenje" $bel(xt)$ skupom M čestica $X_t = \{x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]}\}$. Linija 4 u algoritmu na slici 2.4 uzima uzorke iz modela kretanja, pritom koristeći čestice trenutnog "uvjerenja" kao početne točke. Zatim se u liniji 5 primjenjuje model mjerenja snopa kako bi se utvrdila važnost težine te čestice. Početno uvjerenje $bel(x_0)$ dobiva se slučajnim generiranjem M takvih čestica iz prethodne distribucije $p(x_0)$ te dodjelivanjem jedinstvenog faktora važnosti M^{-1} svakoj čestici.

Slika 2.5 prikazuje varijantu MCL algoritma koji dodaje nasumične čestice. Ovaj algoritam je adaptivan zato što prati kratkoročni i dugoročni prosjek vjerojatnosti $p(z_t|z_{t-1}, u_t, m)$. Njegov prvi dio identičan je algoritmu MCL na slici 2.4, međutim, adaptivni MCL izračunava empirijsku vjerojatnost mjerenja u liniji 8 te održava kratkoročne i dugoročne

Algorithm Augmented_MCL($\mathcal{X}_{t-1}, u_t, z_t, m$)**:**
  static $w_{\text{slow}},\ w_{\text{fast}}$
  $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
  for $m = 1$ to M do
    $x_t^{[m]} = \textbf{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$
    $w_t^{[m]} = \textbf{measurement\_model}(z_t, x_t^{[m]}, m)$
    $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
    $w_{\text{avg}} = w_{\text{avg}} + \frac{1}{M}\ w_t^{[m]}$
  endfor
  $w_{\text{slow}} = w_{\text{slow}} + \alpha_{\text{slow}}(w_{\text{avg}} - w_{\text{slow}})$
  $w_{\text{fast}} = w_{\text{fast}} + \alpha_{\text{fast}}(w_{\text{avg}} - w_{\text{fast}})$
  for $m = 1$ to M do
    with probability $\max(0.0,\ 1.0 - w_{\text{fast}}/w_{\text{slow}})$ do
      add random pose to $\mathcal{X}_t$
    else
      draw $i \in \{1, \ldots, N\}$ with probability $\propto w_t^{[i]}$
      add $x_t^{[i]}$ to $\mathcal{X}_t$
    endwith
  endfor
  return $\mathcal{X}_t$

Slika 2.5: Adaptivna varijanta MCL-a koja dodaje slučajne uzorke. Broj slučajnih uzoraka određuje se usporedbom kratkoročne i dugoročne vjerojatnosti mjerenja senzora.[2]

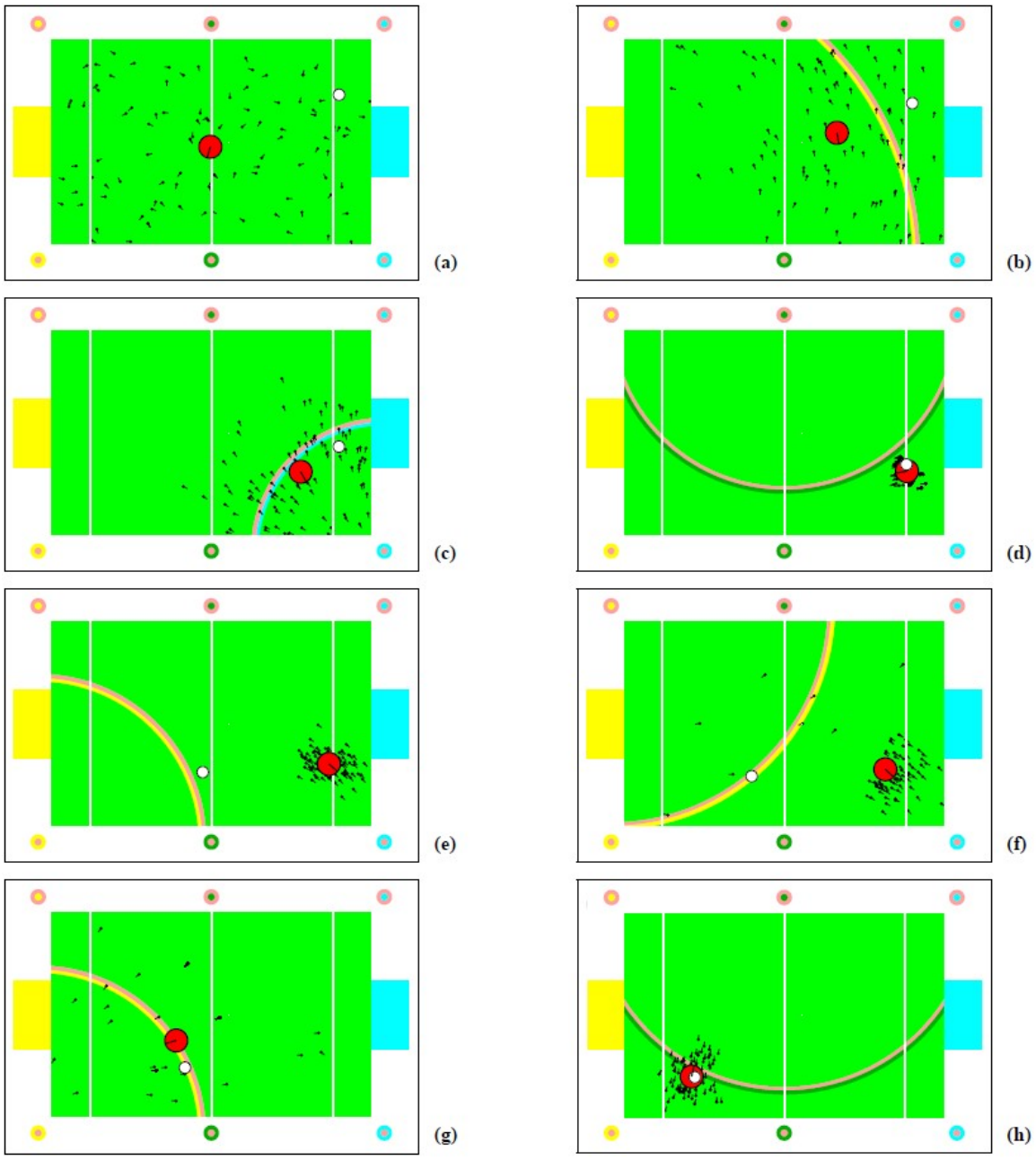
prosjeke ove vjerojatnosti u linijama 10 i 11. Algoritam zahtijeva $0 <= \alpha_{slow} << \alpha_{fast}$. Parametri α_{slow} i α_{fast} stope su propadanja eksponencijalnih filtera koje procjenjuju dugoročni, odnosno kratkoročni prosjek. Suština ovog algoritma može se pronaći u liniji 13: Tijekom postupka ponovnog uzimanja uzoraka dodaje se slučajni uzorak s vjerojatnošću $\max(0.0, 1.0 - \omega_{fast}/\omega_{slow})$; u suprotnom, ponovno uzimanje uzoraka odvija se na uobičajen način. Vjerojatnost dodavanja nasumičnog uzorka uzima u obzir razlike između kratkoročnog i dugoročnog prosjeka vjerojatnosti mjerenja. Ako je kratkoročna vjerojatnost bolja ili jednaka dugoročnoj vjerojatnosti, ne dodaje se nasumični uzorak. Međutim, ako je kratkoročna vjerojatnost lošija od dugoročne, nasumični uzorci dodaju se proporcionalno količniku ovih vrijednosti. Tako, iznenadni pad vjerojatnosti mjerenja inducira povećani broj slučajnih uzoraka. Eksponencijalno zaglađivanje suzbija opasnost od pogreške trenutne buke senzora za loš rezultat lokalizacije. Slika 2.6 ilustrira adaptivni MCL algoritam u praksi.

2.2.2. SLAM

Problem istodobne lokalizacije i mapiranja jedan je od temeljnih problema u robotici. Ovaj se problem obično skraćuje kao SLAM, a poznat je i pod nazivom Istovremeno mapiranje i lokalizacija ili CML. SLAM problemi nastaju kad robot nema pristup karti okoliša, niti ima pristup vlastitim pozama. Umjesto toga, data su samo mjerenja $z_{1:t}$ i kontrole $u_{1:t}$. Izraz "simultana lokalizacija i mapiranje" opisuje rezultirajući problem: u SLAM-u robot dobiva mapu svog okruženja dok se istovremeno lokalizira u odnosu na ovu kartu. SLAM je znatno teži od svih dosad raspravljenih problema s robotikom: teži je od lokalizacije jer je karta nepoznata i mora se usput procijeniti. Teže je od mapiranja s poznatim pozama jer su poze nepoznate i moraju se procijeniti usput.[16] Postoje tri glavne inačice pristupa problemu SLAM-a:

- EKF (eng. Extended Kalman Filters)
- Metoda filtrima čestica
- Tehnike optimizacije zasnovane na dijagramima

Druga glavna, nama zanimljiva, SLAM paradigma temelji se na filtrima čestica. Filtri za čestice predstavljaju posterior dio kroz skup čestica. Za svaku česticu pretpostavkom



Slika 2.6: Lokalizacija Monte Carlo sa slučajnim česticama. Svaka slika prikazuje skup čestica koji predstavlja procjenu položaja robota (male crte označavaju orijentaciju čestica). Veliki krug prikazuje srednju vrijednost čestica, a pravi položaj robota označava mali, bijeli krug. Slike ilustriraju globalnu lokalizaciju (a)-(d) i relokalizaciju (e)-(h).[2]

procjenjujemo koja bi mogla biti istinska vrijednost njenog stanja. Skupljajući mnoštvo takvih nagađanja u skup nagađanja ili u skup čestica, filtar čestica približava se posteriornoj distribuciji. Pod blagim uvjetima pokazalo se da se filter čestica približava stvarnom posteriornom dijelu jer veličina skupa čestica ide u beskonačnost. Također je neparametarski prikaz koji s lakoćom predstavlja multimodalne raspodjele. Ključni problem filtera čestica u kontekstu SLAM-a taj je što je prostor mapa i putova robota ogroman. Filtri za čestice skaliraju se eksponencijalno s dimenzijom osnovnog prostora stanja. Način na koji filtri za čestice postanu prihvatljivi za problem SLAM-a poznat je kao Rao – Blackwellization.[17] Kasnije je skovan naziv fastSLAM (brza simultana lokalizacija i mapiranje). [18] Gmapping, paket unutar ROSa koji se koristio u ovom radu, koristi Rao-Blackwellized filter čestica. Detaljni matematički opis SLAM-a izvan opsega je ovog rada te se čitatelja upućuje na daljnju literaturu, knjigu "Probabilistic Robotics" Sebastiana Thruna.

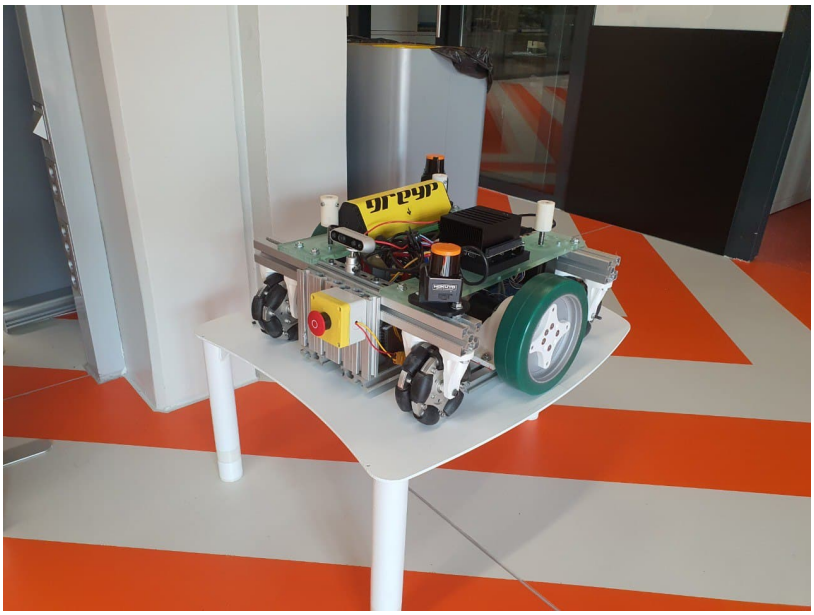
3 Razrada

3.1. Model robota

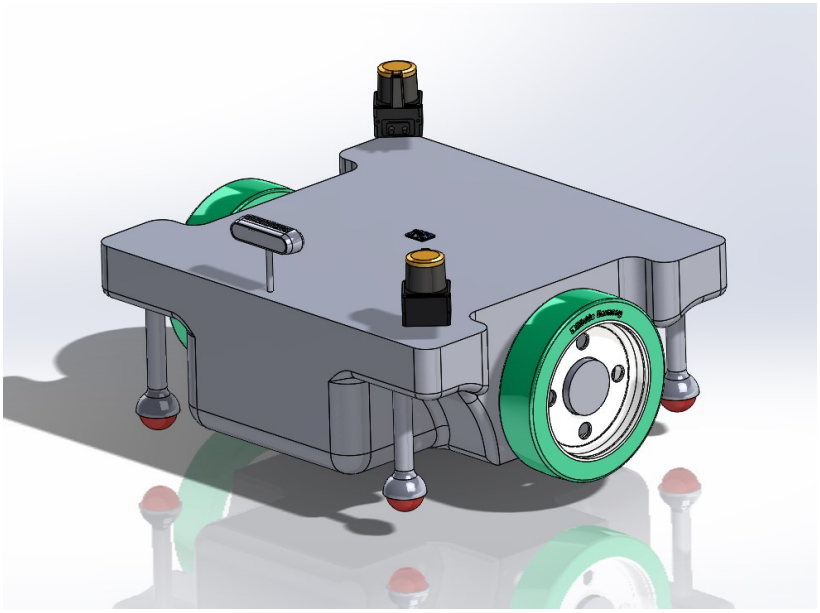
Za valjanu simulaciju prvo je potrebno imati valjani model. Vanjske dimenzije ovog robota određuju vanjske dimenzije RONNA-e, a senzorika koju treba sadržavati bila je predodređena. Robot se sastoji od baze, koja igra ulogu šasije, na koju su montirana dva Blicke GSTA 200/4 kotača. Robot također sadrži dva Hokuyo UST 10LX LiDARa, Intelovu RealSense d435 stereo kameru te Boschovu BNO055 inercijsku mjernu jedinicu. Funkcionalni model robota prikazan je na slikama 3.2 te 3.3. Zadaci su iz ovog modela napraviti URDF datoteku, definirati te priključiti dodatke (*eng. plugin*) za senzore. Generirati okoliš te organizirati radni prostor u ROS-u. Napisati potrebne launch datoteke i upravljati parametrima. Organizirati okolinu unutar rviza te stvoriti kartu. Nakon toga implementirati različite navigacijske algoritme, testirati ih te donijeti zaključak koji je najpogodniji za našu primjenu.

3.1.1. URDF

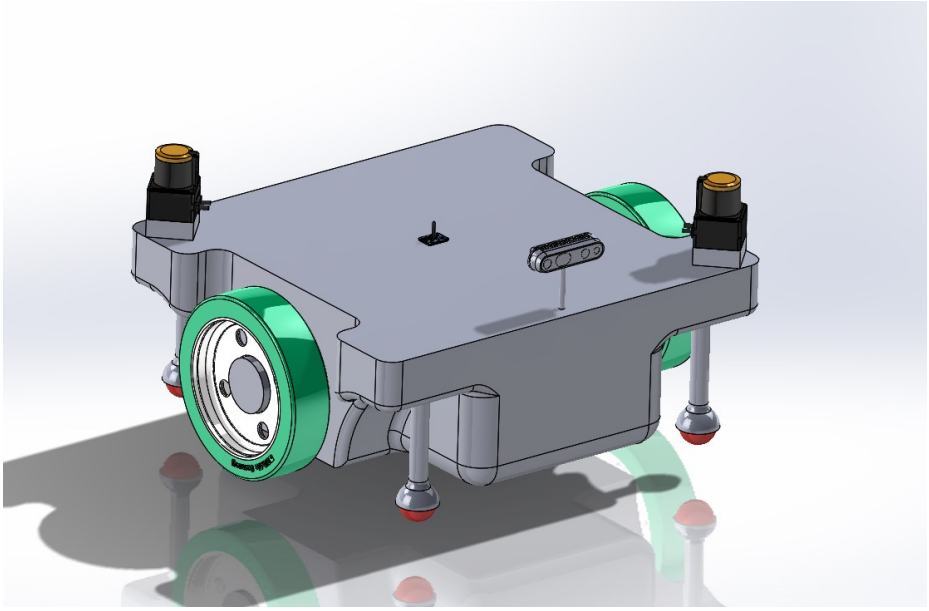
Robot koji je modeliran u prijašnjem poglavlju ima dva kotača povezana na suprotnim stranama šasije te četiri kotačića na svakom kutu šasije, četiri kotačića stabiliziraju robota i slobodno se okreću na temelju kretanja glavnih kotača. URDF je XML format posebno definiran za predstavljanje modela robota do razine komponente. Dvije osnovne URDF komponente koje se koriste za definiranje strukture stabla koja opisuje model robota. Komponenta veze (*eng. link*) opisuje kruto tijelo prema njegovim fizičkim svojstvima (dimenzije, položaj ishodišta, boja itd.). Veze su povezane zglobnim (*eng.*



Slika 3.1: Stvarni prikaz robota



Slika 3.2: 3D funkcionalni model robota



Slika 3.3: Drugi pogled na 3D funkcionalni model robota

joint) komponentama. Zglobne komponente opisuju kinematička i dinamička svojstva veze (to jest, spojene spone, vrste spojeva, os rotacije, količina trenja, prigušenja itd.). URDF opis skup je elemenata veze i skup zglobnih elemenata koji povezuju veze zajedno. Prva komponenta koju opisujemo na robotu je šasija.

Kod 3..1: Opis šasije unutar URDF datoteke

```
<robot name="ronnamobil_kuglicni">
<link name="base_link">
        <inertial>
                <origin xyz="0 0 0" rpy="0 0 0" />
                <mass value="32.662" />
                <inertia        ixx="0.559" ixy="8.2375E-07"
                                ixz="-0.0040674" iyy="0.87949"
                                iyz="-4.3385E-05" izz="0.49361" />
        </inertial>
        <visual>
                <origin xyz="0 0 0" rpy="0 0 0" />
                <geometry>
                        <mesh filename="package://.../base.stl" />
                </geometry>
        </visual>
```

```
        <collision>
                <origin xyz="0 0 0.1" rpy="0 0 0" />
                <geometry>
                        <box size="0.45 0.52 0.14" />
                </geometry>
        </collision>
</link>
</robot>
```

XML kod definira robota nazvanog ronnamobil te ovaj isječak ima jednu vezu (*eng. link*) čija je vizualna komponenta STL datoteka dizajnirana u prethodnom poglavlju koja predstavlja šasiju robota. Šasija će se stvoriti na ishodištu $(0, 0, 0)$ okoliša, bez ikakvih rotacija. Veza je označena s base_link te će ona poslužiti kao temelj stvaranja kinematičkog lanca robota. Uz vizualnu komponentu dodane su i inercijska (*eng. intertial*) te kolizijska komponenta (*eng. collision*). Pri definiranju kolizijske komponente, iako su vizualna svojstva definirana kroz stl datoteku, Gazebov mehanizam za otkrivanje sudara koristi svojstvo sudara za identificiranje granica objekta. Ako objekt ima složena vizualna svojstva (npr. mrežu (*eng. mesh*)), trebalo bi definirati pojednostavljeno svojstvo kolizije kako bi se poboljšale performanse otkrivanja sudara. Unutar inercijskog bloka koda definiramo inercijske značajke našeg modela i masu definiranu u kilogramima. Inercijska matrica šasije robota kojeg opisuje ovaj rad izračunata je s plugin-om unutar okruženja SolidWorks-a. Nakon što je postavljena temeljna komponenta, na nju vežemo ostale. Kada u datoteku URDF dodamo elemente veza, moramo dodati i zglobne elemente kako bismo opisali odnos između veza. Elementi zgloba definiraju je li zglob fleksibilan (pomičan) ili nefleksibilan (fiksan). Za fleksibilne zglobove, URDF opisuje kinematiku i dinamiku zgloba kao i njegove sigurnosne granice. U URDF-u postoji šest mogućih vrsta zglobova od kojih će se u ovom radu koristiti fiksni i kontinuirani. Ovaj postupak pratimo dok ne izradimo cijeli URDF. Vrste zglobnih elemenata su [19]:

- fiksni: svi stupnjevi slobode zaključani. Ova vrsta spoja ne zahtijeva os, kalibraciju, dinamiku, ograničenja ili sigurnosni regulator.
- revolutni: ovaj se spoj okreće oko jedne osi i ima raspon određen gornjom i donjom granicom.
- kontinuirani: ovo je kontinuirani zglobni zglob koji se okreće oko osi i nema gornju i donju granicu.

- prizmatični: ovo je klizni spoj koji klizi duž osi i ima ograničeni raspon određen gornjom i donjom granicom.
- plutajući: ovaj zglob omogućuje kretanje za svih šest stupnjeva slobode.
- planarni: ovaj zglob omogućuje kretanje u ravnini okomitoj na os.

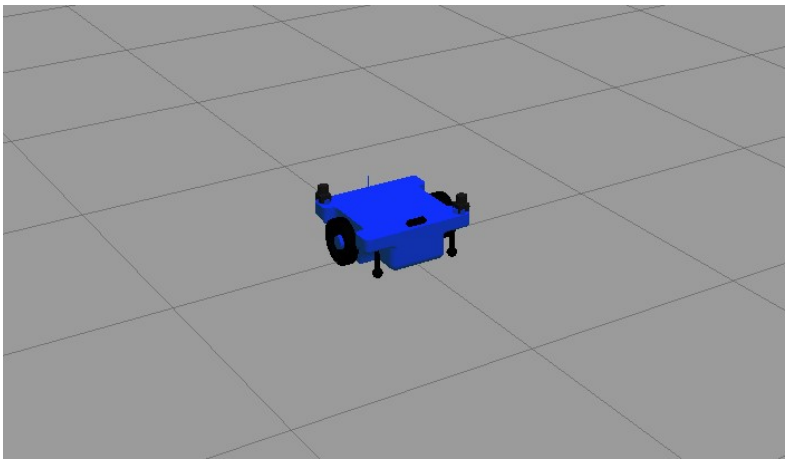
Kod 3..2: Dodavanje zglobne komponente lijevog kotača u URDF datoteku

```
<joint name="leftwheel_joint" type="continuous">
        <origin xyz="0 -0.24 0.08"       rpy="0 0 0" />
        <parent link="base_link" />
        <child  link="left_wheel" />
        <axis   xyz="0 1 0" />
</joint>
```

Gazebo i rviz

Gazebo je besplatno i otvoreno izvorno simulacijsko okruženje robota koje je razvio Willow Garage. Kao višenamjenski alat za programere ROS robota, Gazebo podržava sljedeće[19]:

- Dizajniranje modela robota
- Regresijsko testiranje koristeći realne scenarije
- Simulacija unutarnjeg i vanjskog okruženja
- Simulacija podataka senzora za laserske daljinomjere, 2D/ 3D kamere, senzore u stilu kinect, kontaktne senzore, moment sile itd.
- Napredni 3D objekti i okruženja koja koriste objektno orijentirani grafički mehanizam za prikazivanje (OGRE)
- Nekoliko fizičkih jezgri (*eng. physics engine*) visokih performansi (Open Dynamics Engine (ODE), Bullet, Simbody i Dynamic Animation and Robotics Toolkit (DART)) za modeliranje realne dinamike



Slika 3.4: URDF model s pripadajućim komponentama unutar Gazebo okruženja

Rviz, kratica za ROS vizualizaciju, alat je za 3D vizualizaciju unutar ROS-a. Omogućuje korisniku da vidi simulirani model robota, zapisuje te reproducira podatke s robotovih senzora. Rviz nudi konfigurabilno grafičko korisničko sučelje (GUI) koje omogućuje prikaz podataka koji su relevantni za trenutni zadatak.

3.1.2. Launch datoteke

Launch datoteke formata su .launch i koriste određenu vrstu XML formata. Mogu se smjestiti bilo gdje unutar direktorija paketa, ali najčešće su smještene u direktoriju pod nazivom "launch" unutar direktorija radnog prostora 3.5. Sadržaj datoteke za pokretanje mora biti smješten između par oznaka za pokretanje

```
<launch> ... </launch>
```

Za stvarno pokretanje čvora koriste se oznake ¡node¿, u kojem su potrebni argumenti pkg, type i name.

```
<node pkg = "..." type = "..." name = "..." respawn = true ns = "..." />
```

- pkg/type/name: Argument *pkg* pokazuje na paket pridružen čvoru koji treba pokrenuti, dok se *type* odnosi na ime izvršne datoteke čvora. Također je moguće prepisati ime čvora s *name* argumentom, to će imati prednost nad imenom koje je dodijeljeno čvoru u kodu.

- Respawn / Required: Ako je *respawn = true*, tada će se taj čvor ponovno pokrenuti ako se iz nekog razloga zatvorio. *Required = true* učinit će suprotno, odnosno ugasit će sve čvorove povezane s datotekom za pokretanje ako se ovaj određeni čvor sruši.

- arg: Ponekad je potrebno koristiti lokalnu varijablu u datotekama za pokretanje. To se može učiniti pomoću $< argname = "..." value = "..." >$

Isto tako, launch datoteke uz čvorove mogu pokretati i druge launch datoteke, programe i učitavati i čitati datoteke. Primjer launch datoteke koja se koristi za pokretanje algoritma lokalnog "Timed Elastic Band" planera unutar rviz-a. Uz to što pokreće algoritam planera, datoteka pokreće i Adaptivni Monte Carlo algoritam te rviz u kojeg učitava kartu.

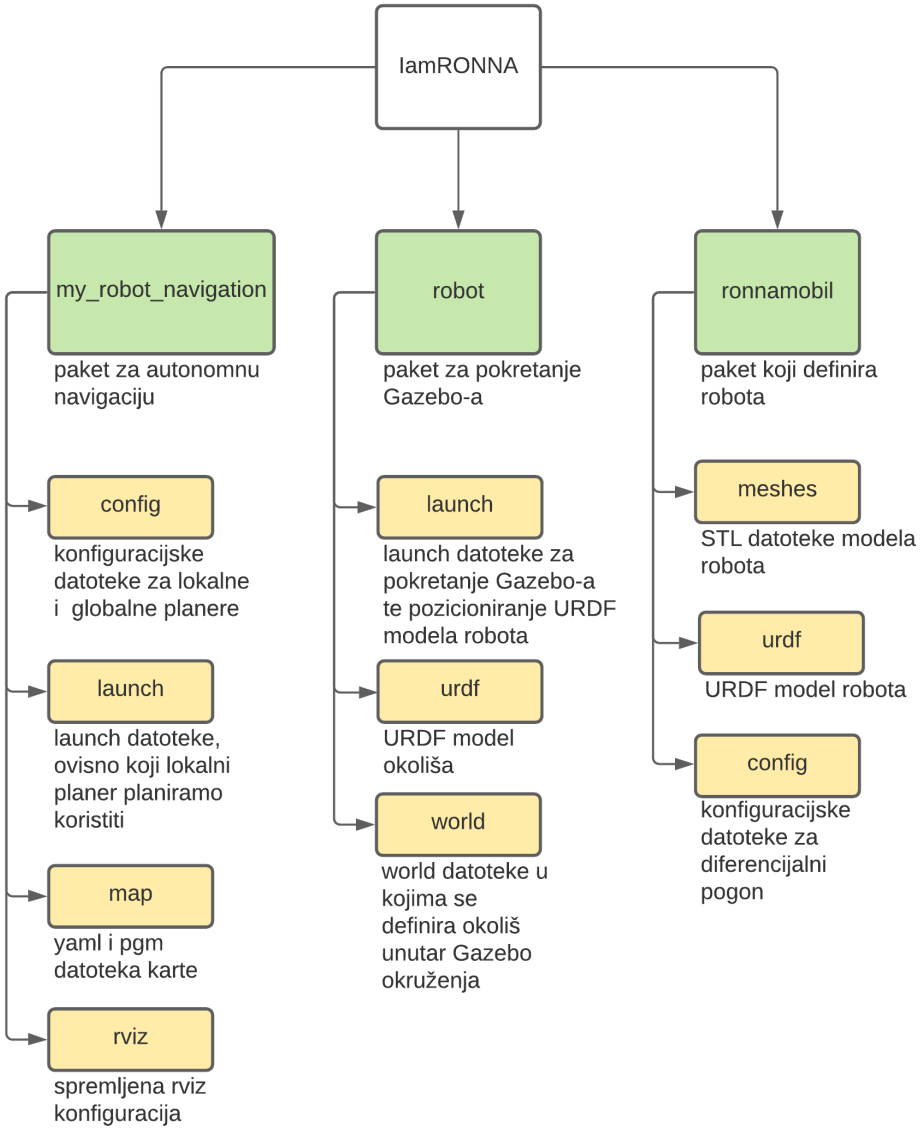
Kod 3..3: Primjer launch datoteke korištene u ovom radu

```
<launch>
        <!-- Arguments -->
        <arg name="map_file" default="$(find my_robot_navigation)/map/map.yaml"/>
        <arg name="open_rviz" default="true"/>
        <arg name="move_forward_only" default="false"/>

        <!-- Map server -->
        <node pkg="map_server" name="map_server"
                type="map_server" args="$(arg map_file)"/>

        <!-- AMCL -->
        <include file="$(find my_robot_navigation)/launch/amcl.launch"/>

        <!-- move_base -->
        <include file="$(find my_robot_navigation)/launch/move_base_teb.launch">
                <arg name="move_forward_only" value="$(arg move_forward_only)"/>
        </include>
        <!-- rviz -->
        <group if="$(arg open_rviz)">
                <node pkg="rviz" type="rviz" name="rviz" required="true"
                        args="-d $(find my_robot_navigation)/rviz/rviz.rviz"/>
        </group>
</launch>
```



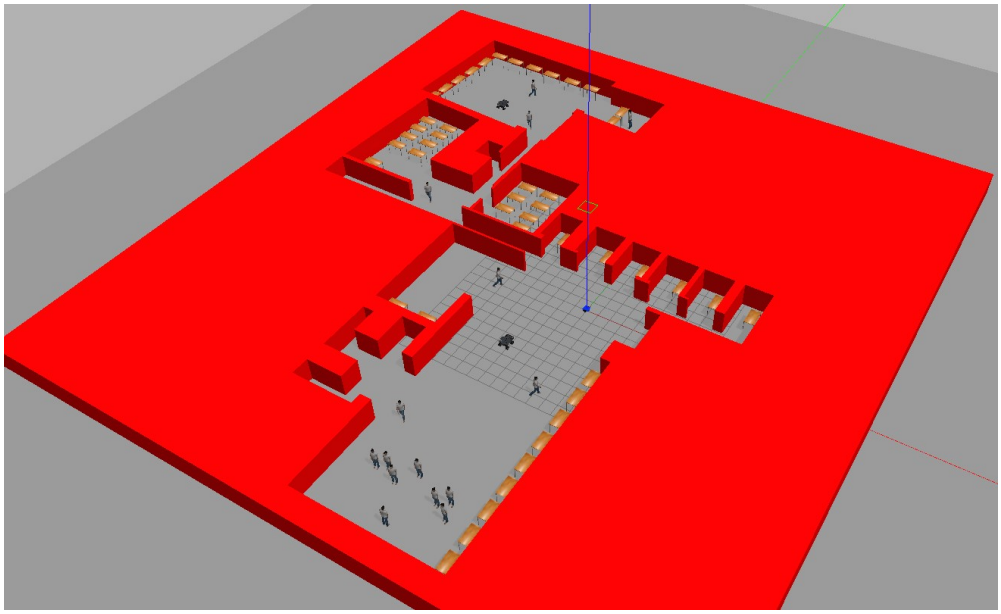
Slika 3.5: Hijerarhija direktorija unutar ROS okruženja

3.2. Izrada karte

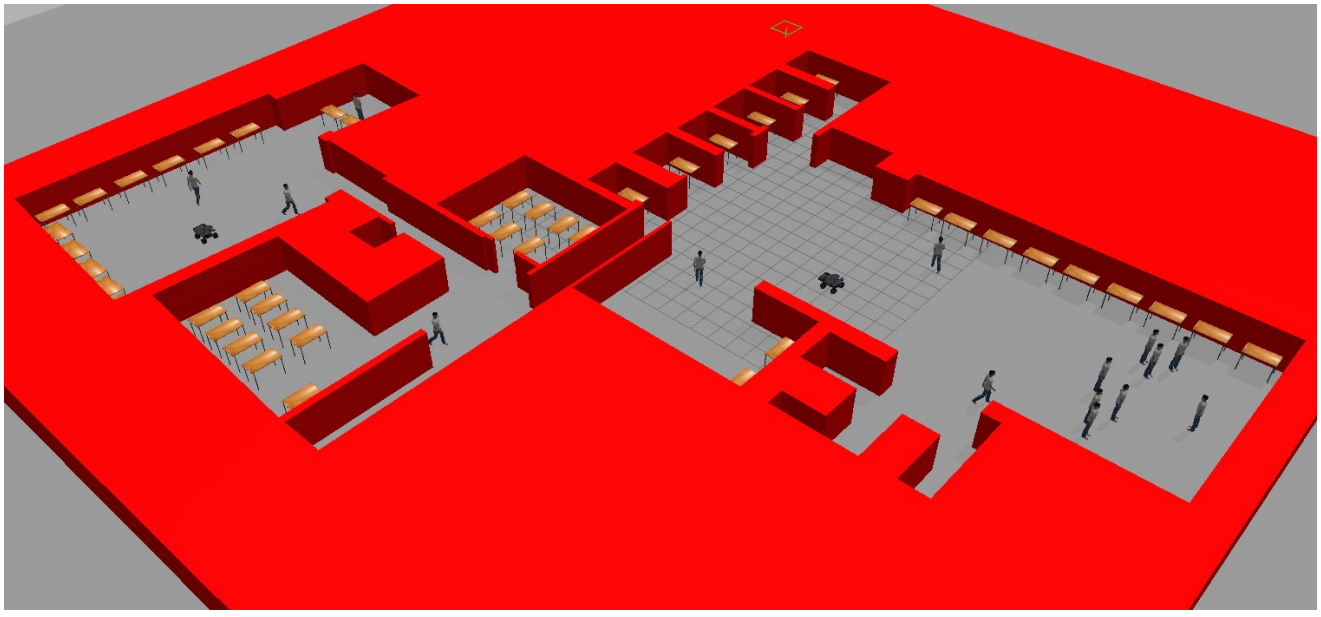
Nakon što je model uspostavljen te uspješno prikazan u Gazebo-u, potrebno je pokrenuti teleop_twist_keyboard paket unutar ROSa. Teleop je skripta koja nam omogućuje ručno upravljanje robotom. Kako je spomenuto u prijašnjim poglavljima, glavni algoritam koji se koristi pri lokalizaciji u ovom radu je AMCL, Adaptivni Monte Carlo algoritam. AMCL zahtijeva statičnu kartu, bilo to pretplaćivanjem na temu koja objavljuje kartu ili zvanjem servisa (*eng. service*) kako bi dobio kartu tako da je kartu potrebno napraviti. Za 3D okruženje unutar Gazebo-a napravljen je, po predlošku tlocrta, model Regionalnog centra izvrsnosti za robotske tehnologije - CRTA, u kojem će robot i u budućnosti raditi. Model se sprema kao .dae datoteka, a zatim se uvodi u ROS okruženje kao .world datoteka. S kombinacijom .launch datoteke i prethodno složenog URDF-a pokrećemo Gazebo unutar kojeg se može vidjeti model robota. Nakon što postoji stabilan okoliš unutar Gazebo-a, potrebno je pokrenuti rviz te početi stvarati vizualizaciju. Na slici 3.10 mogu se vidjeti krajnje odabrane opcije. Za stvaranje karte potrebno je odabrati senzor koji imamo na robotu koji može skenirati prostor. Za robot kojeg opisuje ovaj rad je to LiDAR. Odabire se LaserScan te ga se dodaje u okoliš, na njemu se odabire tema /scan. Nakon što je tema odabrana, trebao bi se vidjeti obris zidova koje skenira LiDAR. Ako se želi vizualizirati model robota, može se dodati RobotModel. Na posljetku je potrebno dodati vizualizaciju karte pod imenom Map, u izborniku ju pretplaćujemo na temu /map te izrada karte može započeti. U terminalu pokrećemo skriptu za teleoperaciju te robota ručno navigiramo po prostoru dok on otkriva i zapisuje podatke. U konačnici se dobije rezultat kao što je prikazan na slici 3.9. Za usporedbu, pogled na 3D model iz ptičje perspektive može se vidjeti na slici 3.8.

3.3. Autonomna navigacija

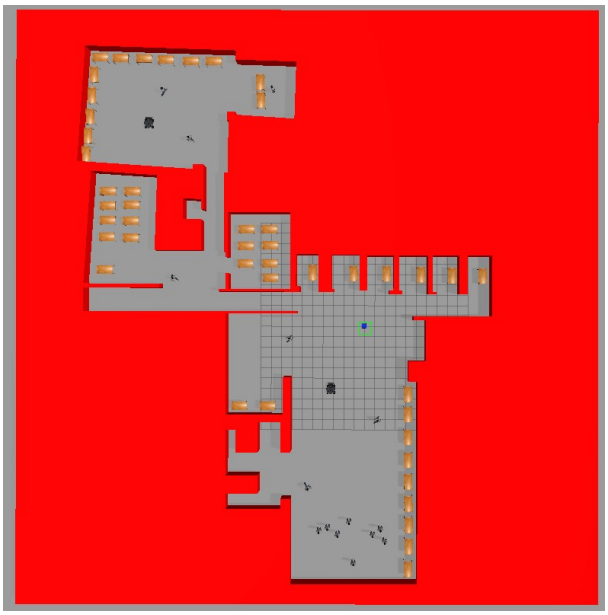
Kao što je već spomenuto, kako bi robot mogao izvesti autonomnu navigaciju, mora imati kartu okoliša koju će koristiti za stvari poput planiranja putanja, izbjegavanja prepreka itd. Postoje mnogi važni parametri pri simulaciji autonomne navigacije mobilnih robota koji dobivaju ili gube na važnosti ovisno o primjeni robota. Robot kojeg opisuje ovaj rad trebao bi nositi RONNA sustav koji teži 320kg, a ukupno težište robota u tom bi se trenutku nalazilo na 900mm. Za pretpostaviti je kako bi i manji trzaj ili neop-



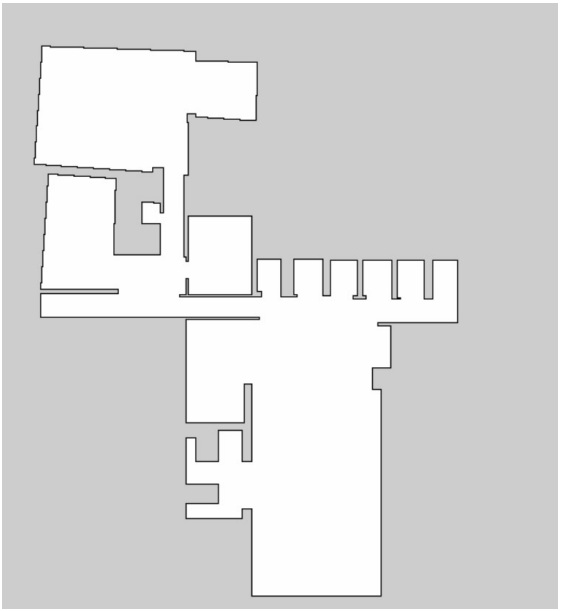
Slika 3.6: 3D prikaz okruženja unutar kojeg će se nalaziti robot



Slika 3.7: 3D prikaz okruženja unutar kojeg će se nalaziti robot iz druge perspektive



Slika 3.8: Pogled na okruženje iz ptičje perspektive



Slika 3.9: Karta stvorena pomoću navigacijskog paketa

timalni pokret uveli nestabilnost na takvu strukturu, dok bi sudar bio katastrofalan i skupocjen.
Najvažnija stavka za ovakav sustav jest algoritam za lokalni planer. U ovom radu simulirana su dva različita algoritma. DWA (*eng. Dynamic Window Approach*) i TEB (*eng. Timed Elastic Band*).

3.3.1. DWA

[20] DWA izvodi optimizaciju temeljenu na uzorku kako bi proizveo par linearnih i kutnih vrijednosti brzine što predstavlja optimalnu kružnu putanju robota. Putanje se simuliraju prema zadanim duljinama horizonta na temelju modela kretanja robota. Zbog konstantne akcije upravljanja duž definiranog horizonta, DWA planer nije u stanju proizvesti kretanje unazad. Princip rada lokalnog planera DWA sastoji se od pet glavnih koraka. Prvi je korak sampliranje brzine u prostoru brzina. Drugo, kratko se vrijeme izvodi simulacija prema naprijed za svaki par brzina kako bi se identificiralo moguće kretanje robota. Kao treći korak rezultati putanje izračunavaju se iz svake simulacije unaprijed pomoću funkcije troškova. Konačno, odabire se najbolja putanja i pridružena vrijednost brzine šalje se osnovnom kontroleru.

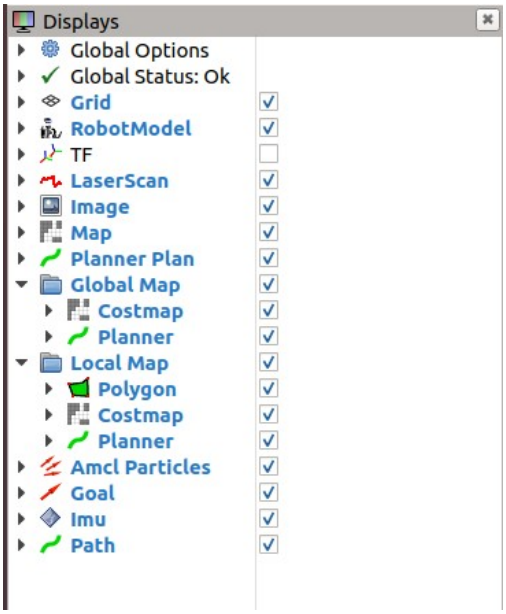
3.3.2. TEB

[20] Lokalni planer TEB implementiran je pomoću algoritma vremenskog elastičnog pojasa koji je modificirani algoritam "elastic band approach". Ova metoda također ima slična svojstva, ali umjesto da koristi sile stezanja i odbijanja, ovaj algoritam optimizira svaki trenutak deformacije putanje i minimalizira funkciju ciljanih troškova. Ovaj algoritam zahtijeva informacije o kinematici, dinamici, geometrijskom obliku, ubrzanju i ograničenjima brzine.

Oba lokalna planera testirana su s istim globalnim i lokalnim parametrima. Budući da je sustav robota težak, ograničenja ubrzanja i brzine postavljena su na niske vrijednosti kako bi sigurnost bila na većoj razini. Osnovni parametri lokalnog planera koji se koriste u ovom eksperimentu prikazani su u tablici 3.1. U lokalnom planeru DWA parametar sim time pruža vremenski interval za optimizaciju putanje. Potrebni uzorci

Tablica 3.1: Parametri za podešavanje lokalnog planera

Kategorija parametra	**Parametar**
Parametri brzine i akceleracije	Maksimalna linearna brzina: $0,4ms^{-1}$ Maksimalna kutna brzina: $0,4rads^{-1}$ Linearno ograničenje akceleracije: $1,5ms^{-2}$ Kutno ograničenje brzine: $1,5rads^{-2}$
Tolerancije cilja	xy tolerancija cilja: $0,22m$ tolerancija kutne rotacije: $0,1rad$
Parametri DWA planera	Sim time: 4 vx_samples: 20 v_theta samples: 40 path distance bias: 32 goal distance bias: 24
Parametri TEB planera	Polumjer otiska: $0,35m$ Minimalna udaljenost od prepreka: $0,6m$ Maksimalna brzina u natrag: $0,1ms^1$

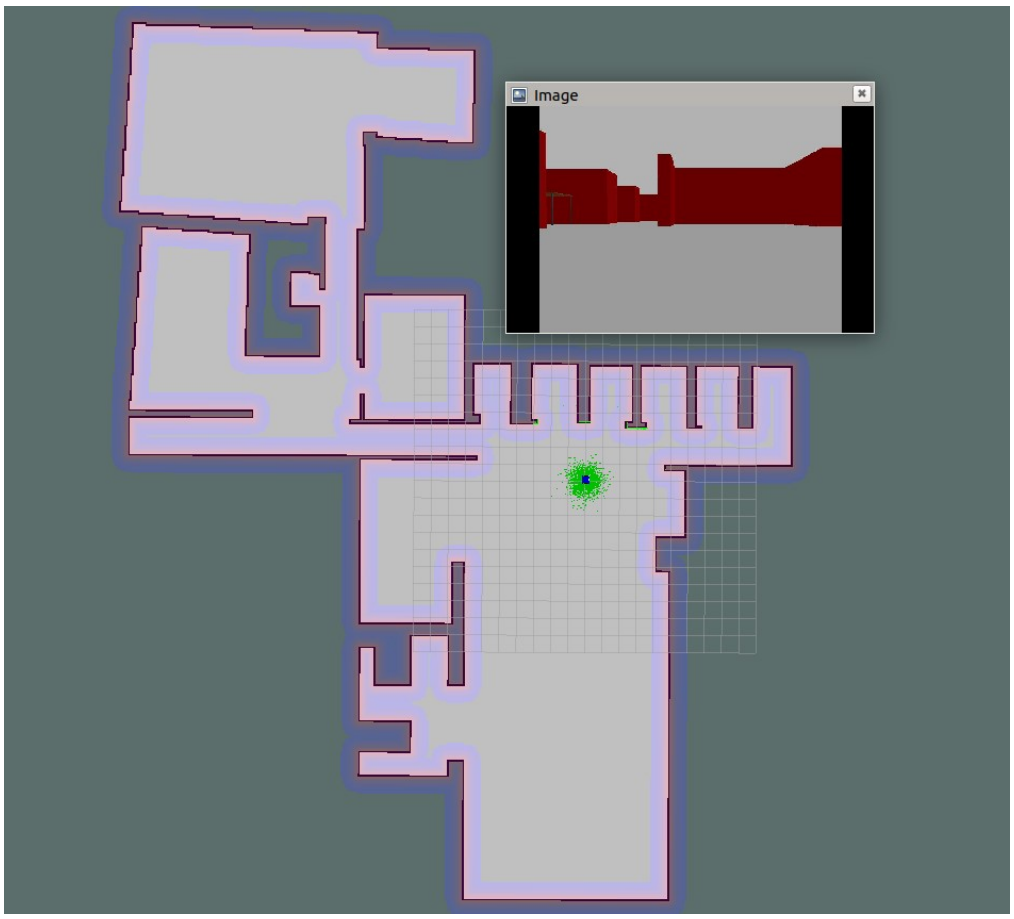


Slika 3.10: Prikaz svih učitanih parametara unutar rviz okruženja

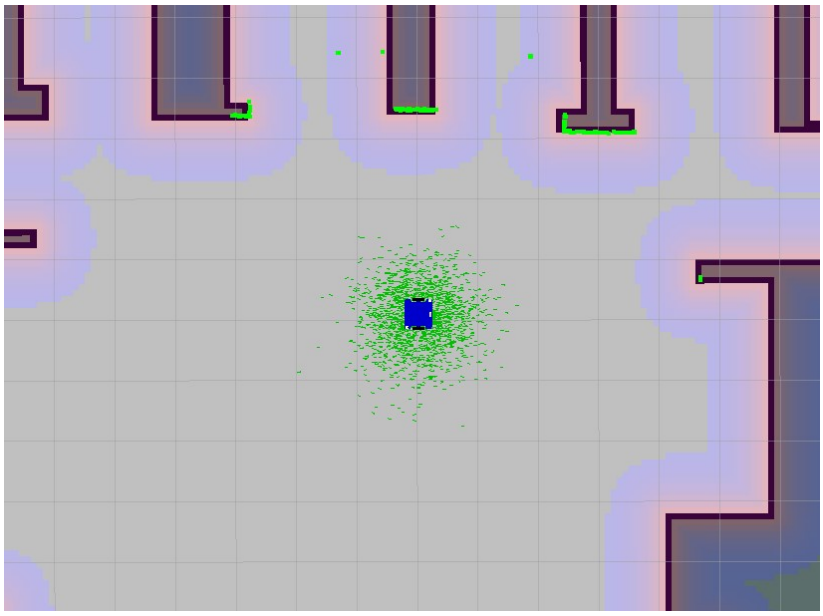
brzine za stvaranje putanje mogu se postaviti pomoću vx_samples i v_theta parametara. Parametri path_distance_bias i goal_distance_bias definiraju koliko lokalni planer ostaje blizu globalnog puta i lokalnog cilja. [21].
Nakon što su svi parametri postavljeni te rviz, kao što je prikazano na slici 3.10 i gazebo pokrenuti, sustav ulazi u fazu simuliranja. Na slici 3.12 možemo vidjeti, u prijašnjim poglavljima objašnjen, Monte Carlo algoritam na djelu. Počinje kao širok snop čestica aproksimirane pozicije i smjera, a kako se simulacija odmiče tako se algoritam sve bolje lokalizira da bi na posljetku bio gust skup, skoro pa uniformnih, čestica. U rvizu vidimo kartu kako je prikazana na 3.11, zajedno sa strujanjem (*eng. stream*) slike s kamere. U ovom se trenutku unutar rviza zadaje Nav Goal, bilo koja lokacija te orijentacija na zadanoj karti do koje će se robot autonomno navigirati.

3.4. Prikupljanje brzina i akceleracija

Kao relevantna metrika usporedbe predloženih planera uzeti su parametri: ukupnog potrebnog vremena za prelazak puta, put prijeđen u metrima, prosječna brzina te su napravljane usporedbe kutnih i linearnih akceleracija. Za prikupljanje ove metrike bilo



Slika 3.11: Prikaz stvorene karte unutar rviz okruženja



Slika 3.12: Prikaz procesa lokalizacije pomoću Monte Carlo algoritma

je potrebno napraviti dva nova čvora u ROS strukturi. Jedan čvor, prikazan u prvom prilogu, koji će mjeriti vrijeme, prijeđen put te prosječnu brzinu te drugi, prikazan u drugom prilogu, koji će mjeriti kutnu te linearnu akceleraciju. Čvorovi su napisani u obliku Python skripti.

3.4.1. Čvor brzina, vremena i prijeđenog puta

U skripti je prvo potrebno definirati, a zatim inicijalizirati klasu. Unutar klase se inicijalizira pretplatnik (*eng. Subscriber*) koji sluša podatke koje objavljuje tema /odom koji potom dolaze kao podaci na callback koji će se preraditi. Unutar callbacka se postavlja da, ukoliko je prvo pokretanje, sustav uzima trenutnu poziciju u obliku koordinata x i y, na koji se nastavno proračunava inkrement prijeđenog puta preko formule $\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$ gdje su x_1 i y_1 trenutni položaji, a x_0 i y_0 početni položaji. Sljedeći blok rješava problem prepoznavanja početka gibanja. Ukoliko je inkrement veći od 5 mm, tajmer se resetira, a stanje kretanja se postavlja na istinito. Potom se ulazi u formulu koja proračunava prijeđenu udaljenost. Kako tema /odom šalje podatke u vrlo kratkim intervalima, potrebno je u svakom intervalu izvršiti pretvorbu iz koordinata u inkrement te svaki dolazeći inkrement zbrojiti sa prošlim kako bi dobili ukupno giba-

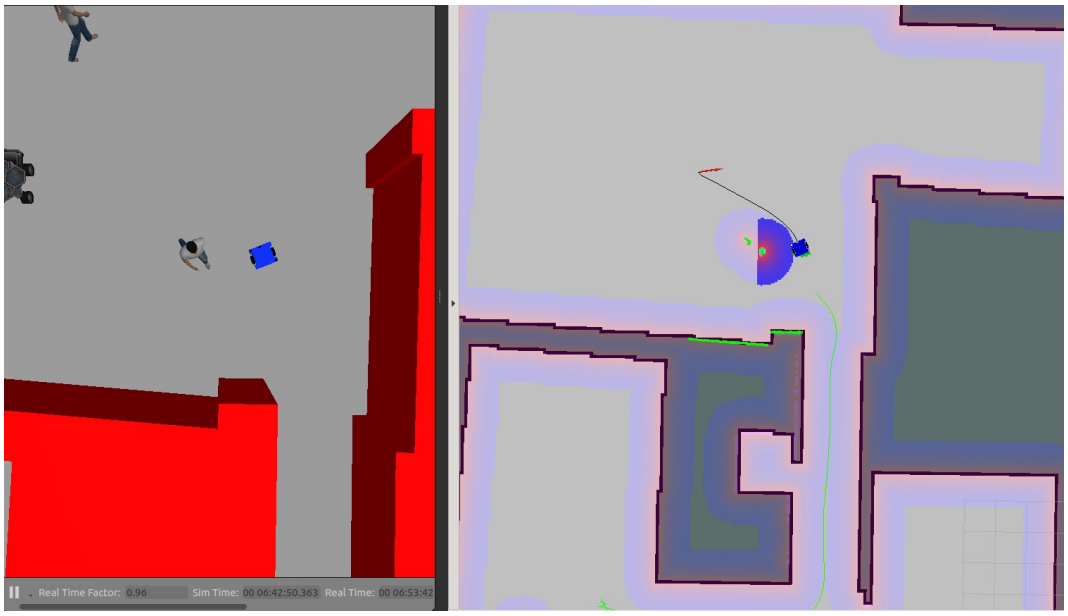
nje. Sljedeći blok rješava problem prepoznavanja zaustavljanja gibanja. Ako je stanje kretanja istinito te ako je inkrement manji od nekog vrlo malog broja (u ovom slučaju $0.005mm$) okida se sekvenca zaustavljanja kretanja. Stanje kretanja se postavlja kao ne istinito te se postavlja krajnje vrijeme. Zatim se proračunava ukupno vrijeme kao $t_konacno - t_pocetno$ te se zbraja ukupan put koji je pređen sa svim pokretima dok je radila skripta. Podaci se kroz set formula preračunavaju i ispisuju, a ulazne se varijable resetiraju. Ukupan kod se nalazi u prvom prilogu.

3.4.2. Čvor kutnih i linearnih akceleracija

Kao i u prvoj skripti, ovdje je također potrebno definirati i inicijalizirati klasu, unutar koje slušamo temu /odom gdje podaci koji će se preraditi dolaze na callback. U prva dva reda callback-a se događaju dvije najvažnije operacije unutar koda. Prva preko ciljanog i izvornog okvira te vremena transformira dobivene podatke sa /odoma u brzine, dok druga brzine pretvara u tražene linearne i kutne akceleracije. Ostatak koda formatira ove informacija u skup informacija koji je moguće interpretirati. Sljedeće dvije varijable su formule koje uzimaju x i y vrijednost dobivenih akceleracija te ih prikazuju kao jedan broj istom metodom korištenom u prvom čvoru. Zatim slijede dva bloka, prvi prepoznaje početak gibanja uspoređujući kutne i linearne akceleracije sa zadanim brojevima. Ukoliko se uvjet zadovolji, stanje kretanja se postavlja na istinito i pokreće se tajmer. Zatim se definira vremenski korak koji će dati podatak koliko je prošlo vremena od zadnje objave na temi /odom. Transformirani podaci sa odoma zajedno sa vremenskim korakom se zajedno stavljaju u listu. Drugi blok prepoznaje zaustavljanje gibanja uspoređujući trenutne vrijednosti linearne i kutne akceleracije sa zadanim brojevima. Zadani brojevi su eksperimentalno dobiveni. Ostatak koda stvara datoteke .csv formata, iz spremljenih lista, za kutnu te linearnu akceleraciju. Ukupan kod se nalazi u drugom prilogu.

4 Rezultati

Na slici 3.8 prikazan je izgled simulacijske okoline. Zidovi su postavljeni crvenom bojom te su integrirani unutar statičke karte i sustav će ih izbjegavati kao statičke prepreke. Ostali modeli, stolovi, ljudi te roboti su postavljeni kao dinamičke prepreke. Sustav za njih ne zna, nego mora reagirati te prilagoditi putanju u trenutku. Na slici 4.1 je prikazana usporedba modela vidljivog u Gazebo simulatoru te ostvarenu prilagodbu putanje unutar rviza. Na karti je postavljeno devet kontrolnih točaka. Polazište je u prostoriji kod točke 1 4.3, robot zatim prolazi kroz uski prolaz do druge prostorije i točke 2 gdje je potrebno zaobići jednu dinamičku prepreku. Robot kreće iz točke dva u točku 3, kroz drugi uski hodnik, gdje izlaskom iz hodnika odmah nailazi na dinamičku prepreku koju je potrebno zaobići. Slijedi putanja od točke 3 do točke 4. Ovaj dio je zanimljiv jer osim što je jedna od dužih relacija, relacija je s najviše zavoja. Ovdje ćemo uspoređivati dijagrame kutnih akceleracija. Točka 4 je natrag u početnoj poziciji a do točke 5 robota dijeli dinamička prepreka. Put od točke 5 do točke 6 je zanimljiv zato što je potrebno proći između statičke i dinamičke prepreke pritom zadovoljavajući uvjete najmanjeg dopuštenog razmaka. Putanja između točke 7 i 8 je najizazovnija. Robot treba izbjeći grupu od 9 ljudi koja predstavlja vrlo kompleksnu dinamičku prepreku. Kako je prošla putanja izazovna zbog manjka prostora, putanja od toče 8 do 9 je izazovna zbog viška prostora. Na ovoj putanji se testira sposobnost lokalizacije, kao što je žutom strelicom prikazano na slici 4.3 amcl algoritam raširi skup čestica u pokušaju lokalizacije dok ne dođe do zida kojeg prepozna iz statičke karte. Posljednja putanja, između točaka 8 i 9 predstavlja pravocrtan potez s jednog kraja karte na drugi. U ovom dijelu će biti prikazana usporedba linearnih akceleracija. Kao što je već spomenuto, promatraju se



Slika 4.1: Usporedba vidljive prepreke u Gazebo simulatoru te ostvarene prilagodbe putanje unutar rviz-a.

dva slučaja. Prvi s TEB algoritmom kao lokalnim planerom, drugi s DWA. Sva se testiranja provode tri puta kako ne bi došlo do nasumičnih ekstremnih slučajeva.

Na slikama 4.3, 4.2 možemo vidjeti rezultat testiranja. Kao što se na prvu može primjetiti, putanje nisu bitno različite jedna od druge.

4.0.1. DWA

DWA algoritam, čiju ukupnu putanju možemo vidjeti na slici 4.2 svaki je put uspješno završio putanju te svaki put imao identičnu putanju. Planer također ima najkonzistentnija vremena između kontrolnih točaka. Nedostatak u usporedbi sa TEB algoritmom se primjećuje pri okretanju za 180°. Dok TEB algoritam ima razređenu strategiju za okret u kojoj se pomakne unazad pod 90° te potom unaprijed pod 90°, DWA se algoritam najčešće okrene u mjestu zbog nemogućnosti pomicanja unazad.

4.0.2. TEB

TEB algoritam, čiju ukupnu putanju možemo vidjeti na slici 4.3 isto je kao i DWA uspješno završio svaku od tri putanje, te kao što se može uočiti na dijagramu 4.9 zadatak

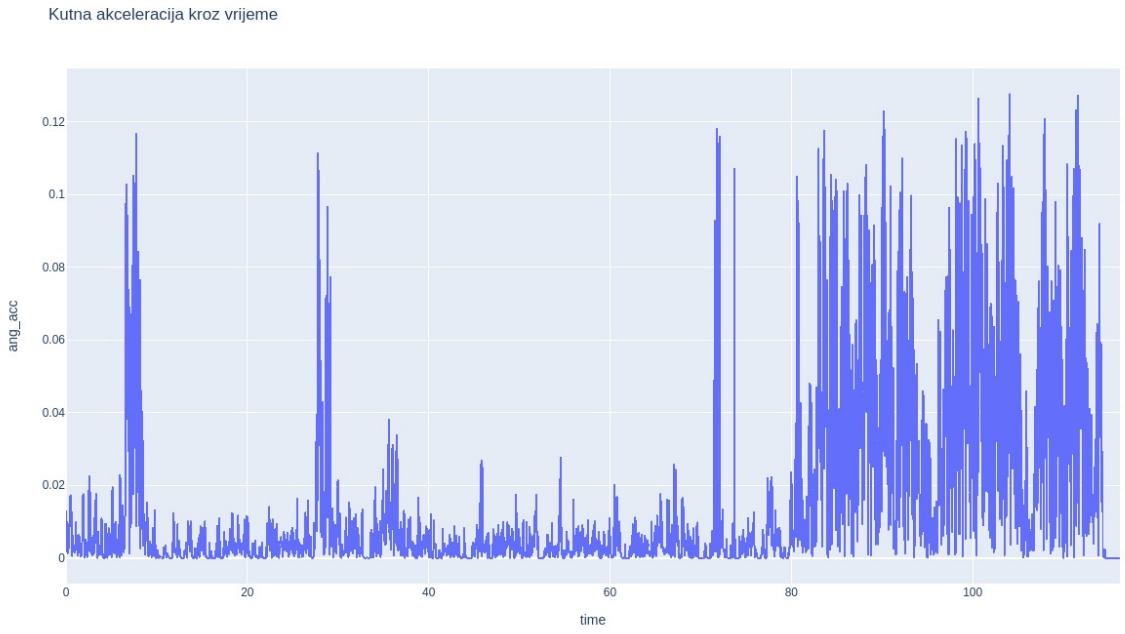
Slika 4.2: Prikaz putanje DWA algoritma za lokalno planiranje

je u prosjeku odradio u najbržem vremenu. TEB algoritam ima specifičnu kvalitetu kakva nije primijećena kod DWA algoritma. Naime, kako se približava prepreci ili zidu postupno usporava te vrlo oprezno i naizgled optimalno zaobilazi, obilježava ga glatkoća pokreta te glatkoća ulaza i izlaza u zavoje Također, iz dijagrama 4.8 se može vidjeti kako TEB algoritam ima sveukupno kraću putanju od DWA algoritma, time upućujući na optimalnost pri odabiru putanje.

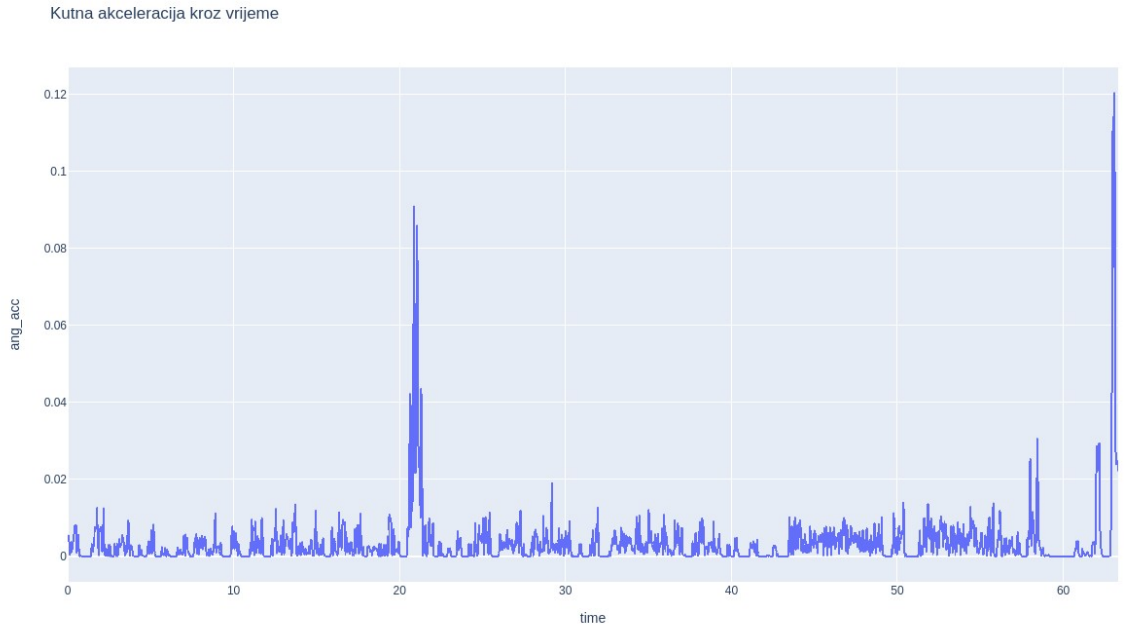
4.0.3. Analiza rezultata

Uspoređujući kutne akceleracije algoritama, točno se može vidjeti kada i kojom glatkoćom je koji planer ulazio u zavoje. DWA algoritam, kojeg se može vidjeti na slici 4.4, ima izražene i učestalije vrhunce te općenito manje strukturirano kutno ubrzanje, što ukazuje na robusnost i učestalu potrebu za ispravljanjem gibanja. TEB algoritam, koji je prikazan na slici 4.5, obilježava mirnije kutno ubrzavanje sa dva veća vrhunca. Jedan pri ulasku u zavoj, a drugi pri konačnom pozicioniranju. DWA algoritam isto tako obilježavaju općenito veći iznosi kutnog ubrzanja. Drugi par dijagrama, prikazani na slikama 4.6 te 4.7, pokazuje linearno ubrzanje pri pravocrtnom gibanju. Pri oba dijagrama se da uočiti sličan uzorak ubrzanja, gdje im se vrhunci skoro poklapaju. Iz priloženog se da zaključiti kako je DWA algoritam ubrzavao pet puta, dok je TEB ubrzavao četiri puta. TEB algoritam ima veća početna ubrzanja, dok DWA algoritam ima jedno ubrzanje više pred kraj. Iz svih dijagrama se može zaključiti kako su algoritmi poštivali zadane parametre ograničenja ubrzanja.

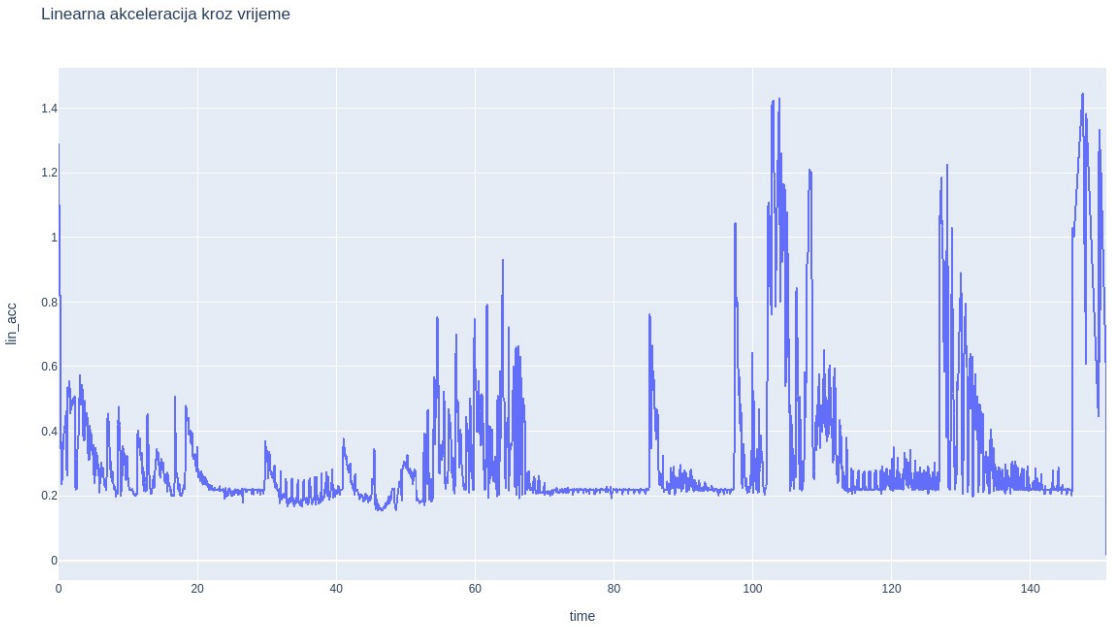
Slika 4.3: Prikaz putanje TEB algoritma za lokalno planiranje



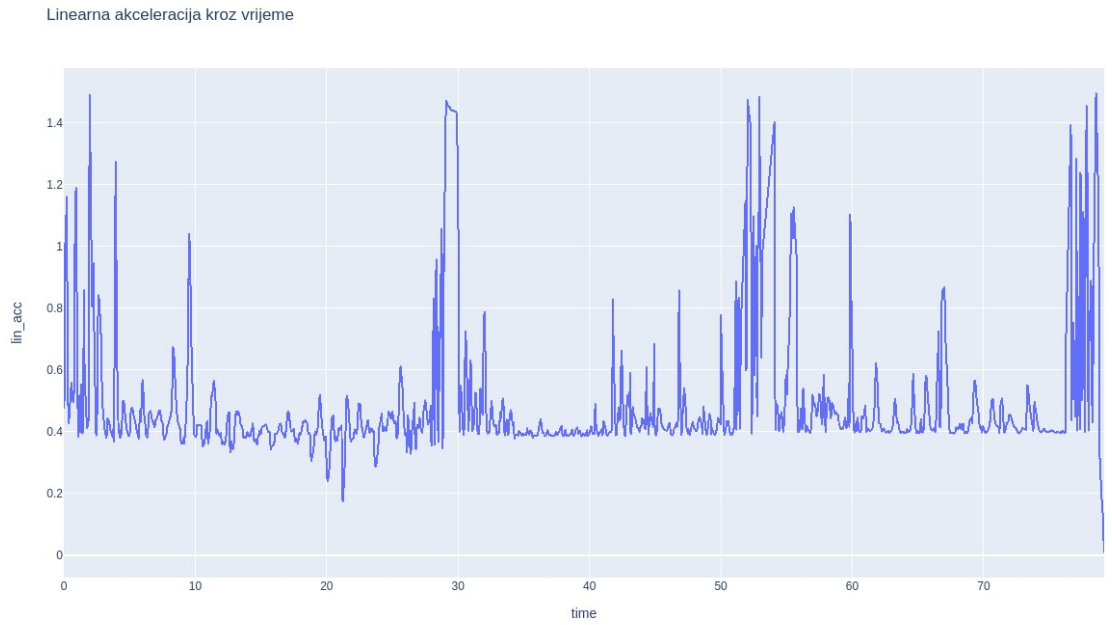
Slika 4.4: Kutna akceleracija DWA algoritma između točaka 2 i 3



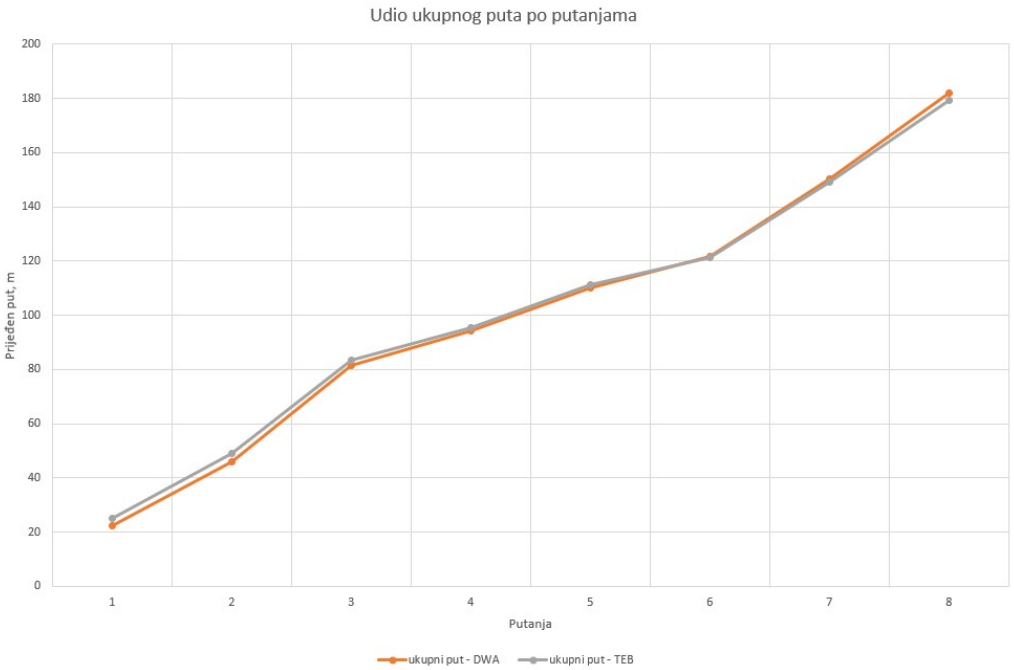
Slika 4.5: Kutna akceleracija TEB algoritma između točaka 2 i 3



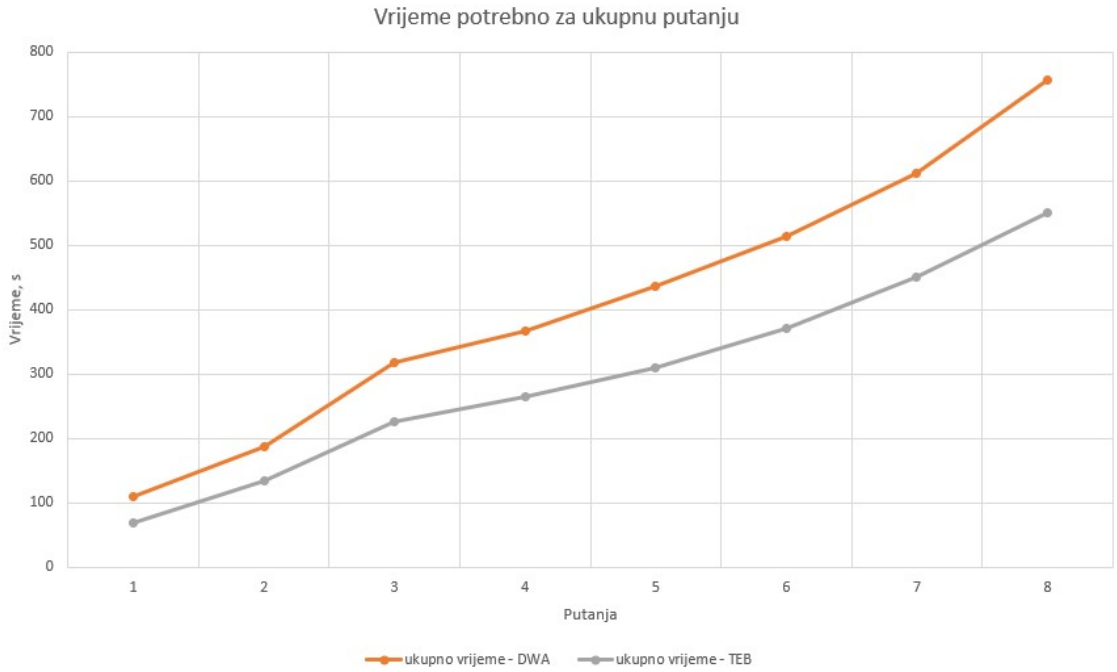
Slika 4.6: Linearna akceleracija DWA algoritma između točaka 8 i 9



Slika 4.7: Linearna akceleracija TEB algoritma između točaka 8 i 9



Slika 4.8: Usporedba prijeđenog puta po putanjama



Slika 4.9: Usporedba potrebnog vremena za sve putanje

5 Zaključak

Glavni cilj ovog rada bio je uspostaviti okvir koji će omogućiti nastavak istraživanja mobilnosti RONNA-e. Okruženje je uspješno uspostavljeno te su algoritmi uspješno implementirani. Drugi problem kojim se ovaj rad bavi je analiza i verifikacija algoritma za lokalno planiranje putanje u raznim konfiguracijama. Odabrano je devet različitih konfiguracija kroz koje su analizirane performanse lokalnih planera ROS-a s pretpostavkom da je na mobilnom robotu s diferencijalnim pogonom težak teret s visokim centrom mase. Nakon provođenja testova, skupljanja metrike i analize rezultata, može se zaključiti kako su lokalni planeri TEB i DWA vrlo učinkoviti pri razvoju strategija planiranja puta. Međutim, kada se ova dva lokalna planera uspoređuju međusobno, niti jedan se ne nameće kao jasan izbor. Svaki od planera ima svoje mane i prednosti te ih treba razmotriti pod različitim kriterijima. Kada se izbor sagleda sa strane ponovljivosti lokalni planer DWA nameće se kao bolji izbor, no što se tiče izbjegavanja prepreka te glatkoće kretanja, kao i konzistentnosti ubrzanja, lokalni planer TEB izlazi kao najbolja opcija. Uzevši u obzir da mobilnost RONNA-e na prvom mjestu zahtijeva sigurnost radnog okoliša te sigurnost samog robota, isto tako i glatkoću kretanja te okretanja, TEB lokalni planer može smatrati boljom opcijom.

A Prvi prilog

```
#!/usr/bin/env python
import rospy
from math import sqrt
from tf.transformations import euler_from_quaternion
from nav_msgs.msg import Odometry
import time


class OdometryModifier:

        def __init__(self):
                self.sub = rospy.Subscriber("odom", Odometry, self.callback)
                self.total_distance = 0. # ukupna udaljenost u metrima
                self.distance_per_move = 0. #ukpuna udaljenost po pokretu
                self.previous_x = 0.
                self.previous_y = 0.
                self.first_run = True

                self.moving = False
                self.start_time = None
                self.end_time = None

        def callback(self, data):
                if(self.first_run):
                        self.previous_x = data.pose.pose.position.x
                        self.previous_y = data.pose.pose.position.y
                x = data.pose.pose.position.x
```

```
                y = data.pose.pose.position.y
                d_increment = sqrt((x - self.previous_x) * (x - self.previous_x) +
                        (y - self.previous_y) * (y - self.previous_y))
                if(d_increment > 0.005):
                        if(not self.moving):
                                self.start_time = time.time()
                                print("Kretanje je zapocelo!")

                        self.moving = True
                        self.distance_per_move = self.distance_per_move + d_increment

                if(self.moving and d_increment < 0.000005):
                self.moving = False
                self.end_time = time.time()
                time_elapsed = (self.end_time - self.start_time)
                self.total_distance = self.total_distance + self.distance_per_move
                print("Ukupan prijeden put iznosi: " + str(self.total_distance))
                print("Put prijeden od zadnje stanice: " +
                        str(self.distance_per_move))
                print("Proteklo vrijeme: " + str(time_elapsed))
                print("Prosjecna brzina: " +
                        str(self.distance_per_move / time_elapsed))

                self.distance_per_move = 0.

        self.previous_x = data.pose.pose.position.x
        self.previous_y = data.pose.pose.position.y
        self.first_run = False


if __name__ == '__main__':
        try:
                rospy.init_node('put', anonymous=True)
                odom = OdometryModifier()
                rospy.spin()
        except Exception as e:
                print(e)
```

B | Drugi prilog

```
#!/usr/bin/env python
import rospy
from math import sqrt
from tf.transformations import euler_from_quaternion
from tf import TransformListener
from nav_msgs.msg import Odometry
import time
import csv
import sys


class AccelerationMonitor:

        def __init__(self):
                self.sub = rospy.Subscriber("odom", Odometry, self.callback)
                self.first_run = True
                self.listener = TransformListener()
                self.lin_acc_data = []
                self.ang_acc_data = []
                self.moving = False
                self.move_start_time = None

        def callback(self, data):
                (trans, rot) = self.listener.lookupTransform('/base_link',
                '/odom', rospy.Time(0.0))
                (lin, ang) = self.listener.lookupTwistFull('/odom', '/base_link',
```

```
                '/odom', (0,0,0), '/map', rospy.Time(0.0), rospy.Duration(0.1))

                lin_al = sqrt((lin[0] * lin[0]) + (lin[1] * lin[1]))
                ang_al = sqrt((ang[0] * ang[0]) + (ang[1] * ang[1]))
                if (lin_al > 0.01 or ang_al > 0.005):
                        if (not self.moving):
                                print('Krecemo!')

                                self.move_start_time = time.time()

                        self.moving = True
                        now = time.time()

                        timestamp = now - self.move_start_time
                        self.lin_acc_data.append((lin_al, timestamp))
                        self.ang_acc_data.append((ang_al, timestamp))


                if (lin_al < 0.01 and ang_al < 0.001 and self.moving):
                        print('Stali smo')
                        header = ['lin_acc', 'time']
                        with open('/home/ros_melodic/catkin_ws/src/distance/
src/lin_acc_data_mbs_1.csv', 'w') as f:
                                write = csv.writer(f)
                                write.writerow(header)
                                write.writerows(self.lin_acc_data)

                        header = ['ang_acc', 'time']
                        with open('/home/ros_melodic/catkin_ws/src/distance/
src/ang_acc_data_mbs_1.csv', 'w') as f:
                                write = csv.writer(f)
                                write.writerow(header)
                                write.writerows(self.ang_acc_data)
                        sys.exit()



if __name__ == '__main__':
        try:
```

```
        rospy.init_node('distance_tracker', anonymous=True)
        odom = AccelerationMonitor()
        rospy.spin()
except Exception as e:
        print(e)
```

Literatura

[1] Roland Siegwart. *Introduction to Autonomous Mobile Robots*, pages 57–423. MIT Press, 2011.

[2] Sebastian Thrun. *Probabilistic Robotics*, pages 77–265. MIT Press, 2005.

[3] Bojan Jerbić Marko Švaco, Darko Chudy. RONNA G4—Robotic Neuronavigation: A Novel Robotic Navigation Device for Stereotactic Neurosurgery. 2020.

[4] Boston Dynamics. Spot, https://shop.bostondynamics.com/spot, 28.6.2021.

[5] Keiji Nagatani Kenjiro Tadakuma, Riichiro Tadakuma. Crawler Vehicle with Circular Cross-Section Unit to Realize Sideways Motion. 2008.

[6] Spyros G. Tzafestas. *Introduction to Mobile Robot Control*, pages 13–537. Elsevier, 2014.

[7] Michael Jenkin Gregory Dudek. *Handbook of Robotics*, pages 737–751. Springer, 2016.

[8] Hokuyo. Hokuyo ust-10lx, https://hokuyo-usa.com/products/lidar-obstacle-detection/ust-10lx, 3.7.2021.

[9] Rodrigo Longhi Guimarães. *Robot Operating System (ROS): The Complete Reference (Volume 1)*, pages 33–161. Springer, 2016.

[10] Navigation Stack, http://wiki.ros.org/navigation/Tutorials/RobotSetup, 1.7.2021.

[11] Karl Iagnemma Woojin Chung. *Handbook of Robotics*, pages 575 –593. Springer, 2016.

[12] Joel Lawhead. *Learning Geospatial Analysis with Python - Third Edition*. Packt Publishing Ltd, 2019.

[13] Giorgio Grisetti; Cyrill Stachniss; Wolfram Burgard. gmapping paket, https://openslam-org.github.io/gmapping.html, 1.7.2021.

[14] J.J. Craig. *Introduction to Robotics: Mechanics and Control*, page 46. Addison-Wesley, 1989.

[15] Bastin G. Campion, G. Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. 1996.

[16] Bailey T. Durrant-Whyte, H. Simultaneous Localization and Mapping: Part I. 2006.

[17] Freitas N. Doucet, A. Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. 2000.

[18] Sebastian Thrun Cyrill Stachniss, John J. Leonard. *Handbook of Robotics*, pages 1154–1171. Springer, 2016.

[19] Thomas Harman Carol Fairchild. *ROS Robotics By Example*, pages 34–71. Packt Publishing Ltd, 2016.

[20] Isira Naotunna; Theeraphong Wongratanaphisan. Comparison of ros local planners with differential drive heavy robotic system. 2020.

[21] Kaiyu Zheng. Ros navigation tuning guide. 2017.