Simulacija robota hodača u Gazebo simulatoru

Grabar, Petar

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Petar Grabar

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

Simulacija robota hodača u Gazebo simulatoru

Mentori:

Doc. dr. sc. Marko Švaco

Student:

Petar Grabar

Zagreb, 2023.

Zahvaljujem se doc.dr.sc. Marku Švaci te Branimiru Ćaranu mag.ing.mech. na pomoći pri realizaciji i ideji ovog rada. Želim se također zahvaliti i svim djelatnicima i asistentima fakulteta na ugodnom boravku i suradnji tokom preddiplomskog studija. Posebno bih se zahvalio svojoj obitelji, djevojci i kolegama na podršci koju su mi ukazali tijekom studija.

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

Zagreb, veljača 2023. Petar Grabar

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

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 23 – 6 / 1**	
Ur.broj: **15 - 1703 - 23** -	

ZAVRŠNI ZADATAK

Student: **Petar Grabar** JMBAG: **0035220823**

Naslov rada na hrvatskom jeziku: **Simulacija robota hodača u Gazebo simulatoru**

Naslov rada na engleskom jeziku: **Simulation of a quadruped robot in the Gazebo simulator**

Opis zadatka:

Primjena robota hodača (eng. *quadruped robot*) je vrsta mobilnih robota čija primjena postaje sve značajnija zbog mogućnosti kretanja po terenima gdje klasični mobilni roboti, koji za gibanje koriste kotače, ne mogu pristupiti. Kako je razvoj robota hodača iznimno kompleksan te im je sukladno tome i cijena visoka, javlja se potreba da se rad s robotima hodačima simulira u virtualnim okruženjima.

U sklopu rada je potrebno:

- Napraviti pregled komercijalno dostupnih robota hodača te ih usporediti.
- Napraviti pregled algoritama hodanja otvorenog koda (*engl. open source*) te ih usporediti.
- Odabrati jedan tip robota hodača te ga opisati URDF datotekom (*Universal Robot Description File*).
- Odabrati jedan od dostupnih algoritama hodanja te ga implementirati na dobivenom modelu robota unutar robotskog operativnog sustava (ROS) i Gazebo simulatora.
- Unutar virtualnog okruženja je potrebno implementirati algoritam za praćenje predefinirane putanje robota hodača.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
30. 11. 2022.	**1. rok:** 20. 2. 2023. **2. rok (izvanredni):** 10. 7. 2023. **3. rok:** 18. 9. 2023.	**1. rok:** 27. 2. – 3. 3. 2023. **2. rok (izvanredni):** 14. 7. 2023. **3. rok:** 25. 9. – 29. 9. 2023.

Zadatak zadao: Predsjednik Povjerenstva:

Doc. dr. sc. Marko Švaco Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
$\alpha, \beta, \phi, \varphi$	°	kut (geometrija robota)
l_1, l_2, l_3	m	duljina (geometrija nogu robota)
x,y,z,x',y	m	koordinate u koordinatnim sustavima
x,y,z,x',y'	m	koordinate u koordinatnim sustavima
q, q_2, q_3	rad	unutarnje (upravljane) koordinate
T_{sw}	s	period zamaha
T_{st}	s	period stajanja
L_{span}	m	polovica duljine koraka
v_d	m/s	željena brzina robota
S_i^j		fazni signal (j-faza, i-noga)
t_{ref}^{elapse}	s	vrijeme od detektiranja "TD" dpgađaja
t_i	s	sat svake noge robota
t_{ref}^{TD}	s	vrijeme detektiranja "TD" događaja
T_{Stride}	s	vrijeme jednog koraka
$\Delta S_{ref,i}$		vrijednosti signala (opis hoda)

SAŽETAK

Glavni cilj ovog rada jest napraviti simulaciju četveronožnog robota (Spot) u Gazebo simulatoru tako da se on može navigirati po mapi uz obilaženje prepreka. Za implementaciju hodanja postoje već gotovi algoritmi hodanja kao što je ovdje Champ koji omogućuje izradu novog robota te je moguće napraviti konfiguraciju za Champ i implementirati algoritme za mapiranje, lokalizaciju i navigaciju četveronožnih robota pomoću ROS-a. Kao dio rada kratko su opisani četveronožni roboti, ROS i Gazebo, a napravljeni je i pregled komercijalno dostupnih četveronožnih robota i njihova usporedba te pregled open source algoritma za hodanje kako bi se mogle opisati moguće kretnje četveronožni robota. U radu je Spot opisan urdf datotekom kojom se definiraju sve značajke robota za simulaciju u Gazebo-u i RViz-u. Prikazani su mapiranje prostora u simulaciji, lokalizacija i navigiranje robota uz čitanje podataka sa senzora. Također je opisana inverzna kinematika robota strukture kao Spot te način generiranja hoda četveronožnih robota načinom troting(hodanje s dijagonalnim nogama). Za navigaciju robota DWA algoritam daje zadovoljavajuće rezultate.

Ključne riječi: četveronožni robot hodač, Gazebo, ROS, Champ, algoritmi hodanja, simulacija praćenja putanje, gmapping

SUMMARY

The main goal of this paper is to simulate a quadruped robot (Spot) in the Gazebo simulator so that it can move around the map while avoiding obstacles. To implement walking, there are ready-made walking algorithms such as Champ here, which allows the creation of a new robot, and it is possible to make a configuration for Champ and implement algorithms for mapping, localization and navigation of a quadruped robot using ROS. As part of the paper, quadruped robots, ROS and Gazebo are briefly described, and an overview of commercially available quadruped robots and their comparison is made, as well as an overview of open-source walking algorithms to describe the possible movement of quadruped robots. In the paper, Spot is described with a urdf file that defines all the features of the robot for simulation in Gazebo and RViz. Mapping of space in the simulation, localization and navigation of the robot along with reading data from the sensors are shown. Also described is the inverse kinematics of the robot structure as Spot, and the method of generating the gait of a quadruped robot using the trotting method(wlking with diagonal legs). For robot navigation, the DWA algorithm gives satisfactory results.

Key words: quadruped robot, Gazebo, ROS, Champ, walking algorithms, path tracking simulation, gmapping

1. UVOD

1.1. Mobilni roboti – hodači

Mobilni roboti su oni koji mogu raditi u svome radnom prostoru, ali ga mogu mijenjati. Odnosno oni imaju sposobnost kretanja u definiranom prostoru kako bi odradili neki zadatak unutar istog. Te kretnje su obavljene samostalno od strane robota bez pomoći ljudi. Svojom sposobnošću mobilnosti oni se mogu koristiti u puno većem opsegu radova u odnosu na fiksne industrijske robote. Dijele se na prizemne, podvodne i leteće. Roboti hodači spadaju u prizemne robote uz robote sa kotačima. Takvi roboti su prikladni za nestandardna okruženja, okruženja bez ravnih podloga i sl. Zapravo oni se koriste u okruženjima gdje se standardni roboti na kotačima nemaju mogućnost kretati, a uz to mogu se koristiti i u okruženjima opasnim po čovjekov život. Izvedbe mogu biti sa raznim brojem nogu, ali najčešće se koriste oni s četiri nogu. Na robotima hodačima s četiri nogu se bazira ovaj rad. Na [Slika 1] prikazana je takva izvedba robota hodača.

Slika 1. Prikaz robota hodača s četiri noge (Spot)[20]

1.2. Robot Operating System

ROS(Robot operating system) je set softverskih biblioteka za opisivanje robotskog sustava. Ako ga se malo opširnije opiše može se smatrati da je ROS set alata, standarda i biblioteka za pomoć pri izradi i stvaranju složenog robotskog sustava koji ima robusno ponašanje na raznim robotskim platformama. Najveća prednost ROS-a je mogućnost stvaranja novog robota.

Dalje će se pregledati ROS terminologija korištena za rješavanje zadatka:

1. Urdf (universal robot desciption file)

Urdf je tekstualni tip datoteke u kojem se definira niz mehaničkih značajki robota. Tako se u njemu nalaze podaci o vizualizaciji robota u okviru CAD-modela robota, pozicije dijelova robota koji su fiksni(link) te onih koji se pomiču odnosno zglobovi na robotu(joint). Za svaki od njih postoje i transformacije kako da se može definirati sami zglob odnosno njegovo kretanje naspram robota. Kod rotacijskih zglobova potrebno je znati oko koje se osi okreću pa je bitno definirati osi(axes). Također u njega se dodaju fizikalni podaci kao što su masa i inercija robota te također kolizijski okviri robota. Ta datoteka se koristi za ukupni opis robota kako bi ga mogli koristiti u simulaciji.

2. Publisher

Izdavač šalje podatke pretplatniku koji ga je iste zatražio.

3. Subscriber

Pretplatnik prima podatke od izdavača koje je zatražio.

Veza izdavača i pretplatnika može se shvatiti kao master-slave komunikacija.

4. Roslaunch

Alat kojim pozivamo određenu launch datoteku koja služi za pokretanje naredbi.

5. Gmapping

Sustav za lokalizaciju i mapiranje nepoznatog prostora.

6. RViz

ROS vizualizacija odnosno alat za 3D vizualizaciju u ROS-u. U njemu se može vidjeti simulirani model robota, a moguće je i prikazati podatke sa senzora na robotu.

7. Amcl

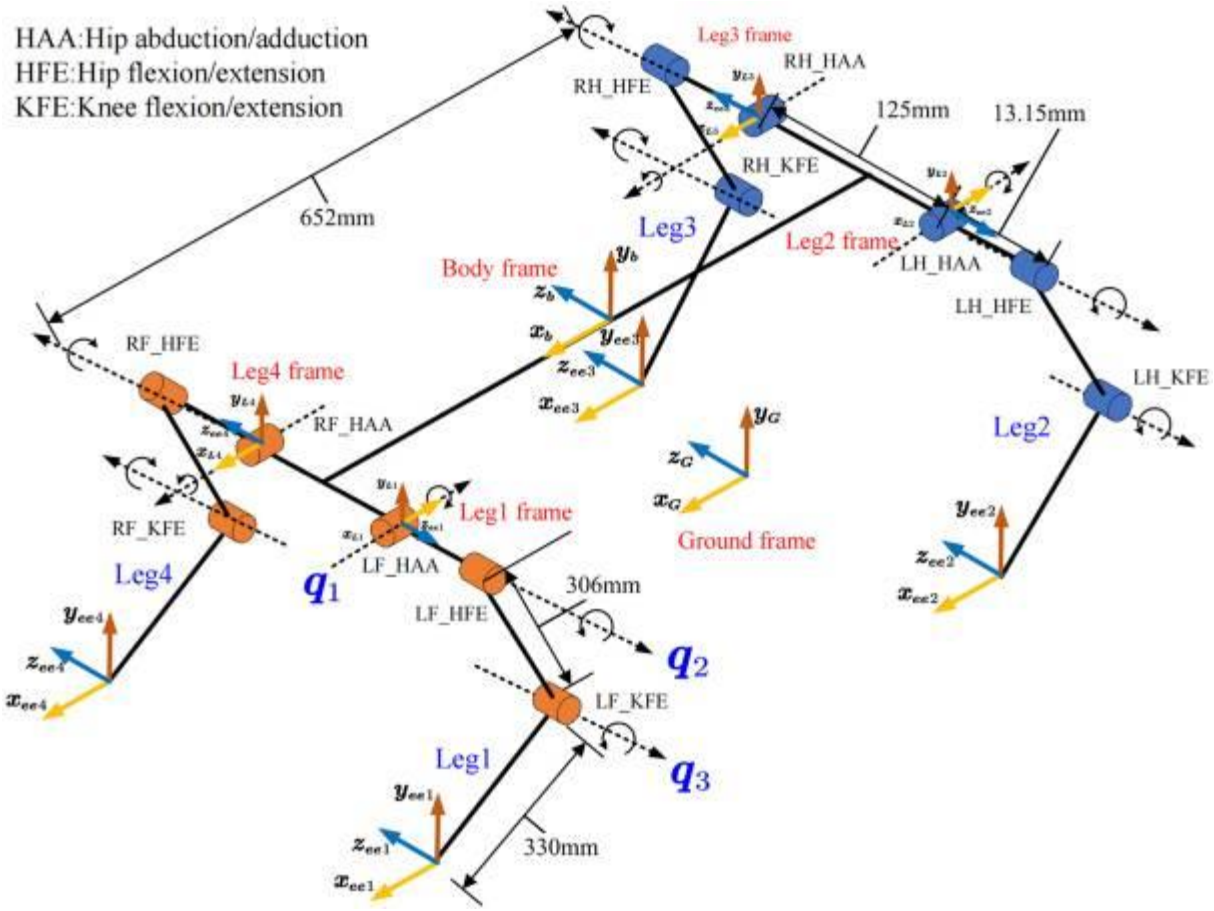
Sustav za lokalizaciju u unaprijed definiranom prostoru.

1.3. Gazebo

Gazebo je simulator robota koji se koristi za izračunavanje fizike robota, generira podatke za senzore na istom i pruža prikladna sučelja. Softver je otvorenog koda pa se može reći da potpomaže napredak robotike. U njemu je sadržano nekoliko fizičkih jezgri za modeliranje realne dinamike kao što su Simbody, Dynamic Animation and Robotic Toolkit, Bullet i ODE(Open Dynamics Engine). U Gazebo-u se simulira unutarnje i vanjsko okruženje robota.

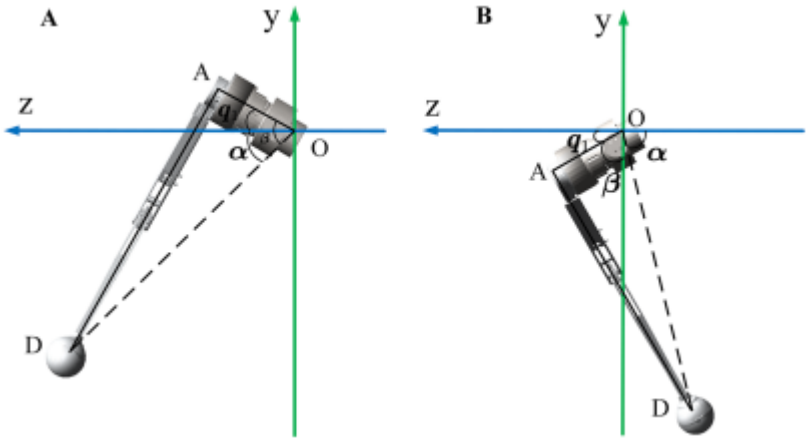
1.4. Inverzna kinematika četveronožni robota

Četveronožni roboti obično imaju 12 stupnjeva slobode gibanja odnosno 12 upravljanih koordinata tako da su na svakoj nogi 3 upravljane koordinate. Struktura jednog takvog robota prikazana je na [Slika 2].



Slika 2. Struktura četveronožnih robota s 12 stupnjeva slobode[2]

Kako bi se mogla izračunti inverzna kinematika za jednu nogu potrebno je prikazati koordinatne sustave te odnose između njih. Na [Slika 3] prikazane su geometrijske relacije jedne noge u yz ravnini.



Slika 3. Geometrijske relacije noge robota u yz ravnini[2]

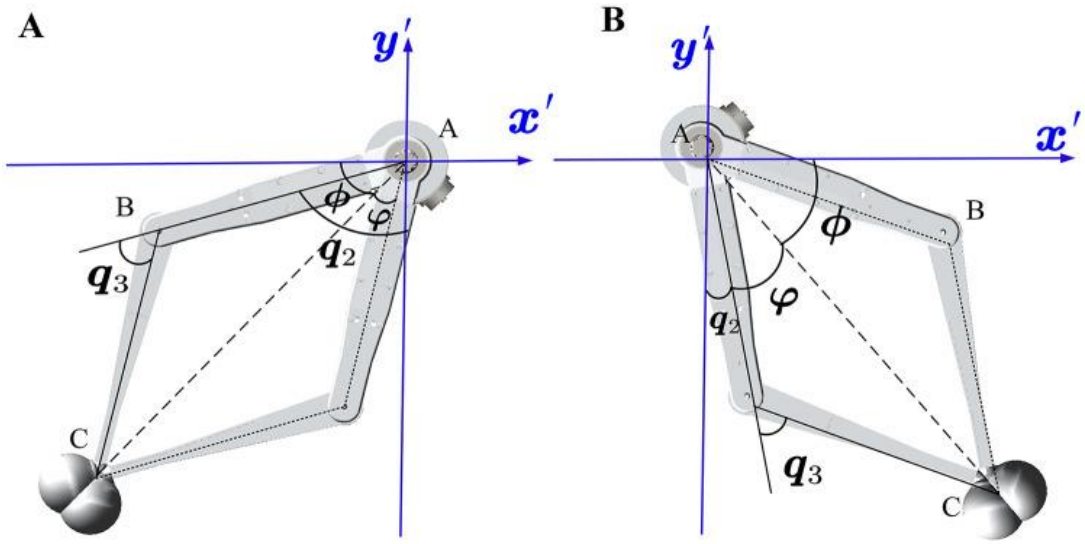
Iz [Slika 3] može se odrediti prva upravljana koordinata noge tj. zakret oko x osi koordinatnog sustava prvog zgloba. Prvi zglob može se zakretati oko x osi pripadajućeg koordinatnog sustava. Tada je prva upravljana koordinata dana jednadžbom:

$$q_1 = \begin{cases} \alpha - \beta & y < 0, z > 0 \\ \pi - \alpha - \beta & y < 0, z < 0 \end{cases} \quad \textbf{(1)}$$

Pri tome vrijedi jednadžba:

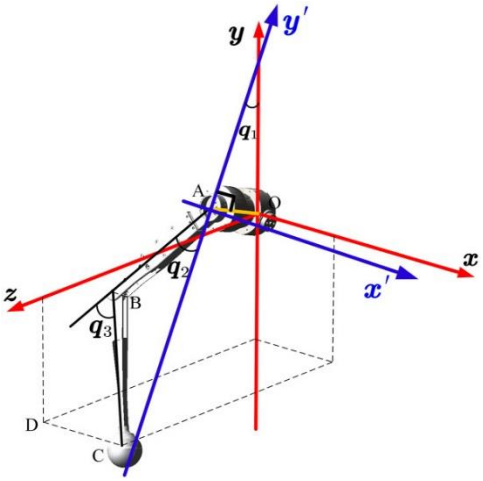
$$\alpha = arccos \frac{|z|}{\sqrt{y^2 + z^2}} \ , \qquad \beta = arccos \frac{l_1}{\sqrt{y^2 + z^2}} \quad \textbf{(2)}$$

Tijekom robotove kretnje gornji dio noge i donji dio noge uvijek su u istoj ravnini. Ta ravnina se koristi kako bi se izračunale druge dvije upravljane koordinate. Prikaz odnosa između gornjeg i donjeg dijela noge nalazi se na [Slika 4] gdje su y' i x' osi ravnine u kojoj se noga nalazi.



Slika 4. Geometrijske značajke gornjeg i donjeg dijal noge robota [2]

Također je potrebno definirati donos koordinatnog sutava ravnine x'y' u odnosu na inicijalni.



Slika 5. Prikaz odnosa koordinatnog sustava ravnine x'y' u odnosu na inicijalni [2]

Kako bi se moglo računati sa novim koordinatama potrebno ih je definirati te uz skicu sa [Slika 5] je taj odnos dan jednadžbom:

$$x' = x \ , \qquad y' = -\sqrt{y^2 + z^2 - l_1^2} \tag{3}$$

Pri tome mora se definirati i dva nova kuta koja su prikazana na [Slika 4]. Ti kutovi dani su jednadžbom:

$$\phi = arccos\frac{|x'|}{x'^2 + y'^2} \ , \qquad \varphi = arccos\frac{l_2^2 + x'^2 + y'^2 - l_3^2}{2l_2\sqrt{x'^2 + y'^2}} \tag{4}$$

Kada je sve definirano može se izračunati ostale dvije upravljane koordinate, a to su zakret oko z osi gornjeg dijela noge i donjeg dijela noge oko osi z gornjeg dijela noge. Pozicija nogu ima dva slučaja i za svaki slučaj dva rješenja pa je tako za y < 0 :

$$q_2 = \begin{cases} \frac{\pi}{2} - \varphi - \phi & x' > 0 \\ -\frac{\pi}{2} - \varphi + \phi & x' < 0 \end{cases}, \ q_3 = arccos\frac{l_2^2 + l_3^2 - x'^2 - y'^2}{2l_2l_3} \tag{5}$$

a za y' > 0:

$$q_2 = \begin{cases} \frac{\pi}{2} + \varphi - \phi & x' > 0 \\ -\frac{\pi}{2} + \varphi + \phi & x' < 0 \end{cases}, \ q_3 = -arccos\frac{l_2^2 + l_3^2 - x'^2 - y'^2}{2l_2l_3} \tag{6}$$

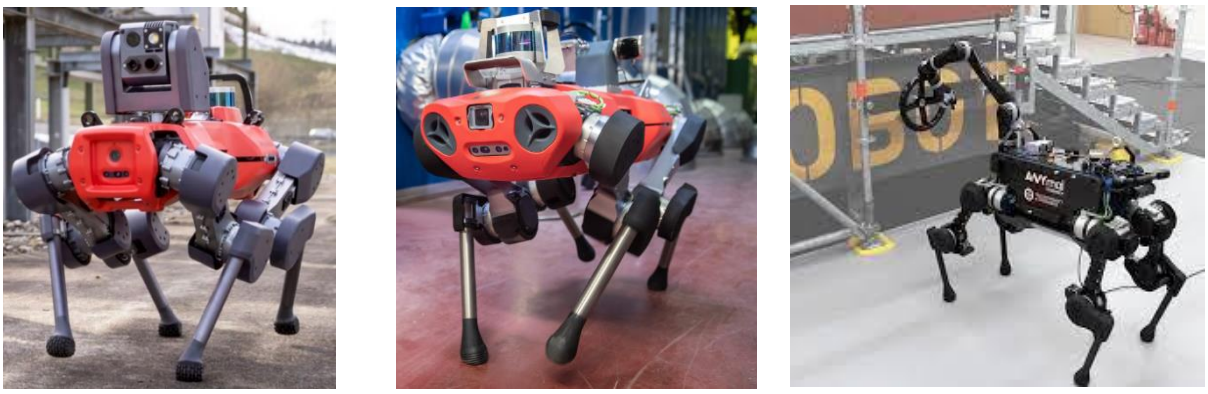
Ovim se jednadžbama može opisati svaka noga ovako strukturiranog četveronožni robota. Više podataka može se pronaći u [2].

2. PREGLED KOMERCIJALNIH ROBOTA HODAČA

Kao dio zadatka potrebno je napraviti pregled komercijalnih robota. Postoji sve više tvrtki koje se bave izradom robota hodača kako su roboti sve više zastupljeni u industriji i neprestano se unaprjeđuju. Dalje u ovom poglavlju su prikazani neki od komercijalnih robota hodača.

2.1. ANYbotics – ANYmal

ANYmal je robot hodač tvrtke Anybotics koja je bila osnovana 2016. godine iz ETH Zurich-a čiji je prvi cilj bio komercijalizacija tehnologije četveronožnih robota. Prvi robot kojeg su razvili bio je ALoF pa kasnije StarlETH još kao ETH Zurich koji su služili isključivo za razvoj tehnologije, a nakon osnivanja 2016 razvijen je prvi od ANYmal robota te 2018. prvi njihov komercijali robot ANYmal B. ANYmal roboti bez obzira na generaciju neke karakteristike imaju iste. Tako se na njima za percepciju koristi LIDAR koji mjeri do udaljenosti od 100m i kamere za udaljenosti do 3m. Namijenjeni su da rade vizualne inspekcije nad sustavima u industriji putem uređajima za inspekciju koji su unaprijed već napravljeni za taj robot. Baterije im rade 90-120 minuta. Svaki od robota ima mogućnost okretanja na mjestu i pomaka u stranu bez rotacije tj. svesmjeran (eng. Omnidirectional).



a) b) c)

Slika 6. ANYmal roboti a)ANYmal D b)ANYmal C c)ANYmal B [4]

2.1.1. Tablica karakteristikka ANYmal robota

Za usporedbu će se formirati tablica sa prikazanim nekim od glavnih karakteristika robota koje će se razmatrati u daljnjem pregledu robota.

Tablica 1. Karakteristike ANYmal robota

Robot	ANYmal D	ANYmal C	ANYmal B
Visina stepenice[mm]	250	350	200
Brzina hodanja[m/s]	1,3	1	1
Uzak prostor[mm]	600	600	-
Kut penjanja za rampu[°]	30	20	25
Najveći korak[mm]	300	250	-
Najniža visina[mm]	600	500	600
Dimenzije [mm]	930x530x890	1050x520x830	800x400x700
Trajanje baterije[min]	90-120	120-180	120-240
Masa[kg]	50/55,7	50	30
Nosivost[kg]	10/15	10	10
IP standard	IP67	IP67	IP67
Cijena[$]	150000	-	-

2.2. Boston Dynamics

Boston Dynamics je američka tvrtka za inženjering i dizajn robotike. Osnovana je 1992. godine. Glavna misija im je stvoriti robote koji se što više približuju mobilnosti, spretnosti i agilnosti ljudi. One robote koje razviju žele implementirati u svakodnevni život ljudi odnosno kako bi obavljali opasne operacije, fizički zahtjevne operacije, termalne inspekcije i sl.

2.2.1. Spot

Spot je četveronožni robot tvrtke Boston Dynamics koji za percepciju koristi stereo kamere, IMU(inertial measurement unit) te pozicijske senzore i senzore sile u udovima. Također je svesmjeran. Može se uspinjati i spuštati po stepenicama. Upravljanje je sa operatorom ali je moguće i da odrađuje neke operacije autonomno. Tvrtka ima mogućnost postavljanja robotske ruke na tijelo robota kao konfiguraciju. Na [Slika 7] prikazan je Spot.

Slika 7. Spot robot[5]

2.2.2. AlphaDog

AlphaDog je robot napravljen od tvrtke Boston Dynamics te mu je glavna funkcija prijenos teških tereta na raznim terenima, a namijenjen je za vojsku. Za percepciju koristi stereo kamere a miče se autonomno sa praćenjem čovjeka koji ima kontroler, a preko kontrolera može primati i jednostavne komande.Ono što je posebno za ovaj robot da mu je pogon na motor s unutarnjim izgaranjem. Opremljen je žiroskopima,stereo kamerama i senzorima sile i pozicije na udovima. Taj robot je prikazan na [Slika 8].

Slika 8. AlphaDog robot[21]

Na [Tablica 1] prikazane su karakteristike Spot i AlphaDog robota.

Tablica 2. Karakteristike Spot-a i AlphaDog-a

Robot	Spot	AlphaDog
Visina stepenice[mm]	300	-
Brzina hodanja[m/s]	1,6	-
Uzak prostor[mm]	-	-
Kut penjanja za rampu[°]	30	-
Najveći korak[mm]	-	-
Najniža visina[mm]	520	-
Dimenzije [mm]	1100x500x700	2000x1900x900
Trajanje baterije[min]	90	-
Masa[kg]	32,7	362,9
Nosivost[kg]	14	180
IP standard	IP54	-
Cijena[$]	75000	-

2.3. Unitree Robotics

Unitree je kineska tvrtka koja se može smatrati konkurentom Boston Dynamics-a. Bave se razvojem, produkcijom i prodajom industrijskih i konzumnih četveronožnih robota i spretnih robotskih ruku. Svaki dio robota koji proizvode razvijen je samostalno. U Kini drže najviše patenata za četveronožne robote. Kao industrijske robote proizveli su B1, AlienGo robote, a još prodaju i Go1 robot.

2.3.1. Unitree AlienGo

AlienGo za percepciju koristi 2 seta kamera za udaljenosti od 0,11m i odometrijsku kameru sa obuhvatnim kuto od 136°. Kao i većina quadraped robota može se penjati stepenicama, rampama, ima mogućnost skakanja, prebacivanja oko sebe i dizanje sa poda. Na sljedećoj strani na [Slika 9]prikazan je taj robot.

Slika 9. Unitree AlienGo[6]

2.3.2. Unitree B1

Unitree B1 može raditi u svim vremenskim uvjetima te kompleksnim okruženjima pa je također pogodan za industrijsku upotrebu. Za AlienGo i B1 robote napravljena je robotska ruka Unitree Z1 kojem robot može obavljati određene operacije. [Slika 10] prikazuje Unitree B1 robot.

Slika 10. Unitree B1 robot sa robotskom rukom Unitree Z1[6]

2.3.3. Unitree Go1

Unitree Go1 napravljen je više za civilnu upotrebu te je jeftiniji i manje opremljen od industrijskih robota Unitree tvrtke. Robot je napravljen da hoda uz čovjeka(mastera) te radi autonomno, ali se robotu može pomoći u pronalaženju bolje rute u kompleksnim okruženjima. [Slika 11] prikazuje Go1 robot.

Slika 11. Unitree Go1 robot[6]

2.3.4. Tablica karakteristika Unitree robota

U [Tablica 3] prikazne su karakteristike Unitree robota.

Tablica 3. Karakteristike Unitree robota

Robot	Unitree AlienGo	Unitree B1	Unitree Go1 Edu
Visina stepenice[mm]	180	200	-
Brzina hodanja[m/s]	1,5	1,2	1
Uzak prostor[mm]	-	-	-
Kut penjanja za rampu[°]	25	30	-
Najveći korak[mm]	-	-	-
Najniža visina[mm]	150	500	130
Dimenzije [mm]	650x310x600	1202x467x297	645x280x400
Trajanje baterije[min]	150-276	150	60-150
Masa[kg]	21,5	50	12
Nosivost[kg]	13	20	5
IP standard	-	IP68	-
Cijena[$]	100000	200000	20000

2.4. Weilan

Weilan je kineska tvrtka osnovana 2019. godine. Tada su predstavili svoj prvi četveronožni personal robot projekt. Namijenjen je za osobnu upotrebu. Trenutno su to najbrži četveronožni roboti na tržištu.

2.4.1. Weilan AlphaDog

Weilan-ov robot AlphaDog ima vizijske senzore i senzore zvuka. Napravljen je kako bi mogao asistirati ljudima u svakodnevnom životu. Robot je jako stabilan te ima dinamičko balansiranje. [Slika 12] prikazuje robota. U [Tablica 4] prikazane su karakteristike Weilan AlphaDog robota.

Slika 12. Weilan AlphaDog

Tablica 4. Karakteristike Weilan AlphaDog-a

Robot	Weilan AlphaDog
Visina stepenice[mm]	-
Brzina hodanja[m/s]	3
Uzak prostor[mm]	-
Kut penjanja za rampu[°]	20
Najveći korak[mm]	-
Najniža visina[mm]	520
Dimenzije [mm]	760x400x450
Trajanje baterije[min]	342
Masa[kg]	24
Nosivost[kg]	3
IP standard	-
Cijena[$]	2450

2.5. MIT Mini Cheetah, Ghost Robotics Vision 60, Hiwonder PuppyPi

Postoje još mnoge tvrtke koje se bave izradom četveronožnih robota jer postoji još mjesta za napredak tehnologije koja donosi mnoge prednosti, a kao što je vidljivo tehnologija je dovoljno napredovala za korištenje takvih robota u gotovo svim okruženjima.

Ovdje su u pregled dodani još Mini Cheetah robot koji je donedavno bio najbrži takvi tip robota, Vision 60 koji se razvija za primjenu u vojsci i Puppy Pi kao primjer robota pogodnog za učenje i upoznavanje s radom ovakvih robota. Na [Slika 13] prikazani su ti roboti.

a) b) c)

Slika 13. a) MIT Mini Cheetah[23] b) Ghost Robotics Vision 60[9] c) Hiwonder PuppyPi[10]

Tablica 5. Karakteristike Mini Cheetah, Vision 60 i PuppyPi robota

Robot	Mini Cheetah	Vision 60	PuppyPi
Visina stepenice[mm]	-	-	-
Brzina hodanja[m/s]	2,45	3	2,45
Uzak prostor[mm]	-	-	-
Kut penjanja[°]	-	-	-
Najveći korak[mm]	-	-	-
Najniža visina[mm]	-	250	-
Dimenzije [mm]	480x270x300	830x460x640	226x149x190
Trajanje baterije[min]	30-120	3,5	60
Masa[kg]	9	20-27	0,72
Nosivost[kg]	-	10	-
IP standard	-	IP55-IP67	0
Cijena[$]	10000	150000	560

2.6. Usporedba komercijalno dostupnih robota

Ako se roboti spomenuti u prijašnjim poglavljima usporede može se vidjeti da postoji 3 grupe primjene ovih robota i to su industrijski, hobistički i oni za osobnu upotrebu. Kod industrijskih robota kao što su ANYmal, Spot, AlienGo i B1 može se vidjeti da je dano puno više specifikacija kako bi se što bolje moglo procijeniti je li robot odgovara zamišljenoj aplikaciji. Isto tako primjećuje se velika razlika u cijeni naspram onih za osobnu upotrebu i hobističkih, ali razlog za to jest kvaliteta konstrukcije i pouzdanost komponenta, izrada robota sa IP zaštitom te razvoj istog. Također u cijenu se često ubraja autonomna platforma, LIDAR senzor i stanica za punjenje. Kada se malo usporedi ta vrstu četveronožnih robota vidi se da su više-manje ti roboti istih karakteristika uz male razlike u brzinama, karakteristikama kretanja i nosivosti. Zbog toga se ne može reći koji je od njih najbolji kako to ovisi isključivo o aplikaciji. Ako se razmotri brzina tada je najbolji Spot robot. Pak ako je u pitanju nosivost tada su to ANYmal D i Unitree B1. Što se tiče mogućnosti kretanja ne vidi se koji je najbolji, ali ako se okrenemo cijeni robota tada bi to opet bio spot.

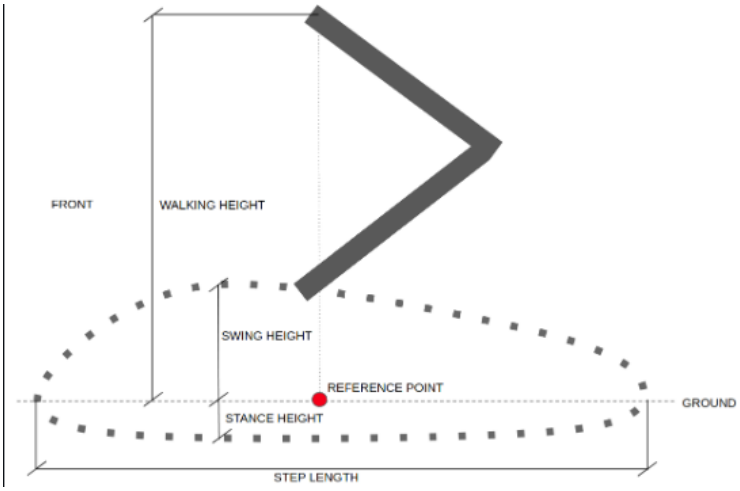
Kada se pogledaju roboti za osobnu primjenu kao Unitree Go1 i Weilan AlphaDog može se vidjeti da se četveronožni roboti približavaju čovjeku na način da se uklope u svakodnevni život. Radi se na tome da takvi roboti počinju služiti za vođenje slijepih osoba, mogu služiti za prijenos stvari i sl. I dalje oni su još skupi za obične ljude, ali sa povećanjem područja primjene i eventualnim pojednostavljenjem tehnologije može se vidjeti prostor za napredak i uklapanje četveronožnih robota u svakodnevni život.

Kao zadnja grupa navedeni su roboti za hobističku upotrebu koji mogu poslužiti za učenje o robotskim sustavima, upravljanjima istih, učenje o viziji robota te programiranju. Postoji puno izvedba ovakvih robota, a cijena im je dovoljno pristupačna za svakog koga zanima to područje.

3. Pregled open source algoritama hodanja

3.1. Champ

Champ je algoritam otvorenog hoda koji upravlja hodanjem četveronožnih robota. Cijeli champ paket sadrži algoritam hodanja, preddefinirane robote i champ setup asistenta koji služi za konfiguraciju novo razvijenih robota. [Slika 14] prikazuje način postavljanja parametara za hodanje.



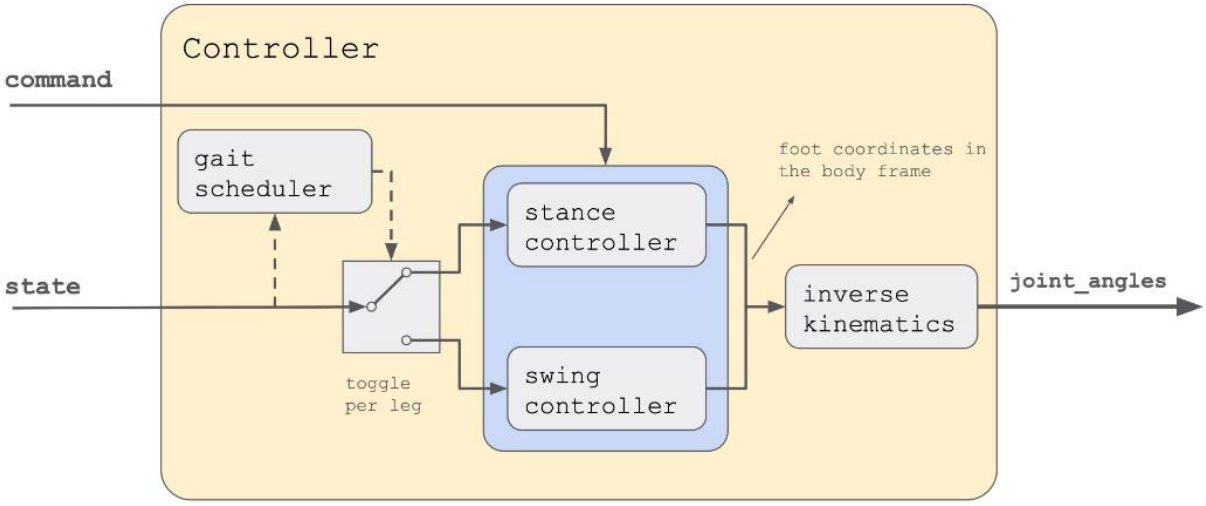
Slika 14. Parametri za Champ[11]

Parametri koje je moguće mijenjati su sljedeći:

- Orijentacija koljena: kako su postavljena koljena robota (.>>, .><, .<<, .< > gdje je točka prednja strana robota).
- Maksimalna linearna brzina po x osi: definira maksimalnu brzinu robota unaprijed i iza.
- Maksimalna linearna brzina po y osi: definira brzinu robota kad se giba ustranu.
- Trajanje stava: definira koliko sekundi je svaka noga robota na podu kada robot hoda.
- Visina noge pri hodanju: visina trajektorije kada je noga u pokretu u metrima.
- Visina noge pri stajanju: definira dubinu trajektorije noge kada stoji na podu u metrima.
- Visina robota kad hoda: definira visinu od poda do gornjeg zgloba noge u metrima.
- CoM X translacija: referentna točka ako centar mase robota nije u sredini robota.
- Skaliranje odometrije: služi kompenzaciji pogreška odometrije na otvorenim sustavima.

3.2. Stanford Quadruped

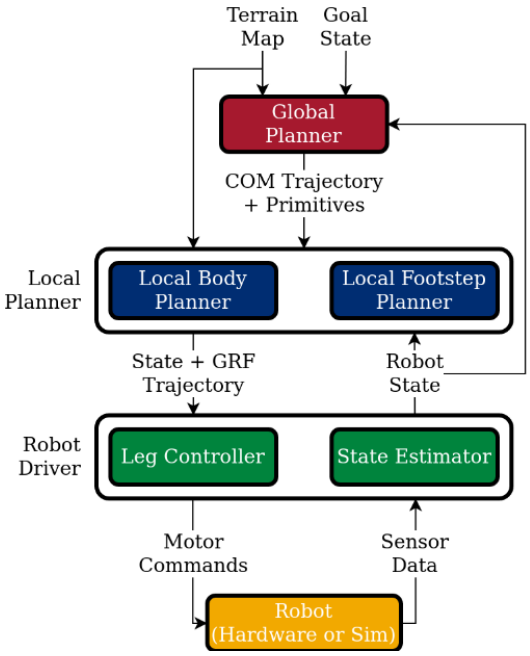
U ovom kontroleru može se vidjeti četiri glavne komponente: kontroler hoda, kontroler stava, kontroler zamaha i model inverzne kinematike. Kontroler hoda odgovoran je za planiranje koja stopala trebaju biti na podu a koje se miču. Kontroler stava zaslužan je za stopala na podu i radi na način da zakreće motore u suprotnom smjeru od željenog smjera kretanja robota prema naprijed ili iza, a isto tako i za okretanje robota. Kontroler zamaha podiže stopala koja su završila fazu stava i dovodi ih do sljedećeg mjesta dodira, a ta mjesta su definirana na način da se motor jednako zakrene u fazi zamaha kao što se zakrenuo u fazi stava. Oba ta kontrolera generiraju ciljne položaje u Kartezijevom koordinatnom sustavu te se korištenjem inverzne kinematike pretvaraju u zakrete motora. Zakreti motora popunjavaju varijablu stanja i vraćaju se u model. Na [Slika 15] prikazan je model kontrole robota.



Slika 15. Stanford quadruped kontroler[12]

3.3. Quad-SDK

Quad-SDK je softver otvorenog koda temeljen na ROS-u za agilno četveronožno kretanje robota. Dizajn je usmjeren na vertikalnu integraciju alata za planiranje, kontrolu, procjenu pozicija, komunikaciju i razvoj koji omogućuju agilno četveronožno kretanje u simulaciji i na hardveru. Zamišljeno je da ima minimalo korisničkih promjena. Omogućuje istraživačima da eksperimentiraju implementacijom različitih komponenti dok koriste postojeći softver. Nudi i podršku za gazebo simulaciju, paket alata za vizualizaciju i obradu podataka. Na [Slika 16] prikazan je grafički sažetak kontrolera.



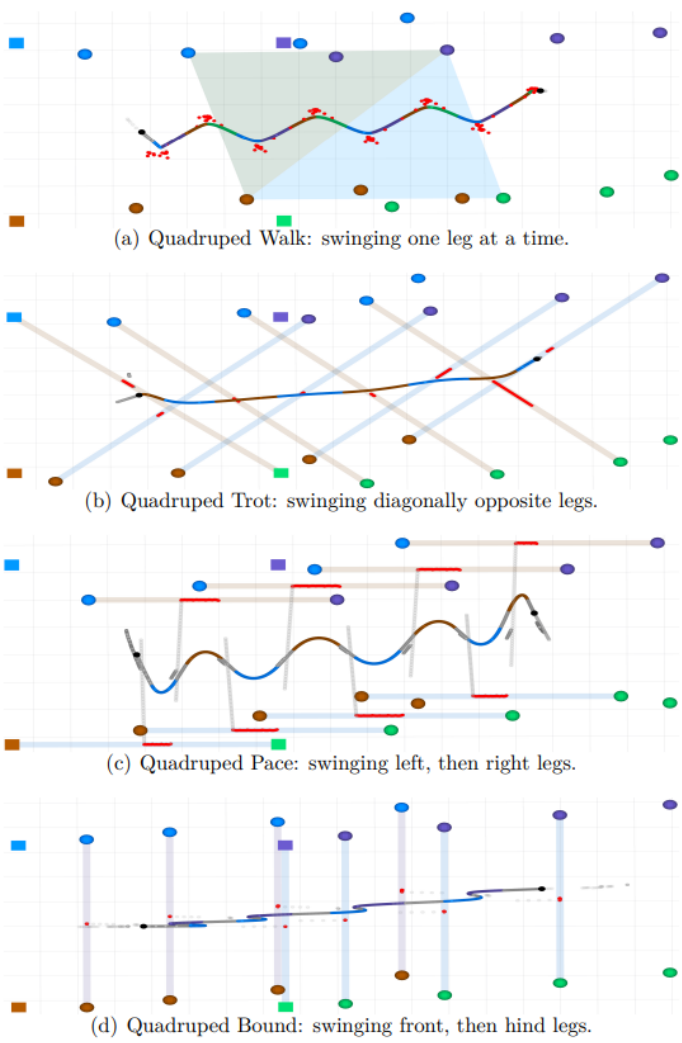
Slika 16. Grafički sažetak Quad-SDK kontrolera[13]

Launch datoteke:

- Quad_gazebo.launch – pokreće simulaciju i inicijalizira kontrolu.
- Quad_plan.launch – pokreće planiranje.
- Quad_visualization.launch – pokreće vizualizaciju robota i tražena grafička sučelja.
- Remote_driver.launch – poziva vizualizacijske skripte i skripte za mapiranje.
- Planning.launch – pokreće planiranje i kontrolu.
- Mapping.launch – pokreće izdavača za podatke o terenu(okruženju) robota
- Visualization_plugins.launch – pokreće dodatke vizualizacije kako bi podatki mapiranja bili pogodni za čitanje RViz-u.

3.4. Towr

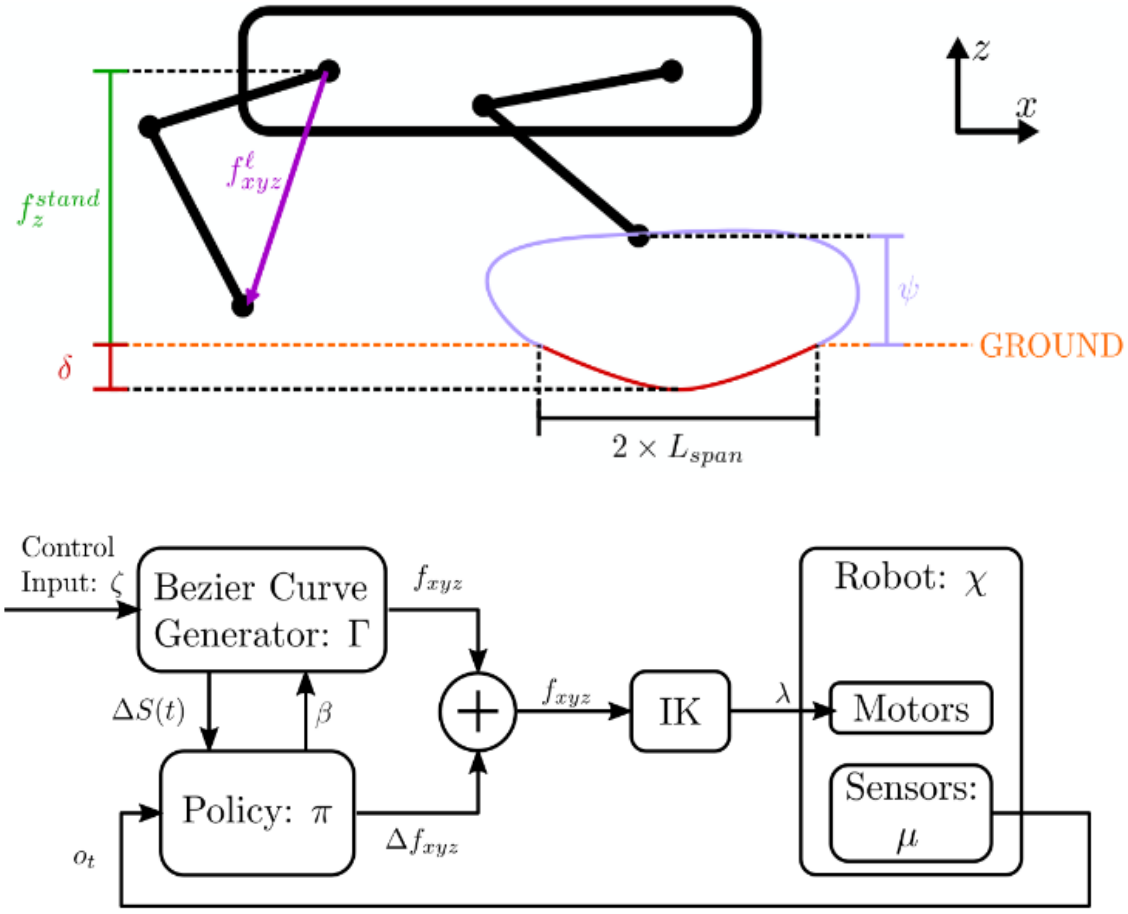
Towr generira fizički izvedive pokrete za robote hodače rješavanjem problema optimizacije. Fizički se daju ograničenja i željeni cilj, a kontroler generira plan kretanja sa raznim načinima hoda kao dvonožno hodanje, kaskanje četveronožnih robota i slično. Kontroler izrazito brzo računa plan kretanja. Fizička ograničenja se temelje na poziciji noge prilikom hodanja, definiranja terena te stošcima trenja. Algoritam je osmišljen tako da se centar mase robota uvijek nalazi unutar potpornih okvira kao što su trokuti i linije među nogu na podu, ali se time gubi na nepomičnoj bazi robota ukoliko je neki senzor na njoj. [Slika 17] prikazuje načine kretanja sa Towr kontrolerom.



Slika 17. Načini hodanja robota koristeći Towr kontroler[24]

3.5. Spot Micro kontroler

Kontoler koji koristi tzv. Bezier Gait generator. Osnovna prilagodba generatora Bezierove krivulje daje 2D koordinate stopala tijekom vremena(vodoravne i okomite). Za bočno kretanje koristi se zakretanje 2D putanje oko z osi robota. Alogritam je takvog tipa da robot mora sam naučiti hodati. Cilj je da se sa par ulaza kontroleru mogu dobiti raznolika kretanje robota. Za skretanje koristi se intuicijom da za skretanje u smjeru kazaljke na satu sva četiri stopala moraju pratiti krug u smjeru suprotnom od kazaljke na satu gdje bi se obje prednje noge trebale kretati prema stražnjem lijevom, a obje stražnje noge trebale bi se kretati prema stražnjem desnom dijelu robota. Na [Slika 18] prikazana je shema kontrolera i generator Bezierovih krivulja.



Slika 18. Prikaz hoda sa Bezierovim krivuljama i shema kontrolera[16]

3.6. Usporedba kontrolera

Algoritmima iz prijašnjih poglavlja može se zadovoljiti sve kretnje četveronožnih robota. Kretnje koje nam prijašnji algoritmi mogu ostvariti su hodanje, ravnoteža(stajanje), tzv. trotting , skakanje, statično hodanje, galopiranje i slično.

Kao najbolji trenutno razvijeni algoritam hodanja jest Quad-SDK te je također osvojio nagradu za najbolji članak na ICRA sajmu 2022. Kod njega se vidi da je vrlo pogodan za upotrebu, a robot se dobro može adaptirati novom stanju i terenu. Ne koristi klasično hodanje sa fiksnom bazom nego ju koristi pri kretanju što ima za korist veću iskoristivost sustava i agilnijim kretanjem.

Champ je također dobro razvijen algoritam hodanja međutim kod njega robot nije u mogućnosti hodanja stepenicama i neravnim terenima. Bez obzira na taj nedostatak prednosti su mu da je jednostavan za interpretaciju, a može se simulirati i mapiranje prostora te je već napravljen da pomaže kod izrade novog dizajna robota.

Sve se više radi na algoritmima koji mogu naučiti robota da hoda samostalno i pokazuju dobre rezultate. Takav je Spot micro kontroler prikazan prije. U eksperimentu robot se naučio sam hodati u 149 epoha.

Za četveronožnih hobističke robote dovoljno dobri su i jednostavni kontroleri kao što je to Stanford quadruped kontroler. Sa svakim od ovih kontrolera može se robotom upravljati i preko joystika tako da operater može imati kontrolu nad radom robota.

4. URDF datoteka i opis robota (Spot)

Za simulaciju u radu odabran je Spot robot kojim će se upravljati sa Champ kontrolerom. Prvo će se Spot robot opisati URDF datotekom kako bi ga se moglo simulirati u Gazebo simulatoru sa što boljom interpretacijom stvarnih karakteristika robota. URDF datotaka sadržava mnogo karakteristika kao što su inercija, masa, geometrija te zglobovi robota. Služi za cjelokupni opis robota.

4.1. Kratki opis kostura URDF datoteke

Kao pobliži opis dan je dio XML koda URDF-a za Spot robot sa objašnjenjem funkcija unutar datoteke te njihovo fizikalno značenje. Dalje je dani dio koda URDF datoteke.

```
<?xml version="1.0" encoding="utf-8"?>

<robot name="spot">
  <link name="body">
    <visual>
      <geometry>
        <mesh filename="package://spot_description/meshes/body.dae"/>
      </geometry>
    </visual>
    <collision>
      <geometry>
        <mesh
filename="package://spot_description/meshes/body_collision.stl"/>
      </geometry>
    </collision>
    <inertial>
      <mass value="16.0"/>
      <inertia ixx="0.0973333333333" ixy="0.0" ixz="0.0"
iyy="1.02466666667" iyz="0.0" izz="1.04493333333"/>
    </inertial>
  </link>
  <link name="base_link"/>
  <joint name="base_link_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <parent link="body"/>
    <child link="base_link"/>
  </joint>
```

Unutar stabla urdf datoteke nalaze se dva seta podataka pod nazivom link i joint. Prvo se radi takozvani bazni link koji nam služi za stvaranje kinematičkog lanca robota. U ovom kodu taj link je definiran kao base_link. On nema nikakve komponente stvara se u xyz kordinatama iznosa 0 bez ikakvih rotacija. Dalje se radi link pod nazivom body koji ima vizualnu, kolizijsku i inercijsku komponentu. U vizualnoj komponenti definira se geometrija koja se želi vizualizirati odnosno u ovom slučaju trup robota. Za definiranje geometrije se može cijeli robot modelirati u CAD alatima te se spremiti kao .stl ili .dae datoteka kako bi se mogla interpretirati. U kolizijskoj komponenti se nalazi geometrija koja služi da se definira što će

robot smatrati sudarom. Ona se obično pojednostavljeno u slučaju da robot ima složenu vizualizaciju kako je ovdje. Dalje se iz CAD modela mogu uzeti podaci kao masa i inercija pojedinih dijelova što se opisuje inercijskom komponentom u urdf.u. Nakon postavljanja svih parametara prve komponente može se krenuti s vezama između dijelova robota.

Te veze definirane su joint-om, a postoje sljedeće vrste:

- Fiksni – nema nijedan stupanj slobode gibanja.
- Revolutni – okreće se oko jedne osi te ima definirane granice.
- Kontinuirani – okreće se oko jedne osi te pritom nema granica.
- Prizmatični – klizni spoj duž jedne osi sa definiranim granicama.
- Planarni – kretanje u ravnini okomitoj na neku os.
- Plutajući – svi stupnjevi slobode su omogućeni.

```
<joint name="front_left_hip_x" type="revolute">
    <origin rpy="0 0 0" xyz="0.29785 0.055 0"/>
    <axis xyz="1 0 0"/>
    <parent link="body"/>
    <child link="front_left_hip"/>
    <limit effort="100" lower="-0.78539816339744827899"
upper="0.78539816339744827899" velocity="1000.00"/>
  </joint>
```

Kako se vidi sa slike zglob je revolutni, a okreće se oko osi x što definira linija 233. Kao granice vidljivo je da postoji gornja i donja te su one definirane u radijanima. Vidi se da je za taj zglob transformacija translacija po x osi za 0,29785m, a po y osi 0,055m te da postoji zakret oko x osi prema granicama. Ta transformacija je odnos trupa i prednjeg lijevog kuka robota.

4.2. URDF xacro datoteka za Spot

Dalje će biti prikazan xacro urdf koji služi da kod bude pregledniji te kako kod možemo bolje sistematizirati po djelovima. Glavna urdf datoteka je generirana iz xacro koda.

```
<?xml version="1.0" ?>
<robot name="spot" xmlns:xacro="http://www.ros.org/wiki/xacro">
    <xacro:include filename="$(find
spot_description)/urdf/constants.urdf.xacro" />

    <link name="body">
        <visual>
            <geometry>
                <mesh filename="package://spot_description/meshes/body.dae"
/>
            </geometry>
        </visual>
        <collision>
            <geometry>
                <mesh
filename="package://spot_description/meshes/body_collision.stl" />
            </geometry>
        </collision>
        <inertial>
            <mass value="${base_mass}"/>
            <inertia ixx="${(1/12) * base_mass * (base_width * base_width +
base_height * base_height)}" ixy="0.0" ixz="0.0"
                     iyy="${(1/12) * base_mass * (base_length *
base_length + base_height * base_height)}" iyz="0.0"
                     izz="${(1/12) * base_mass * (base_length *
base_length + base_width * base_width)}" />
        </inertial>
    </link>

    <link name="base_link"/>
    <joint name="base_link_joint" type="fixed">
        <origin xyz="0 0 0" rpy="0 0 0" />
        <parent link="body" />
        <child link="base_link" />
    </joint>

    <link name="front_rail"/>
    <joint name="front_rail_joint" type="fixed">
        <origin xyz="0.223 0 0.0805" rpy="0 0 0" />
        <parent link="body" />
        <child link="front_rail" />
    </joint>

    <link name="rear_rail"/>
    <joint name="rear_rail_joint" type="fixed">
        <origin xyz="-0.223 0 0.0805" rpy="0 0 0" />
        <parent link="body" />
        <child link="rear_rail" />
    </joint>

    <!-- Optional custom includes. -->
    <xacro:include filename="$(optenv SPOT_URDF_EXTRAS empty.urdf)" />
    <xacro:include filename="$(find
spot_description)/urdf/accessories.urdf.xacro" />
```

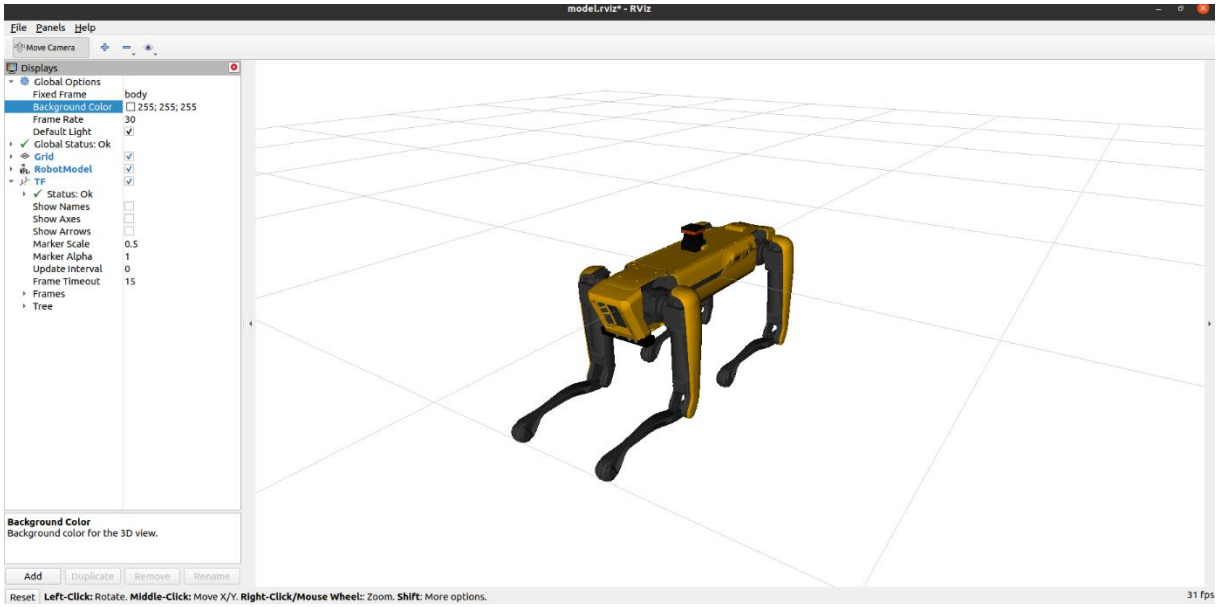
```
    <xacro:include filename="$(find
spot_description)/urdf/spot_leg.urdf.xacro" />
    <xacro:spot_leg leg_name="front_left"/>
    <xacro:spot_leg leg_name="front_right"/>
    <xacro:spot_leg leg_name="rear_left"/>
    <xacro:spot_leg leg_name="rear_right"/>

   <gazebo>
      <plugin name="p3d_base_controller" filename="libgazebo_ros_p3d.so">
            <alwaysOn>true</alwaysOn>
            <updateRate>10.0</updateRate>
            <bodyName>base_link</bodyName>
            <topicName>odom/ground_truth</topicName>
            <gaussianNoise>0.01</gaussianNoise>
            <frameName>world</frameName>
            <xyzOffsets>0 0 0</xyzOffsets>
            <rpyOffsets>0 0 0</rpyOffsets>
        </plugin>
    </gazebo>

    <gazebo>
        <plugin name="gazebo_ros_control"
filename="libgazebo_ros_control.so">
            <legacyModeNS>true</legacyModeNS>
        </plugin>
    </gazebo>
</robot>
```

4.3. Vizualizacija robota u RViz-u

Sada kada je napravljen urdf, robot se želi vizualizirati i testirati radi li sve kako je robot zamišljen ili izveden u ovom slučaju. To se može napraviti u Rviz okruženju. Rviz nudi prikaz cijelog robota, transformacije između koordinatnih osi, a može se i pregledati gibanje zglobova i njihove granice te još mnogo drugih parametara kako bi se robot što bolje predočio korisniku. U champ paketu postoji već gotovo rješenje za vizualizaciju robota, potrebno je samo imati opis robota odnosno urdf datoteku.

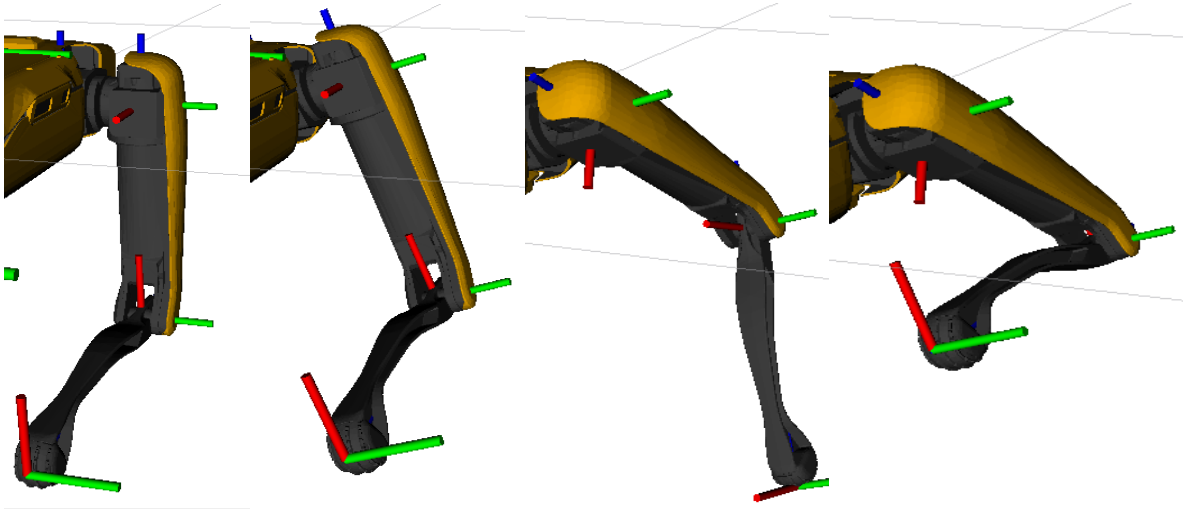
Vizualizacija se pokreće sljedećom naredbom: roslaunch spot_viz view_model.launch .

Na [Slika 19] prikazana je vizualizacija urdf datototeke.



Slika 19. Vizualizacija Spot-a u Rviz-u

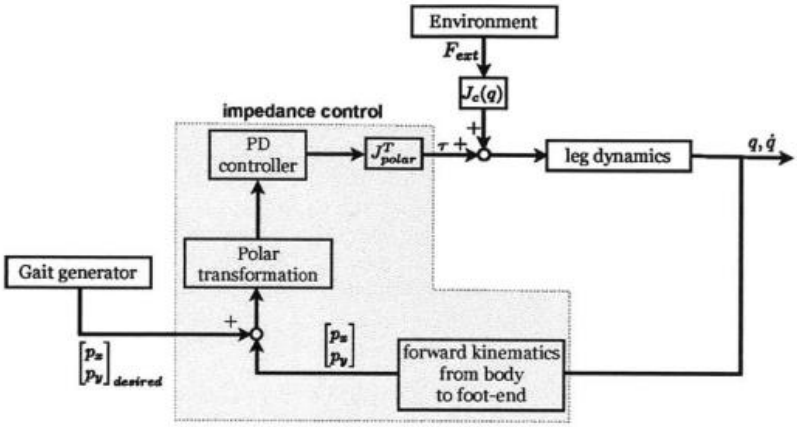
Kako se prije navelo u Rviz-u se može provjeriti gibaje robota i njegovih zglobova pa je tako na [Slika 20] prikazano zakretanje pojedninih zglobova jedne noge robota.



Slika 20. Zakretanje zglobova jedne noge Spot-a

5. ALGORITAM ZA HODANJE – CHAMP

Dalje u zadatku treba implementirati jedan od algoritama hodanja u ROS-u i Gazebo-u. Odabrani je Champ algoritam pa će se pregledati što sve sadrži istoimeni paket. U paketu se nalaze datoteke koje se mogu služiti za samo hodanje robota, upravljanje robota tipkovnicom ili džojstikom, mapiranje prostora te autonomna navigacija. Također paket sadrži opise nekih od komercijalno dostupnih robota kao što su Spot, ANYmal, Cheetah, Aliengo itd. Nadalje u paketu postoji i asistent za konfiguraciju koji služi kako bi se sa champ algoritmom mogao simulirati bilo koji robot, ukoliko za njega postoji urdf datoteka. To znači da se razvoj novog četveronožnog robota može uveliko olakšati te da se taj robot bez izrade može testirati. Korišteni dijelovi paketa u ovom radu su hodanje robota, mapiranje, navigacija te konfiguracija Spot robota za Champ algoritam te upravljanje tipkovnicom.



Slika 21. Shema champ kontrolera za jednu nogu[25]

Kako [Slika 21] prikazuje shemu kontrolera za jednu nogu koji se implementira potrebno ju je malo bolje objasniti. Za ulaz se koristi generator hoda prikazan na [Slika 14]. Kada položaj noge detektiraju enkoderi u zglobovima, direknom kinematikom i Jacobijevom matricom se dobije pozicija stopala i brzine za postizanje te pozicije. Tada se putem povratne veze i ulaza računa greška te se ona transformira u polarne koordinate koje se množe sa parametrima regulatora kako bi se izračunali momenti koje motori trebaju ostvariti za gibanje robota.

5.1. Generiranje hoda

Da bi se samo hodanje bolje shvatilo potrebno je taj dio kontrolera sagledati te odrediti što sve je definirano te kako se generira hod robota. Željenu brzinu i hod dobije se koordiniranjem četiri nogu robota s tzv. faznim signalima S. Fazni signal svake noge je potom podijeljen na razdoblje zamaha T_{sw} i razdoblje stajanja T_{st} koji se određuju prema željenim brzinama. Fazni signal S_i^j je zadan za svaku nogu tako da je $S_i^j \in [0, 1]$, a i $\in \{FR, FL, BR, BL\}$ koji su

indeks noge(FR - prednja desna)i j ∈ $\{st, sw\}$ što definira stanje noge (zamah, stajanje). Pritom su razdoblja zamaha i stajanja dana jednadžbom:

$$T_{sw} = 0{,}25s \ , \quad T_{st} = \frac{2L_{span}}{v_d} \tag{7}$$

gdje je v_d željena brzina, a L_{span} polovica duljine koraka.

Fazni signali koji su generirani moraju biti sinkronizirani s okolinom pa se treba ispravno detektirati događaji kada noga dotakne tlo(TD) i kada se ona podigne s tla(LO). Time se dolazi do potrebe definiranja kako će se oni detektirati. Događaji se detektiraju ako se na senzore sila na stopalima postavi limit koji definira neku silu dodira, te ako se ta vrjednost premaši pojavljuje se „TD“ događaj, a „TO“ događaj ako je ta vrijednost manja od zadanog limita. Prema tome su definirane sljedeće jednadžbe:

$$t_{ref}^{elapse} = \begin{cases} t - t_{ref}^{TD} \\ T_{Stride} \quad ako\ je\ t_{ref}^{elapse} > T_{Stride} \end{cases} \tag{8}$$

$$t_{ref}^{TD} = t \quad kada\ S_{ref}^{sw} > 0.9\ i\ detektiran\ "TD" \tag{9}$$

$$t_i = t_{ref}^{elapse} - \Delta S_{ref,i} T_{Stride} \tag{10}$$

gdje je t_{ref}^{elapse} prošlo vrijeme od detektiranja „TD“ događaja, t_i je sat svake noge gdje je $i \in \{FR, FL, BR, BL\}$, a $t_{\text{ref}}^{\text{TD}}$ se ažurira u svakom detektiranom „TD“ događaju.

Sada se mogu definirati funkcije faznih signala na sljedeći način:

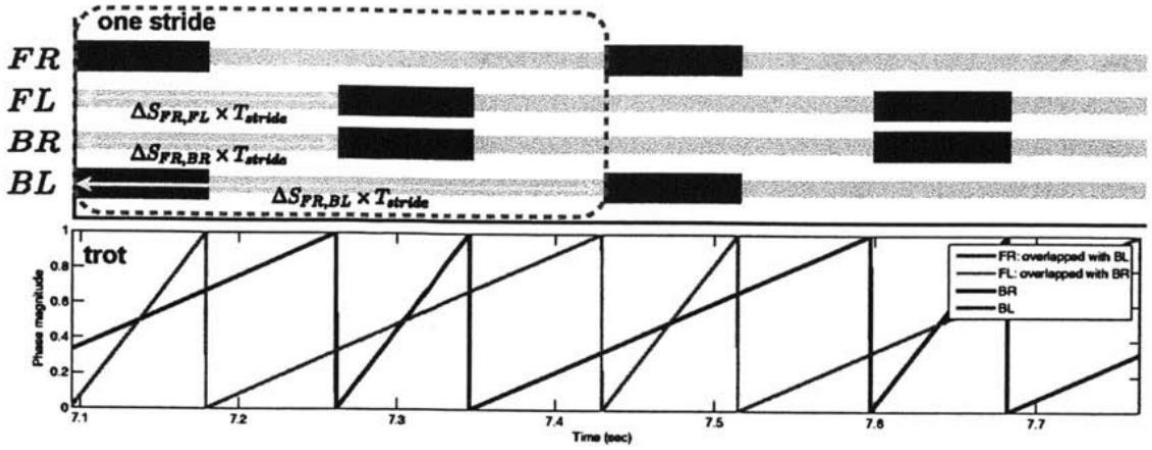
$$S_i^{st} = \frac{t_i}{T_{St}} \ , \ 0 < t_i < T_{St} \tag{11}$$

$$S_i^{sw} = \begin{cases} \dfrac{t_i + T_{sw}}{T_{sw}} \quad ako\ je\ -T_{sw} < t_i < 0 \\ \dfrac{t_i - T_{st}}{T_{sw}} \quad ako\ je\ T_{st} < t_i < T_{Stride} \end{cases} \tag{12}$$

Kada se zadaju parametri za tzv. troting u vektorskom obliku dani jednadžbom:

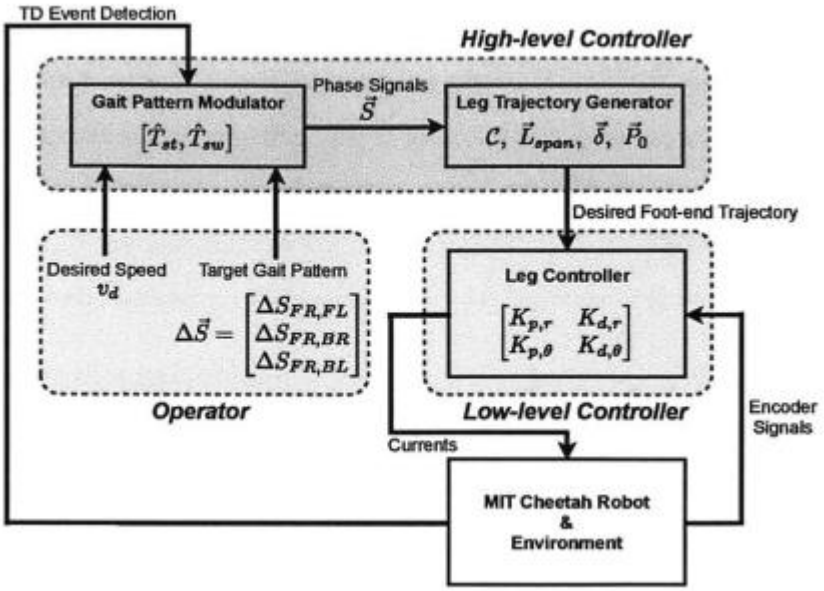
$$\Delta S_{trot} = \begin{bmatrix} \Delta S_{FR,BL} \\ \Delta S_{FR,BR} \\ \Delta S_{FR,BL} \end{bmatrix} = \begin{bmatrix} 0{,}5 \\ 0{,}5 \\ 0 \end{bmatrix} \tag{13}$$

može se generirati signal kako je prikazano na [Slika 22]



Slika 22. Troting hodanje koje se implementira u simulaciji[25]

Sada se putem zadanih parametara za sve dijelove može prikazati ukupna shema Champ algoritma što prikazuje [Slika 23].



Slika 23. Shema Champ algoritma[25]

Ako se pregleda poglavlje 5.1 može se vidjeti kako se dobije ulaz u sustav prikazan na gornjoj slici. Putem generiranih faznih signala [Slika 22] dolazi se do generatora hoda koji je definiran i objašnjen u poglavlju 3.1. On definira trajektoriju nogu u fazi zamaha i fazi stajanja. Na temelju željene trajektorije i podataka sa senzora uz regulaciju prikazanu na [Slika 25] dobiju se momenti koje motori na nogama moraju ostvariti, te se zna pozicija i brzina zakreta svih zglobova na robotu. Dalje će se proći najbitniji dijelovi ROS-a koje champ koristi.

5.2. Move_base Champ-a

Najvažniji dio koji Champ koristi jest u move_base dio koda kako on prima podatke o mapama troškova te definira planer koji će se koristiti za kretanje robota. On sadrži dva najbitnija dijela:

1. move_base_params

```
base_global_planner: global_planner/GlobalPlanner
base_local_planner: dwa_local_planner/DWAPlannerROS
shutdown_costmaps: false
controller_frequency: 6.0
controller_patience: 3.0
planner_frequency: 0.5
planner_patience: 5.0
oscillation_timeout: 10.0
oscillation_distance: 0.2
conservative_reset_dist: 0.1
```

gdje su najvažniji parametri:

- controller_frequency – frekvencija u Hz kada će se izvršiti petlja ragulacije i poslati komande za brzine kretanja.
- planner_frequency – frekvencija u Hz za izvšavanje globalne petlje planiranja.
- oscillation_timeout – koliko vremena u sekundama robot stoji ukoliko ne može izračunati put do cilja prije nego li pokrene gibanje za oporavak.
- Base_global_planer i base_local_planer – ime dodatka planera za definiranje putanje.

2. base_local_planer_holonomic_params

```
DWAPlannerROS:
  #http://wiki.ros.org/dwa_local_planner
  min_vel_trans: 0.01
  max_vel_trans: 0.8
  min_vel_x: -0.025
  max_vel_x: 0.8
  min_vel_y: 0.0
  max_vel_y: 0.0
  max_vel_rot: 0.8
  min_vel_rot: -0.8
  acc_lim_trans: 3.0
  acc_lim_x: 3.0
  acc_lim_y: 0.0
  acc_lim_theta: 5
  trans_stopped_vel: 0.1
  theta_stopped_vel: 0.1
  xy_goal_tolerance: 0.05
  yaw_goal_tolerance: 0.1
  sim_time: 3.5
  sim_granularity: 0.1
  vx_samples: 20
  vy_samples: 0
```

```
  vth_samples: 40
  path_distance_bias: 34.0
  goal_distance_bias: 15
  occdist_scale: 0.05
  forward_point_distance: 0.2
  stop_time_buffer: 0.5
  scaling_speed: 0.25
  max_scaling_factor: 0.2
  oscillation_reset_dist: 0.05
  use_dwa: true
  prune_plan: false
```

gdje su najvažniji parametri:

- xy_goal_tolerances – tolerancija u metrima za postizanje clja u smjerovima x i y ravnina.
- yaw_goal_tolerance – tolerancija u radijanima za zakretanje robota kada postigne cilj.
- path_distance_bias – Bias za namještanje koliko bi robot trebao biti blizu putanje koju je dobio.
- goal_distance_bias – Bias za namještanje koliko bi robot trebao postići svoj lokali cilj.
- occdist_scale – koliko bi robot trebao izbjegavati prepreke.

O opisu ostalih parametara više u [26] i [27].

5.3. Korišteni alati u Champ-u

Za rješavanje zadatka champ koristi sljedeće najbitnije čvorove u ROS-u:

1. Gmapping putem launch datoteke sa sljedećim prametrima

```
<launch>
    <arg name="frame_prefix" default=""/>

    <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping"
output="screen">
        <param name="base_frame" value="$(arg frame_prefix)base_footprint"
/>
        <param name="odom_frame" value="$(arg frame_prefix)odom" />
        <param name="map_frame"  value="/$(arg frame_prefix)map" />

        <param name="map_update_interval" value="1.0"/>
        <param name="maxUrange" value="5.0"/>
        <param name="minRange" value="0.5"/>
        <param name="sigma" value="0.05"/>
        <param name="kernelSize" value="1"/>
        <param name="lstep" value="0.05"/>
        <param name="astep" value="0.05"/>
        <param name="iterations" value="5"/>
        <param name="lsigma" value="0.075"/>
        <param name="ogain" value="3.0"/>
        <param name="lskip" value="0"/>
        <param name="minimumScore" value="200"/>
        <param name="srr" value="0.01"/>
```

```
        <param name="srt" value="0.02"/>
        <param name="str" value="0.01"/>
        <param name="stt" value="0.02"/>
        <param name="linearUpdate" value="0.5"/>
        <param name="angularUpdate" value="0.44"/>
        <param name="temporalUpdate" value="-1"/>
        <param name="resampleThreshold" value="0.5"/>
        <param name="particles" value="80"/>
        <param name="xmin" value="-50.0"/>
        <param name="ymin" value="-50.0"/>
        <param name="xmax" value="50.0"/>
        <param name="ymax" value="50.0"/>
        <param name="delta" value="0.05"/>
        <param name="llsamplerange" value="0.05"/>
        <param name="llsamplestep" value="0.05"/>
        <param name="lasamplerange" value="0.005"/>
        <param name="lasamplestep" value="0.005"/>
        <param name="transform_publish_period" value="0.1"/>
    </node>
</launch>
```

Gmapping tj. SLAM(Simultaneos localization and mapping) uzima poruke poslane od strane skeniranih podataka lasera te radi mapu putem nav_msgs/OccupanyGrid. Prednost mu je u tome da on radi u nepoznatom okruženju. Kada se mapiranje završi potrebno je pokrenuti mapserver mapsaver servis kako bi se ta mapa mogla spremiti. SLAM izdaje tri teme:

- MapMetaData i Ocuppancygrid – dobivaju se podaci mape, oni se zaključaju i povremeno ažuriraju kako se čitaju novi podaci.
- Entropy – procjena distribucije entropije po pozi robota (što je veći broj veća je nesigurnost)

S gornjeg koda može se prikazati najbitnije parametre gmappinga, a oni su:

- Map_update_interval – parametar koji prikazuje vrijeme u sekundama između ažuriranja mape, što je veći sustav je manje opterećen računanjem ali se mapa sporije generira.
- Iterations – broj iteracija ponavljanja podudaranja.
- MinimumScore – minimalni iznos kako bi se ishod skeniranja smatrao dobrim. Služi za izbjegavanje skakanja u procjeni položaja kada se robot nalazi na velikim otvorenim prostorima, a koristi laserske skenere ograničenog dometa. Time se dobije kriva pretpostavka da na nekoj udaljenosti postoji predmet, iako je on u stvarnosti nepostojeći, a robot samo nema mogućnost vida dalje u prostoru.
- Lstep – korak optimizacije pri translaciji.
- Astep – korak optimizacije pri rotaciji.
- Occ_thresh – prag na vrijednostima popunjenosti (eng. ocuppancy) gdje se ćelije s većim iznosom smatraju zauzetima tj. na ti mjestima se nalazi prepreka.

O opisu ostalih parametara više u [17].

2. Amcl putem sljedeće launch datoteke

```
<launch>
    <arg name="frame_prefix" default=""/>

    <node pkg="amcl" type="amcl" name="amcl" output="screen">
         <!-- <param name="initial_pose_x" value="1.5"/>
        <param name="initial_pose_y" value="1.5"/>
        <param name="initial_pose_a" value="0.0"/>  -->
        <param name="use_map_topic" value="true"/>
        <param name="base_frame_id" value="$(arg
frame_prefix)base_footprint"/>
        <param name="odom_frame_id" value="$(arg frame_prefix)odom"/>
        <param name="global_frame_id" value="$(arg frame_prefix)map"/>
        <param name="transform_broadcast" value="true"/>
        <param name="gui_publish_rate" value="100"/>
        <param name="kld_err" value="0.05"/>
        <param name="kld_z" value="0.99"/>
         <param name="laser_lambda_short" value="0.1"/>
        <param name="laser_likelihood_max_dist" value="2"/>
        <param name="laser_max_beams" value="60"/>
        <param name="laser_model_type" value="likelihood_field_prob"/>
        <param name="laser_sigma_hit" value="0.2"/>
        <param name="laser_z_hit" value="0.5"/>
        <param name="laser_z_short" value="0.05"/>
        <param name="laser_z_max" value="0.05"/>
        <param name="laser_z_rand" value="0.5"/>
        <param name="max_particles" value="1000"/>
        <param name="min_particles" value="500"/>
        <param name="odom_alpha1" value="0.6"/>
        <param name="odom_alpha2" value="0.6"/>
        <param name="odom_alpha3" value="0.6"/>
        <param name="odom_alpha4" value="0.6"/>
        <param name="odom_alpha5" value="0.6"/>
        <param name="odom_model_type" value="omni-corrected"/>
        <param name="recovery_alpha_slow" value="0.001"/>
        <param name="recovery_alpha_fast" value="0.1"/>
        <param name="resample_interval" value="1"/>
        <param name="transform_tolerance" value="0.1"/>
        <param name="update_min_a" value="0.1"/>
        <param name="update_min_d" value="0.1"/>
    </node>
</launch>
```

Amcl je probalistitički sustav lokalizacije robota koji se kreće u 2D. Koristi Monte Carlo lokalizacijski pristup koji koristi filtar čestica za praćenje položaja robota u već preddefiniranoj mapi. Najbitniji parametri amcl-a su:

- max_particles i min_particles – definiraju maksimalnu i minimalnu količinu čestica. Što je više čestica više podataka o poziciji robota je dostupno te se može preciznije lokalizirati.

- odom_model_type – koristi se za definiranje kretanja baze, u ovom slučaju je odabran omni-corrected pošto se baza može gibati u sve smjerove.
- gui_publish_rate – frekvencija u Hz kojom su putanje i skeniranja izdana za vizualizaciju.
- kld_err – maksimalna pogreška prave i estimirane distribucije čestica.

Više o opisu amcl-a i njegovih parametara dane su u [19].

3. Champ_teleop

Champ_teleop je čvor koji služi za upravljanje robotom sa tipkovnicom ili joystickom. Na [Slika 24] prikazane su mogućnosti kretanja tipkovnicom.

```
Reading from the keyboard  and Publishing to Twist!
---------------------------
Moving around:
u    i    o
j    k    l
m    ,    .
For Holonomic mode (strafing), hold down the shift key:
---------------------------
U    I    O
J    K    L
M    <    >
t : up (+z)
b : down (-z)
anything else : stop
q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
CTRL-C to quit

currently:      speed 0.5       turn 1.0
```

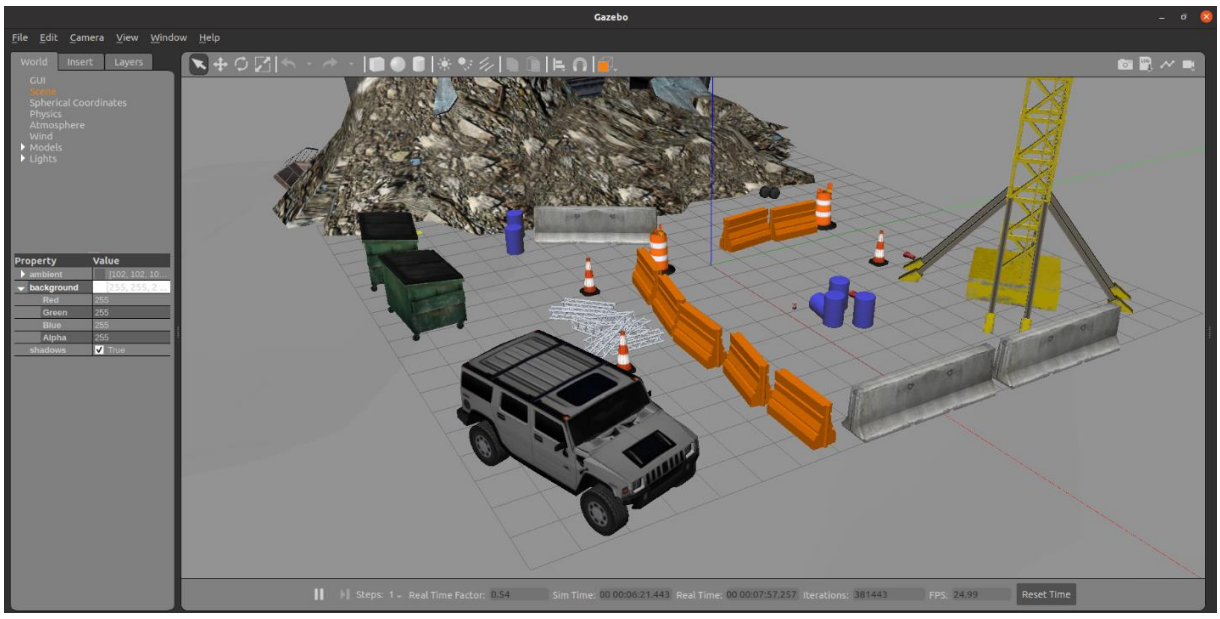
Slika 24. Mogućnosti kretanja robota tipkovnicom

Ukoliko se koristi joystick može se simulirati više kretnji robota kao što je npr. zakretanje baze.

6. IMPLEMENTACIJA CHAMP-A

6.1. Virtualno okruženje robota

Prvi korak jest napraviti virtualno okruženje robota odnosno okruženje u kojem će se moći testirati robot sa svim svojim specifikacijama te funkcijama koje algoritam nudi. Kako Champ algoritam nije namijenjen uspinjanju robota po rampi i stepenicama mora imati ravnu podlogu da se robot može kretati normalno. Na [Slika 25] prikazano je okruženje u Gazebo-u kojem će se robot nalaziti nazvano outdoor.world.



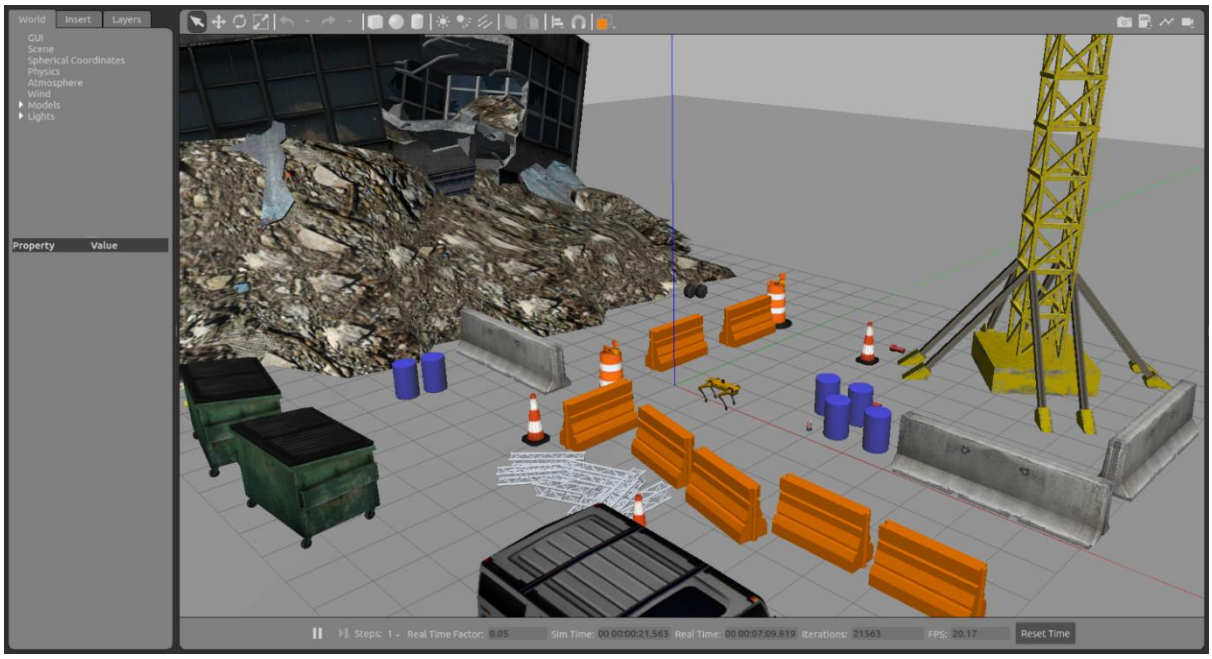
Slika 25. Virtualno okruženje robota u Gazebo-u

6.2. Simulacija mapiranja i stvaranja robota u virtualnom okruženju

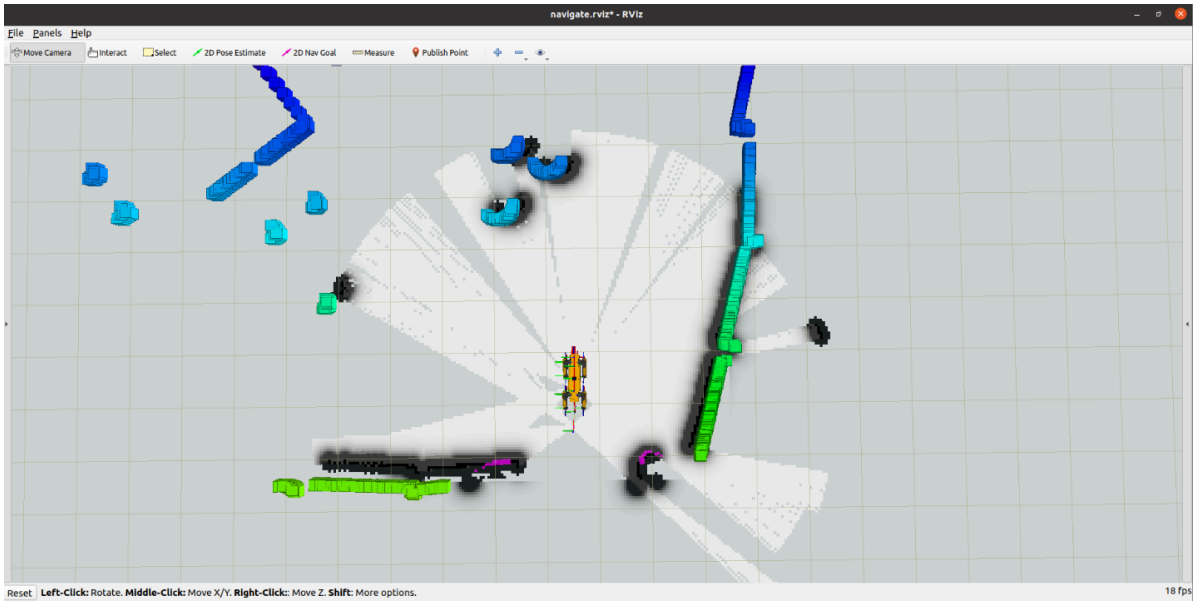
Kako bi se pokrenula simulacija robota u Gazebo-u potrebno je napraviti launch datoteku koja će pokrenuti Gazebo te stvoriti outdoor.world, stvoriti robota, pokrenuti vizualizaciju mapiranja u RViz-u i omogućiti kretanje robota tipkovnicom. Datoteka se izvršuje naredbom u terminalu koja glasi: roslaunch spot_config launch_m_w_r.launch .

Prilikom mapiranja robot se služi ugrađenim kamerama kao senzorima kako bi mogao napraviti mapu, a na bazi robota se nalazi i senzorom sa radnim prostorom od 30m i 270°. Pokretanje robota vrši se tipkovnicom te su pritom korištene sljedeće kretnje sa pripadajućim tipkama na tipkovnici: naprijed(i), iza(m) , zakret ulijevo(j), zakret udesno(l), što je prikazano na [Slika 24].

Za mapiranje se koristi gmapping koji je sustav lokalizacije te u isto vrijeme kad radi lokalizaciju, radi i mapiranje. Kada se u terminalu izvrši pozivanje prije prikazane launch datoteke i pokrene robota par koraka naprijed može se vidjeti da se je mapa krenula generirati sa preprekama i slobodnim dijelovima zakretanje. Na ekranu se nalazi sljedeće, prikazano na [Slika 26] i [Slika 27].

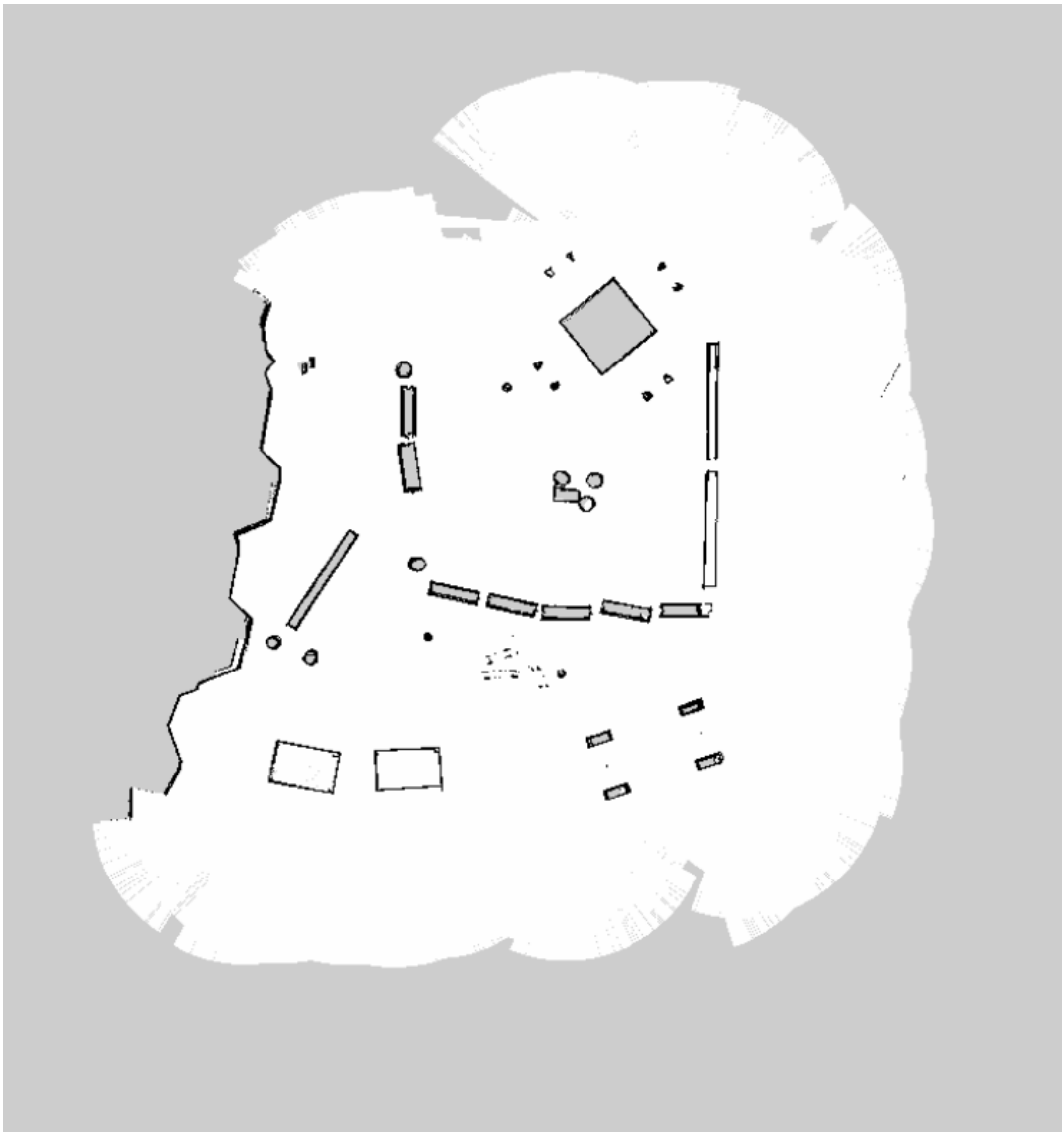


Slika 26. Prikaz robota u Gazebo-u unutar virtualnog okruženja



Slika 27. Mapiranje u prvih par koraka robota

Kada se mapiranje završi dobije se kompletna mapa prostora u kojem se robot nalazi prikazan kao 2D mapa sa generiranim dijelovima mape koji su zauzeti tj. sa preprekama koje se nalaze u virtualnom okruženju. Kako bi se ta mapa sačuvala potrebno ju je spremiti. Pri tome se je potrebno služiti paketom iz ROS-a imena map_server. On sadrži map_serveer koji daje podatke karte i map_saver koji sprema generiranu mapu u datoteku. [Slika 28] prikazuje generiranu mapu virtualnog okruženja. Svijetla bijela boja prikazuje slobodan prostor, crne linije prepreke, a siva boja neistraženi prostor.



Slika 28. Generirana mapa virtualnog okruženja robota

7. PRAĆENJE PREDDEFINIRANE PUTANJE I LOKALIZACIJA ROBOTA

Kako je navigacija robota u prostoru zahtjevna disciplina valja je dobro opisati. Robot pri tome mora znati svoj položaj u zadanoj okolini, mora sakupljati i čitati podatke sa senzora, mora odlučiti kojim putem će doći do cilja te kontrolirati svoje izlaze kako bi mogao pratiti odlučeni put te doći do samog cilja.

Launch datoteka mora se napraviti tako da ima sljedeće funkcije: pokrenuti Gazebo i učitati svijet outdoor.world, stvoriti robota na inicijalnoj poziciji, pokrenuti Rviz te se robot mora lokalizirati u generiranoj mapi iz prošlog poglavlja. Nakon toga robotu je potrebno zadati cilj do kojeg treba doći putem 2D Nav Goal alata u rvizu. Kako bi došao do cilja mora napraviti plan kretanja i reagirati na potencijalne prepreke na putu.

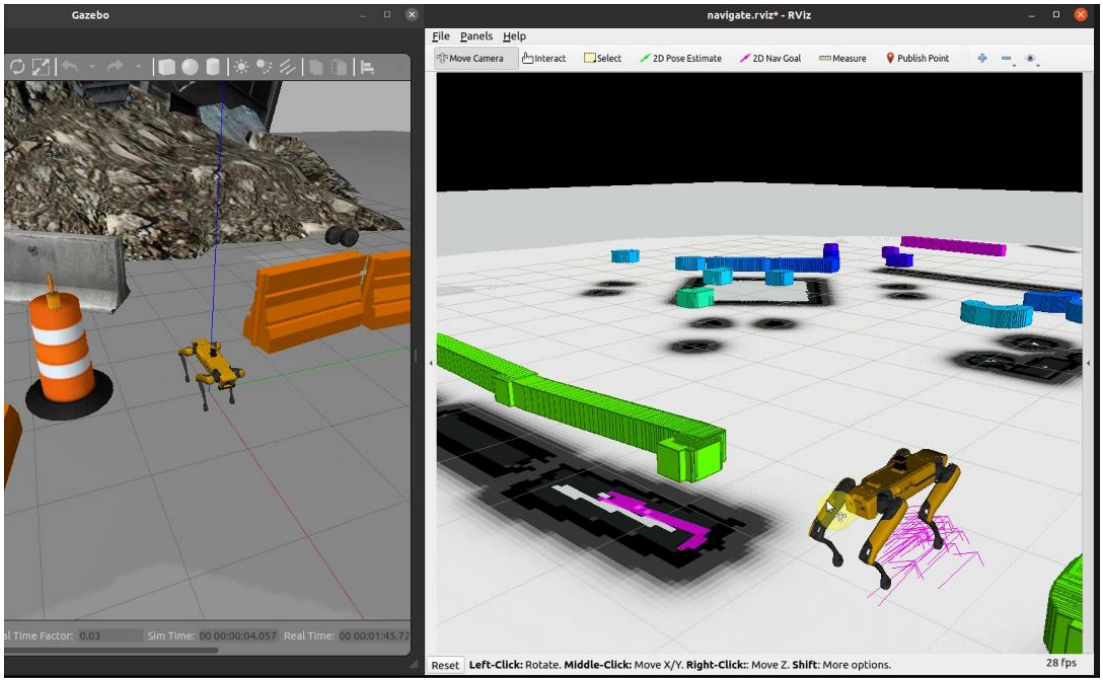
Algoritam koji služi za lokalizaciju naziva se Amcl te on zahtijeva statičnu kartu koju se generiralo u prošlom poglavlju [Slika 28]. Ono što taj algoritam radi jest da raspoređuje čestice na poznatoj mapi te tako pretpostavlja lokaciju robota, a potom filterom određuje pravu poziciju robota. U inicijalnom položaju robot je predstavljen na krivim koordinatama pa je potrebno otprilike unijeti pravu poziciju robota te ga nakon toga provesti par koraka kako bi položaj čestica konvergirao i kako bi se robot mogao lokalizirati.

Putanja se generira putem algoritma lokalnog planera, u ovom slučaju DWA algoritma. Taj algoritam radi u par koraka. Prvo se sampliraju brzine, potom se izvodi simulacija da se identificira moguće kretanje robota. Nakon toga rezultati dobiveni prethodnim računom se uspoređuju pomoću funkcija troškova čiji se Bias-i mogu definirati, a potom ona putanja koja ima najmanji trošak šalje se kontroleru kako bi se robot pokrenuo.

Kada robot dođe u željenu poziciju preko terminala se javlja da je cilj postignut i robot se zaustavlja.

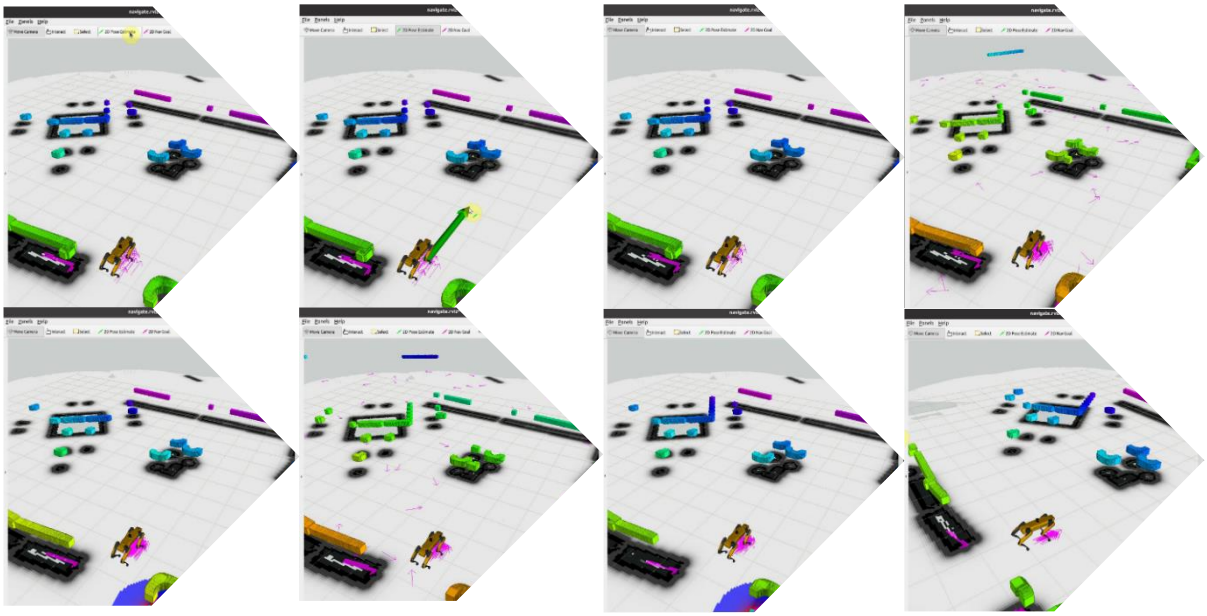
7.1. Lokalizacija robota na mapi

Na [Slika 29] prikazana je inicijalna reprezentacija robota u RViz-u sa oblakom strelica(rozo) u odnosu na pravu u Gazebo simulatoru.



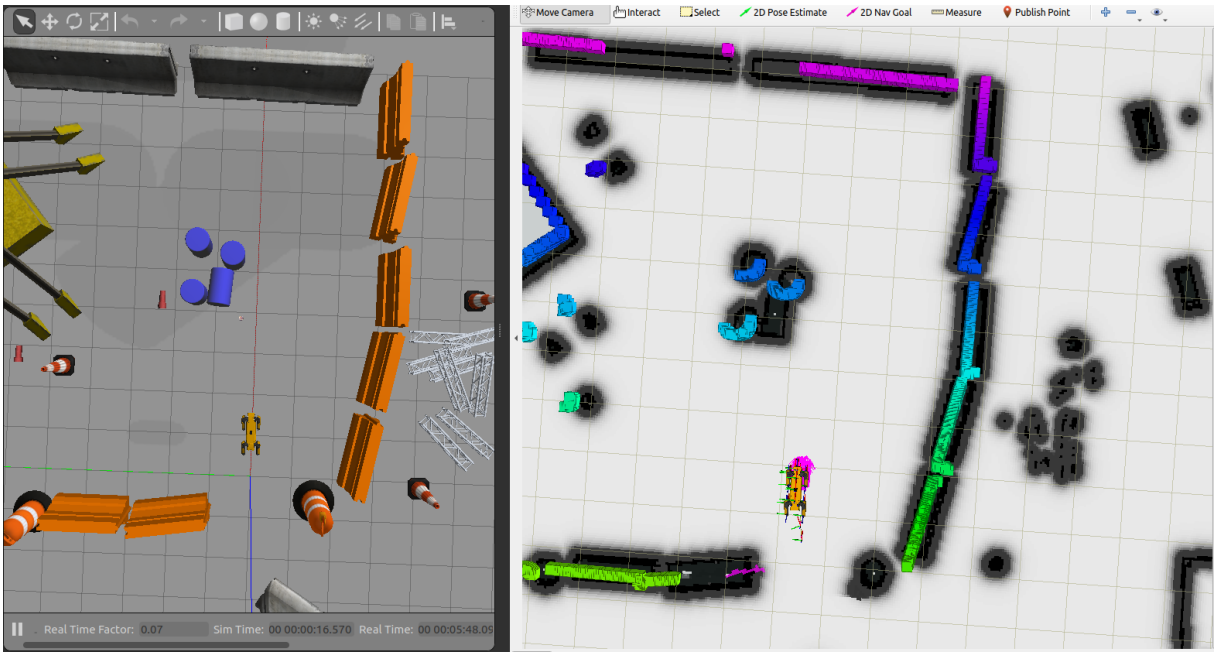
Slika 29. Prva reprezentacija lokacije robota

Kako bi se robot lokalizirao i kako bi simulacija dala što bolju naznaku prostora robota potrebno je u RViz-u koristiti 2D Pose Estimate alat koji stvara oblak čestica koje reprezentiraju lokaciju robota. Tada je potrebno robotu preko tipkovnice zadati kretanje te se pri njemu čestice pomiču tako da konvergiraju prema poziciji robota, a stvaraju se i nove čestice kako bi robot imao više podataka o poziciji. Kada se oblak čestica nalazi dovoljno blizu robota može se smatrati da se robot lokalizirao. [Slika 30] prikazuje proces lokalizacije robota.



Slika 30. Proces lokalizacije

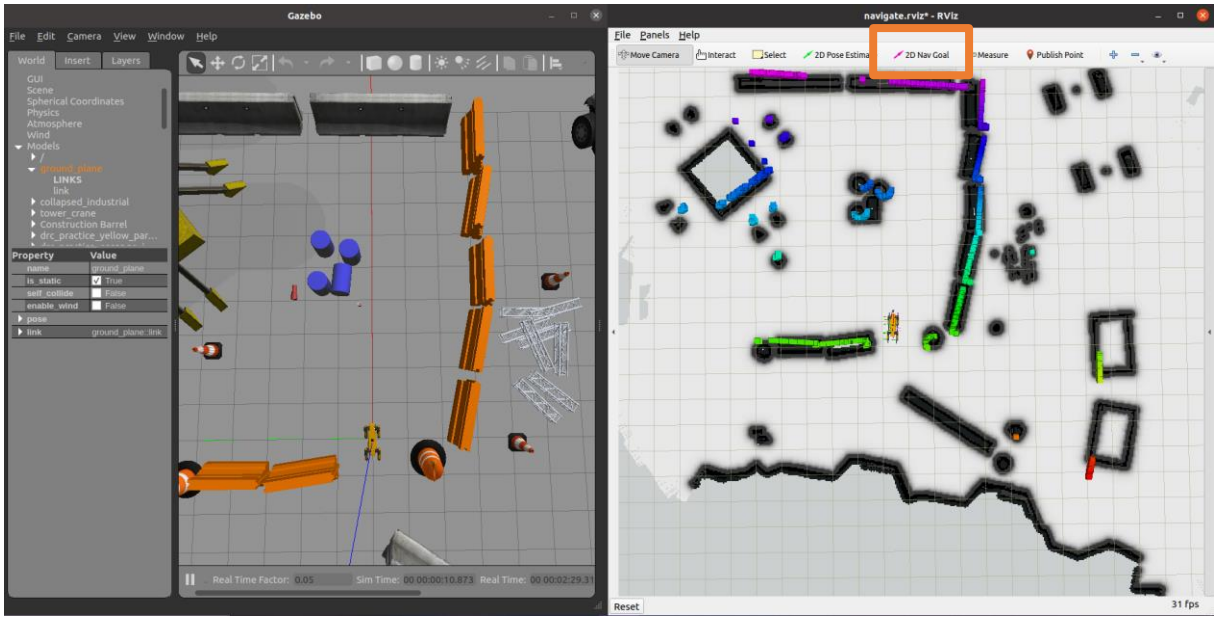
Kako je robot lokaliziran može se pokazati i usporedba lokacije u RViz-u i Gazebo-u. [Slika 31] prikazuje tu usporedbu.



Slika 31. Usporedba lokacije u Gazebo-u i RViz-u

7.2. Simulacija autonomnog kretanja robota

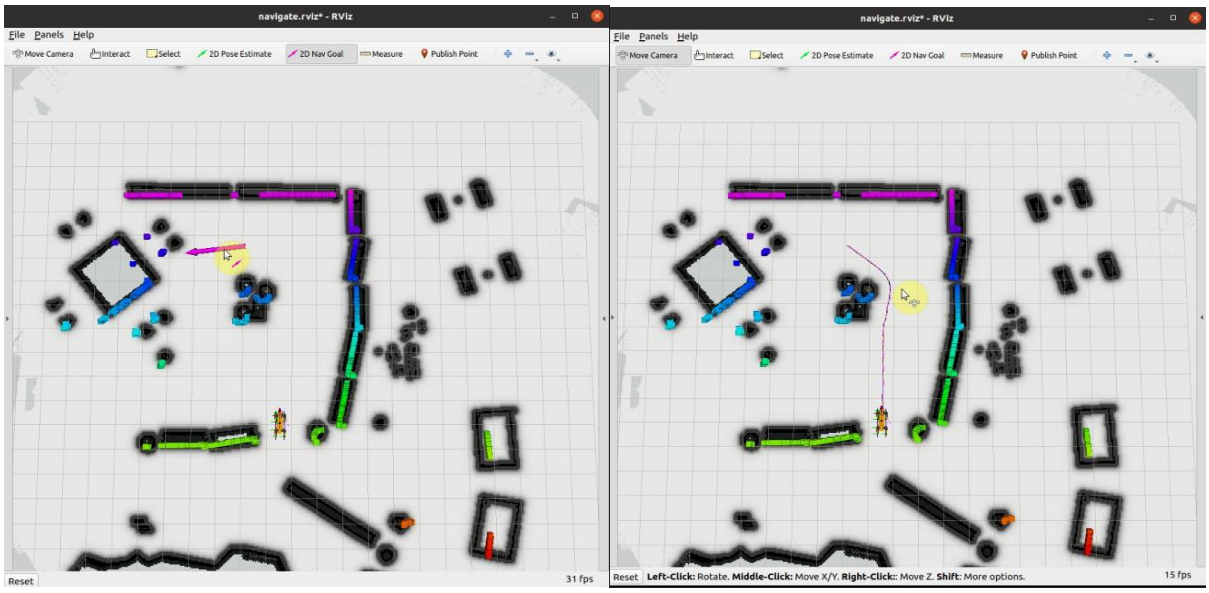
Kada se izvrše komande za pokretanje navigacije i robot se lokalizira dobije se prikaz kao na [Slika 32].



Slika 32. Prikaz ekrana prilikom pokretanja navigiranja

Nadalje je robotu potrebno zadati cilj putem 2D Nav Goal alata koji je na gornjoj slici obrubljen crveno.

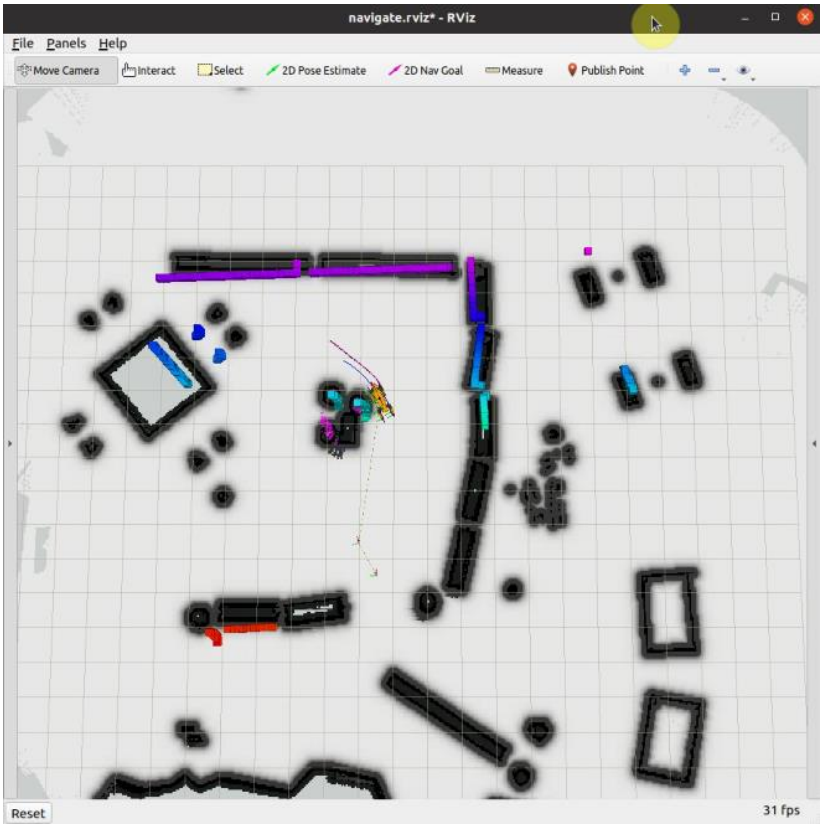
Na [Slika 33] prikazan je željeni cilj robota te prva putanja koja se generirala.



Slika 33. Željeni cilj robota i generirana putanja

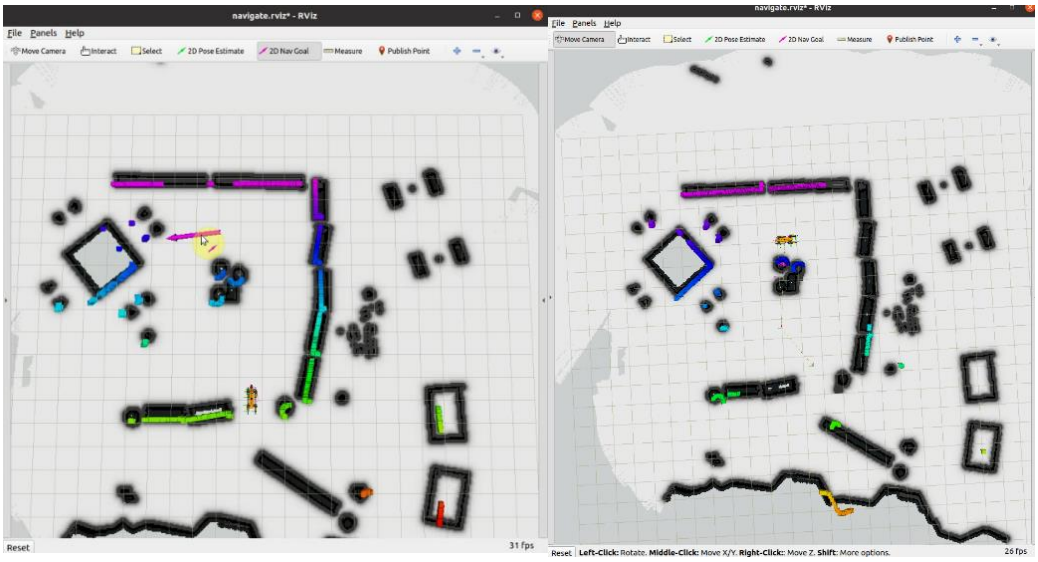
Gornja slika prikazuje generiranu putanju globalnog planera, ali lokalni planer radi na malo drugačiji način pa tako on da prije dođe do cilja, radi određena odstupanja s globalne putanje.

Primjer razlike globalne i lokalne putanje prikazan je na [Slika 34].



Slika 34. Razlika globalne i lokalne putanje

Nadalje ostaje napraviti usporedbu željenog cilja i postignutog cilja robota što se vidi na [Slika 35].



Slika 35. Usporedba željenog cilja i postignutog cilja

8. ZAKLJUČAK

U ovom radu prikazali su se četveronožni roboti te jedan od načna kako se oni mogu simulirati za moguće testiranje u nekom okruženju. Primjena takvih robota postaje sve značajnija te se mogu koristiti u raznim okruženjima. Primjeri primjena su bilo kakva vizualna inspekcija, mapiranje neistraženog prostora i sl. Prikazali su se komercijalno dostupni četveronožni roboti te se prikazalo njihove prednosti u odnosu na mobilne robote na kotačima. Kako je spomenuto glavna prednost im je mogućnost kretanja na nesređenim površinama. Nadalje se trebalo usporediti te robote kako bi se prikazala njihova primjerena primjena te mogućnosti koje oni pružaju. Prednost kod razvijanja četveronožnih robota su dostupna gotova rješenja algoritama kretanja koji pružaju veliko olakšanje pri izradi novi robota. Prikazan je kratki pregled algoritama hodanja sa gotovo svim mogućnostima kretanja. U ovom slučaju se koristio Champ paket koji sadrži algoritam hodanja, konfiguraciju robota te setup asistenta za moguću izradu novog robota hodača pri čemu bi se postupak razvoja ovakvim open source paketom značajnno ubrzao. Robot koji je konfigurin jest Spot te je on opisan urdf datotekom. Također je u radu prikazano kako je moguće simulirati jedan od tih robota (Spot) te implementirati algoritam u simulaciju. U simulaciji je tada potrebno sa ROS-om upravljati robotom, iščitavati podatke sa senzora, slati podatke o transformacijama i sl. Nadalje se prikazalo kako se može mapirati virtualno okruženje u kojem se nalazi robot putem gmappinga. Kada se generirala mapa virtualnog okruženja robot se morao lokalizirati u istoj što se ostvarilo pomoću Amcl paketa. Lokaliziranog robota se tada moglo poslati na bilo koju točku i orijentaciju u mapi te je on mogao izbjegavati prepreke na putu. Kada se simulacija testirala vidjelo se kako su rezultati zadovoljavajući putem DWA planera.

LITERATURA

[1] https://www.ros.org/ 06.02.2023

[2] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8725662/ 18.02.2023

[3] https://www.ros.org/ 06.02.2023

[4] https://www.anybotics.com , 10.02.2023.

[5] https://www.bostondynamics.com/ , 10.02.2023.

[6] https://shop.unitree.com/ 10.02.2023.

[7] http://www.weilan.com/ 10.02.2023.

[8] https://robots.ieee.org/robots/minicheetah/ 10.02.2023.

[9] https://www.ghostrobotics.io/vision-60 10.02.2023.

[10] https://hiwonder.hk/ 10.02.2023.

[11] https://github.com/chvmp/champ 11.02.2023.

[12] https://github.com/stanfordroboticsclub/StanfordQuadruped 11.02.2023.

[13] https://github.com/robomechanics/quad-sdk 11.02.2023.

[14] https://github.com/ethz-adrl/towr 11.02.2023.

[15] https://github.com/OpenQuadruped/spot_mini_mini 11.02.2023.

[16] https://sites.google.com/view/drgmbc 11.02.2023.

[17] http://wiki.ros.org/gmapping 18.02.2023.

[18] http://wiki.ros.org/map_server 18.02.2023.

[19] http://wiki.ros.org/amcl 18.02.2023.

[20] https://n1info.hr/wp-content/uploads/2021/06/robot--1024x576.jpg 21.02.2023

[21] https://robots.ieee.org/robots/alphadog/alphadog-thumb@2x.jpg 21.02.2023

[22] https://i2-prod.dailystar.co.uk/incoming/article24531814.ece/ALTERNATES/s1200d/0_download.jpg 21.02.2023

[23] https://news.mit.edu/sites/default/files/download/201903/MIT-Mini-Cheetah-01-PRESS.jpg 21.02.2023

[24] https://www.research-collection.ethz.ch/bitstream/handle/20.500.11850/272432/thesis_winkler.pdf?sequence=1&isAllowed=y 21.02.2023

[25] https://dspace.mit.edu/handle/1721.1/85490 18.02.2023.

[26] http://wiki.ros.org/move_base 23.02.2023.

[27] http://wiki.ros.org/dwa_local_planner 23.02.2023.