Automatska kalibracija mobilnog robota

Kolar, Luka

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Luka Kolar

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

Doc. dr. sc. Marko Švaco, mag. ing. mech.

Student:

Luka Kolar

Zagreb, 2023.

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

Zahvaljujem mentoru Doc. dr. sc. Marku Švaci, mag. ing. te Branimiru Ćaranu, mag. ing. na dostupnosti, pristupačnosti, savjetima i pomoći prilikom izrade ovog završnog rada.

Zahvaljujem svojim roditeljima Zlatku i Jadranki, bratu Ivanu, djevojci Mariji i svim svojim prijateljima na podršci i pomoći tijekom studiranja.

Luka Kolar

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: **Luka Kolar** JMBAG: **0035215538**

Naslov rada na hrvatskom jeziku: **Automatska kalibracija mobilnog robota**

Naslov rada na engleskom jeziku: **Automatic mobile robot calibration**

Opis zadatka:

Osigurati što točniju estimaciju odometrije mobilnog robota izazovan je zadatak. Na estimaciju odometrije utječu sustavne i nesustavne greške. Sustavne greške najčešće su posljedica proizvodnih procesa komponenata robota te njihove završne montaže, dok nesustavne greške najčešće proizlaze iz interakcije robota s okolinom; npr. trenje kotača i podloge i na takve faktore je teško utjecati. Dvije sustavne greške koje najznačajnije utječu na estimaciju odometrije mobilnog robota su nejednolik promjer kotača i dimenzijske netočnosti udaljenosti između kotača. UMBMark je standardna metoda za ispitivanje točnosti mobilnih robota i njihovu kalibraciju. Metoda se primjenjuje tako da se robotu zada referentna kvadratna putanja dimenzija 4x4 m te se mjere odstupanja prilikom izvođenja referentne putanje. Sukladno tome računaju se kalibracijski parametri koji se upisuju u jednadžbu direktne kinematike mobilnog robota. Za kalibraciju mobilnog robota, potrebno je koristiti vanjski referentni lokalizacijski sustav, a u današnje vrijeme sve više se koriste vizijski sustavi, poput OptiTrack-a.

U završnom radu na dostupnoj opremi u Laboratoriju za računalnu inteligenciju potrebno je:

- Pokrenuti i kalibrirati vanjski vizijski sustav OptiTrack.
- Omogućiti komunikaciju OptiTrack-a s ROS-om.
- Napraviti čvor u ROS-u za generiranje i slanje kvadratne putanje (4x4 m) mobilnom robotu.
- Usporediti referentnu kvadratnu putanju sa stvarnom putanjom koju je robot obišao i grafički prikazati rezultate.
- Napraviti čvor u ROS-u koji će korištenjem početne i krajnje lokacije mobilnog robota izračunati kalibracijske parametre robota.
- Implementirati kalibracijske parametre na mobilnom robotu i usporediti praćenje putanje prije i nakon kalibracije mobilnog robota.

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

Zadatak zadan:
30. 11. 2022.

Datum predaje rada:
1. rok: 20. 2. 2023.
2. rok (izvanredni): 10. 7. 2023.
3. rok: 18. 9. 2023.

Predviđeni datumi obrane:
1. rok: 27. 2. – 3. 3. 2023.
2. rok (izvanredni): 14. 7. 2023.
3. rok: 25. 9. – 29. 9. 2023.

Zadatak zadao:
Doc. dr. sc. Marko Švaco

Predsjednik Povjerenstva:
Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
x	m	Pozicija robota u smjeru osi X
y	m	Pozicija robota u smjeru osi Y
θ	rad	Orijentacija robota u X-Y ravnini
v	m/s	Translacijska brzina robota
v_r	m/s	Translacijska brzina desnog kotača
v_l	m/s	Translacijska brzina lijevog kotača
ω	rad/s	Kutna brzina robota
ω_r	rad/s	Kutna brzina desnog kotača
ω_l	rad/s	Kutna brzina lijevog kotača
r	m	Polumjer kotača robota
$\dot{x}$	m/s	Brzina robota u smjeru osi X
$\dot{y}$	m/s	Brzina robota u smjeru osi Y
$\dot{\theta}$	rad/s	Rotacijska brzina robota u X-Y ravnini
b	m	Razmak među kotačima robota
E_d	-	Kalibracijski parametar promjera kotača
E_b	-	Kalibracijski parametar udaljenosti među kotačima
$x_{c.g.}$	m	Težište klastera u smjeru osi X
$y_{c.g.}$	m	Težište klastera u smjeru osi Y
$r_{c.g.}$	m	Apsolutna udaljenost težišta klastera od ishodišta
$E_{max,syst}$	m	Maksimalna sustavna pogreška robota
L	m	Duljina stranice kvadratne trajektorije
α	rad	Kut odstupanja
β	rad	Kut odstupanja
R	m	Polumjer zakrivljenosti prilikom robotovog gibanja ravno

SAŽETAK

Zadatak ovog rada je automatska kalibracija mobilnog robota koja se obavlja tako da se robot upravljan pomoću ROS-a *(engl. Robot Operating System – ROS),* u kojem će se pomoći čvora generirati i slati kvadratna putanja veličine 4x4 m. Nakon toga, pomoću vizijskog sustava *OptiTrack* će se bilježiti stvarna putanja robota te usporediti s referentnom putanjom. Idući korak je primjena čvora u ROS-u koji korištenjem početne i krajnje lokacije bilježene pomoću OptiTrack-a izračunava kalibracijske parametre robota po UMBMark metodi kalibracije te njihova implementacija radi ispravljanja greški odstupanja od referentne putanje. Na kraju, pomoću grafičkog prikaza se uspoređuje praćenje putanje prije i nakon kalibracije mobilnog robota.

Ključne riječi: mobilni robot, UMBMark metoda kalibracije, odometrija, ROS (Robot Operating System), OptiTrack

SUMMARY

The task of this final paper is the automatic calibration of a mobile robot, which is performed by controlling the robot using ROS (Robot Operating System - ROS), in which a node will be used to generate and send a 4x4 m square path. After that, the actual path of the robot will be recorded by OptiTrack vision system and compared to the reference path. The next step is using a node in ROS that uses the initial and final location recorded by OptiTrack and calculates the calibration parameters of the robot according to the UMBMark calibration method and their implementation in order to correct errors of deviation from the reference path. At the end, the tracking of the trajectories before and after the calibration of the mobile robot is compared using a graphical display.

Key words: mobile robot, UMBMark calibration method, odometry, ROS (Robot Operating System), OptiTrack

1. UVOD

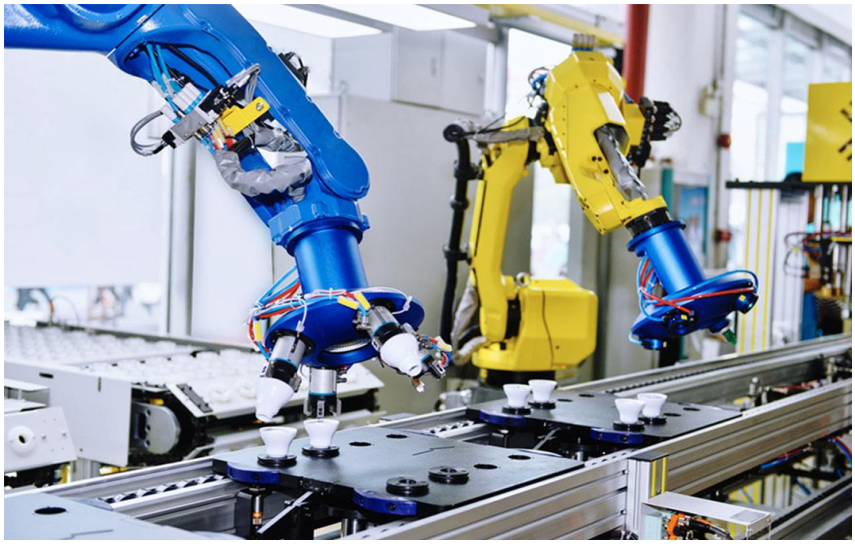
Robotika je interdisciplinarni sektor znanosti i inženjerstva posvećen dizajnu, konstrukciji i korištenju mehaničkih robota u svrhu obavljanja zadataka te rješavanja praktičnih problema koji svake godine doživljava sve veći rast. Objedinjuje područja poput strojarstva, elektrotehnike i računarstva te se koristi u područjima poput proizvodnje, medicine, vojne industrije, održavanja te mnogim drugim. Prednosti robotike su bolja ponovljivost i veća preciznost u radu od čovjeka te eliminacija opasnih poslova zbog sposobnosti obavljanja rada u teškim i riskantnim okruženjima.

Jedna od grana robotike je i mobilna robotika čija je glavna karakteristika sposobnost kretanja robota po prostoru. Mobilni roboti mogu biti zasebni ili u kombinaciji s instaliranim industrijskim robotima na sebi što omogućuje veći stupanj autonomnosti. Kako bi kretanje bilo što kvalitetnije, potrebno je što bolje pozicionirati robota, odnosno osigurati što točniju estimaciju odometrije. Na estimaciju odometrije najznačajnije utječu parametri poput nejednolikog promjera kotača i dimenzijske netočnosti udaljenosti između kotača. Iz tog razloga, u ovom radu će biti obrađena kalibracija mobilnog robota koja se sastoji od izračuna tih parametara te njihove implementacije u jednadžbu direktne kinematike mobilnog robota. Za upravljanje robotom i njegovu kalibraciju će biti korišten ROS te vanjski vizijski sustav OptiTrack koji će biti objašnjeni u nastavku. Budući da ROS nije samostalan operacijski sustav, potrebno je odabrati temeljni operacijski sustav na kojem će biti instaliran. Odabran je Linux Ubuntu koji je i temeljna platforma za korištenje ROS-a.

2. MOBILNA ROBOTIKA

Zbog brojnih pogodnosti kao što su brzina i ponovljivost izvođenja zadataka, visoka nosivost i točnost pozicioniranja, industrijski roboti su doživjeli velik uspjeh u proizvodnji. Unatoč tome, ograničen domet povezan s određenim radnim prostorom i nemogućnost kretanja su im glavni nedostatci. S druge strane, mobilni roboti nadilaze te probleme te im je kretanje u prostoru (tlo, zrak ili voda) glavna karakteristika.

Razlika u konstrukciji između mobilnih i industrijskih robota je prikazana na slikama 1. i 2.

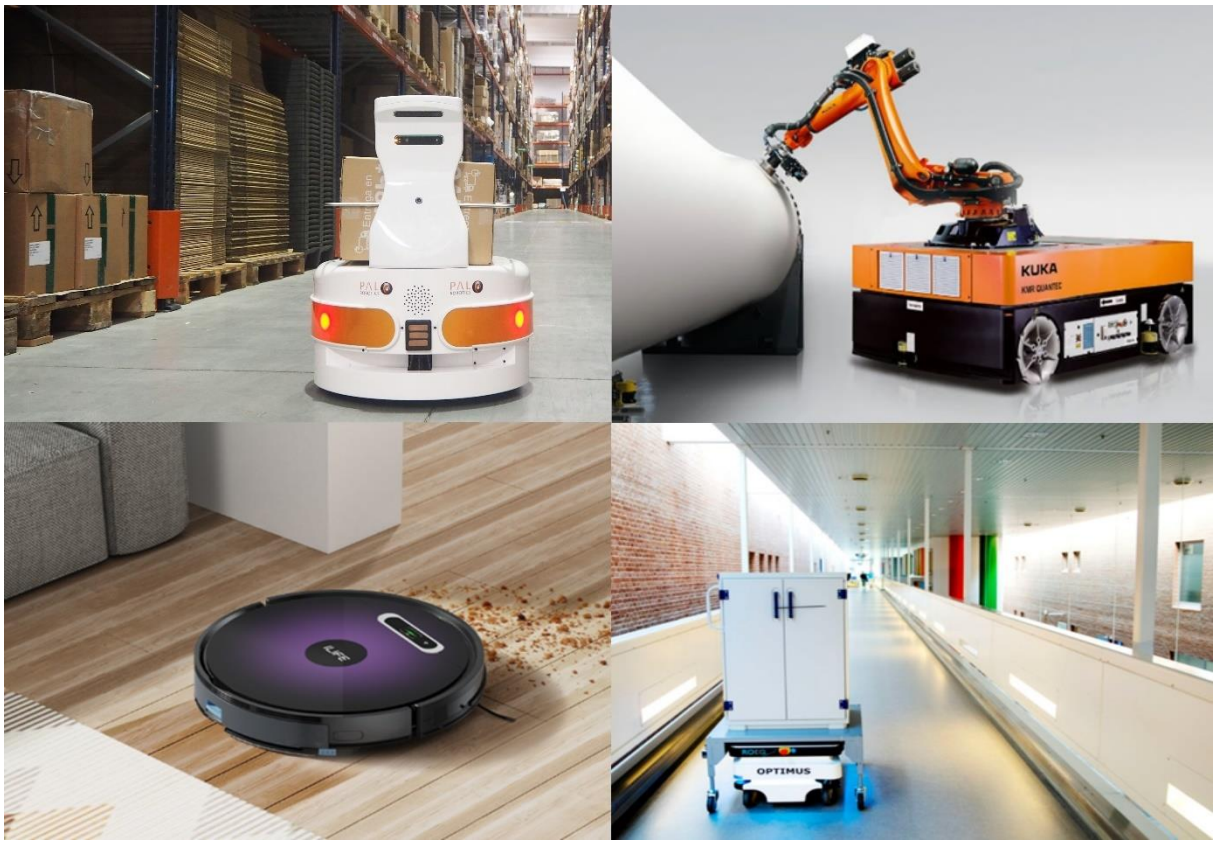


Slika 1. Industrijski roboti [2]

Slika 2. Mobilni robot [3]

Premda mobilni roboti imaju širok skup primjena (koje će biti navedene u nastavku) upravo zbog karakteristike kretanja, za kvalitetnog mobilnog robota potrebna je integracija znanja mnogih inženjerskih disciplina. Za probleme kretanja potrebno je razumijevanje i primjena mehanike, kinematike, dinamike, upravljanja i regulacije. Stvaranje robusne percepcije okoline robota podrazumijeva snalaženje u područjima poput računalnog vida, implementacija senzora koji pretvaraju fizikalne veličine u signale i njihova obrada pomoću algoritama unutar upravljačkog sklopa. Lokalizacija i navigacija robota kao i izbjegavanje prepreka također zahtijeva poznavanje algoritama te umjetne inteligencije.

Iako je rješavanje problema u nedostižnim ili opasnim okolinama za ljude jedan od glavnih motiva razvoja mobilne robotike, njihova najčešća primjena je ipak u skladištima (transport paketa, čišćenje skladišta), industriji (mobilni roboti s integriranim industrijskim robotom – premještanje radne stanice), kućanstvima (robotski usisavač) te medicini (transport sterilne robe). Primjene su prikazane na slici 3. navedenim redom.



Slika 3. Primjena mobilnih robota

2.1. Sustavi mobilnih robota

Kako je već navedeno, kvalitetnog mobilnog robota čini integracija znanja mnogih inženjerskih disciplina. Stoga, isti se sastoji od nekolicine podsustava koji se implementiraju na njegovu konstrukciju te u međusobnoj interakciji omogućuju kretanje i kvalitetno odrađivanje zadanih zadataka. Ti podsustavi su:

- *Upravljački sustav.* Osnovna jedinica sustava je mikrokontroler koji je zaslužan za prikupljanje senzorskih podataka, donošenje odluka te slanje upravljačkih naredbi aktuatorima.
- *Senzorski sustav.* Služi za prikupljanje informacija iz okoline u svrhu ostvarivanja percepcije iste te mjerenje unutarnjih stanja robota. Rezultat je snalaženje u prostoru i izbjegavanje prepreka te primanje informacija o brzini robota, kutevima zakreta motora i slično.
- *Aktuatorski sustav.* Najčešće sadrži elektromotore istosmjerne struje opremljene s enkoderima radi procjene unutarnjeg stanja robota.

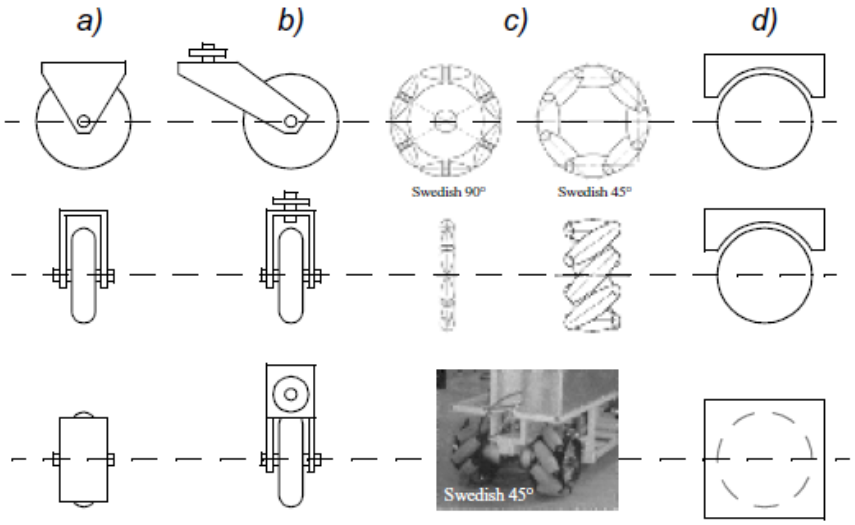
Dakle, nakon primanja podataka sa senzora, mikrokontroler pomoću algoritma i matematičkog modela robota računa brzine motora koje prima aktuatorski sustav. Na kraju, prijenos gibanja s aktuatorskog sustava na podlogu se ostvaruje preko kotača (postoje i drugi tipovi ostvarivanja prijenosa gibanja koje koriste hodajući mobilni roboti, dronovi, podmornice i sl., ali se u ovom radu neće razmatrati).

2.2. Konstrukcije kotača i pogon mobilnih robota

Kotači su najpopularniji izbor mehanizma gibanja mobilnog robota. Prednost tog načina gibanja je poprilično dobra ravnoteža zbog činjenice da su svi kotači u dodiru s površinom. Iako konstrukcija od tri kotača garantira postojanu ravnotežu, mobilni roboti s dva kotača također mogu biti stabilni. Ako se koristi sustav s više od tri kotača, poželjno je postojanje sustava suspenzije kako bi se osiguralo održavanje kontakta s površinom u slučaju susreta mobilnog robota s neravnom površinom. Iako ta mogućnost nije isključena, treba napomenuti da broj kotača ne bi trebao biti veći od četiri zbog povećanja kompleksnosti kinematičkog modela. Međutim, konstruktori mobilnih robota s kotačima se suočavaju s problemima kao što su mogućnost pružanja dovoljne količine trenja između kotača i što većeg raspona terena te mogućnost pružanja adekvatne kontrole nad brzinom robota.

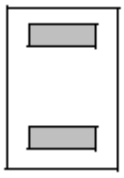
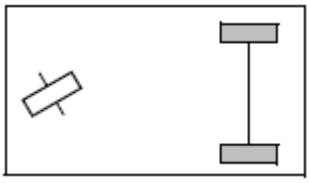
Postoje četiri osnovna tipa kotača koji su prikazani na slici 4. te moguće konfiguracije s obzirom na njihov broj koje su prikazane u tablici 1. To su:

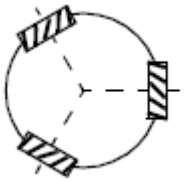
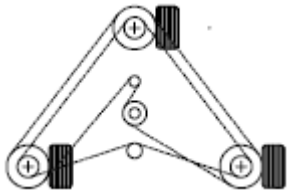
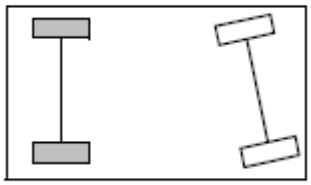
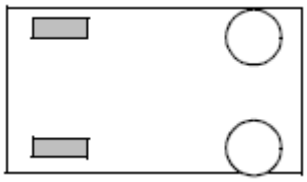
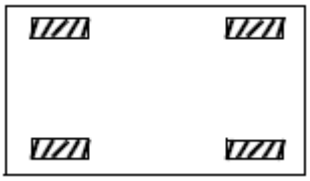
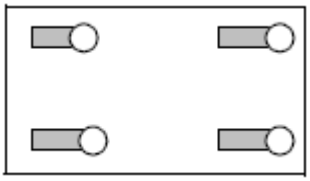
- *Fiksni kotač (a)*
- *Kutni rotirajući kotač (b)*
- *Švedski kotač (c)*
- *Sferni kotač (d)*



Slika 4. Osnovni tipovi kotača[4]

Tablica 1. Konfiguracije kotača s obzirom na njihov broj [4]

Broj kotača	**Prikaz konfiguracije**	**Opis**
2		Dva fiksna kotača
3		Dva povezana fiksna kotača s jednim slobodno rotirajućim kotačem straga

3		Tri švedska ili sferna kotača; moguća kretnja u svim smjerovima
		Tri sinkronizirana kotača; orijentacijom se ne može upravljati
4		Dva pogonska kotača straga i dva upravljiva kotača sprijeda
		Dva fiksna i dva sferna kotača
		4 švedska, svesmjerna kotača
		4 pogonska i upravljiva kutno rotirajuća kotača

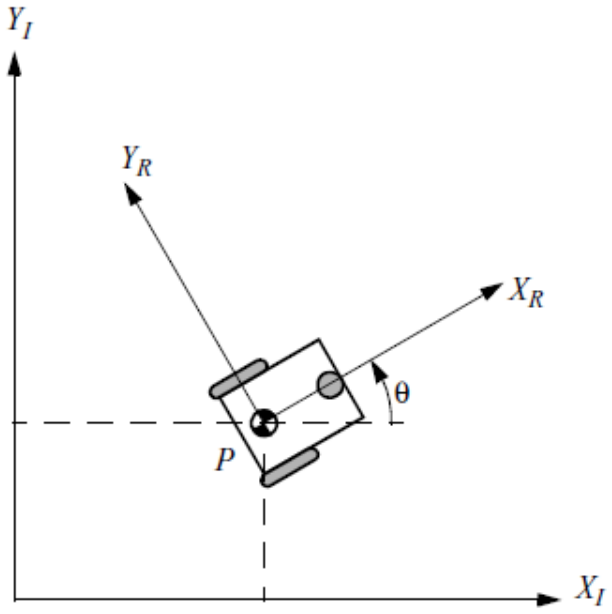
Tip, broj te konfiguracija kotača direktno utječu na kinematiku mobilnog robota. Stoga, s obzirom na kinematiku robota postoje četiri osnovna pogona mobilnih robota:

- Diferencijalni pogon *(engl. differential drive)*
- *Ackerman steering* pogon
- Svesmjerni pogon *(engl. omnidirectional drive)*
- Sinkronizirani pogon *(engl. synchro drive)*

Budući da je izvedba zadatka ovog rada zamišljena na diferencijalnom robotu, u nastavku će detaljno biti objašnjen kinematički model tog tipa.

2.3. Kinematika diferencijalnog mobilnog robota

Diferencijalnog mobilnog robota karakteriziraju dva fiksna kotača (gdje je moguće svakom pojedinačnom kotaču zadati kutnu brzinu okretanja) sa ili bez jednog pasivnog rotirajućeg kotača. Budući da se robot kreće po čvrstoj, ravnoj podlozi, njegov položaj se definira unutar X, Y ravnine te rotacijom oko vertikalne osi Z. Stoga, broj stupnjeva slobode je tri. Radi boljeg opisivanja položaja u prostoru uvodi se referentni (globalni) koordinatni sustav koji je fiksan, odnosno nepomično vezan za jednu točku u prostoru te robotov (lokalni) koordinatni sustav koji je vezan za robotovo centar rotacije, no njegovom kretnjom se pomiče unutar referentnog koordinatnog sustava. Prikaz lokalnog koordinatnog sustava diferencijalnog robota unutar globalnog koordinatnog sustava se nalazi na slici 5.



Slika 5. Mobilni robot u X-Y ravnini [4]

Iz slike 5. se može vidjeti da je globalni sustav označen s X_I i Y_I, a lokalni s X_r (definiran u smjeru kretanja robota) i Y_r. Budući da se stanje svakog robota opisuje njegovom pozicijom i orijentacijom unutar globalnog koordinatnog sustava te da je gibanje robota moguće u X,Y ravnini te rotacijom oko Z osi kako je prethodno spomenuto, to stanje se može zapisati kao vektor s tri elementa. Stanje je opisano koordinatama x i y koje predstavljaju udaljenost lokalnog i globalnog koordinatnog sustava te kutom θ koji predstavlja zakrenutost lokalnog u odnosu na globalni koordinatni sustav.

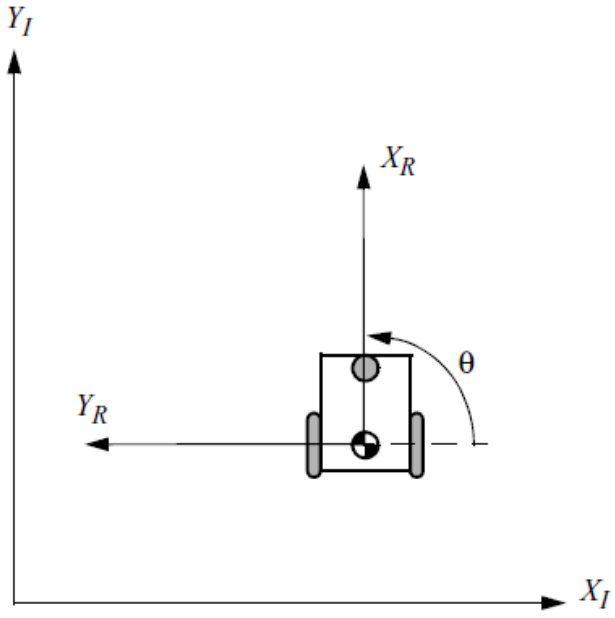
Zapis stanja mobilnog robota se nalazi u izrazu (1).

$$\boldsymbol{q} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{1}$$

Kako bi se robotov položaj mogao potpuno odrediti unutar referentnog koordinatnog sustava, potrebno je definirati transformacije jednog u odnosu na drugi. Zbog toga se uvodi matrica transformacija T vidljiva u zapisu (2).

$$T = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Slika 6. pokazuje usklađenost lokalnog s globalnim koordinatnim sustavom za kut $\theta = \frac{\pi}{2}$.



Slika 6. Usklađenost lokalnog s globalnim koordinatnim sustavom [4]

Pri kojoj matrica transformacija *T* izgleda ovako:

$$T(\frac{\pi}{2}) = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Budući da se svakom kotaču mogu zadati brzine pojedinačno, njihovim okretanjem u istom smjeru istom brzinom se robot giba pravocrtno. Ako se kotači gibaju istom brzinom, ali u suprotnom smjeru, robot rotira oko centra rotacije za kut θ. U skladu s promjenama brzina robota se mijenja i njegov položaj te je iste potrebno definirati. Translacijska brzina robota utječe na njegove translacijske pomake dok kutna brzina (izvedena pomoću enkodera na motoru) utječe na rotaciju oko osi Z. Obje su definirane u izrazima (4) i (5) kako su navedene.

$$v = \frac{v_r + v_l}{2} \tag{4}$$

$$\omega = \frac{r}{l}(\omega_r - \omega_l) \tag{5}$$

Gdje su:

- $v_r/v_l -$ translacijska brzina desnog/lijevog kotača
- $\omega_r/\omega_l -$ kutna brzina desnog/lijevog kotača
- $l -$ udaljenost kotača od centra rotacije
- r – polumjer kotača

Odnosi između brzina pojedinačnih kotača su definirani preko polumjera *r* te glase:

$$v_r = \omega_r r \tag{6}$$

$$v_l = \omega_l r \tag{7}$$

S obzirom na to da izrazi (4) i (5) definiraju brzinu robota u lokalnom koordinatnom sustavu, potrebno je primijeniti matricu transformacija T iz izraza (2) kako bi se dobila brzina robota u globalnom koordinatnom sustavu. Budući da je kutna brzina jednaka u oba koordinatna sustava, za rotacijski dio nije potrebno koristiti matricu transformacija. Stoga, izraz za translacijsku brzinu nakon primjene matrice *T* glasi:

$$\begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} v \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} v\ cos\theta \\ v\ sin\theta \\ 0 \end{bmatrix} \tag{8}$$

Na kraju, translacijska i kutna brzina u globalnom koordinatnom sustavu glase:

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\ cos\theta \\ v\ sin\theta \\ \omega \end{bmatrix} \tag{9}$$

3. ROS

Kako je navedeno u uvodu, ROS nije samostalan operacijski sustav. Stoga je potrebno odabrati temeljni operacijski sustav na kojem će biti instaliran. Budući da je Linux Ubuntu temeljna platforma za korištenje ROS-a, odabrana je verzija Ubuntu 20.04 koja će se pokretati preko *dual boot* opcije. Pokretanje Linux operacijskog sustava preko *dual boot* opcije odabrano je zato što ta opcija omogućuje pokretanje Linuxa na stvarnom hardveru te pruža bolju izvedbu od virtualne mašine. Nedostatak je fleksibilnost promjene operacijskog sustava, no to ne predstavlja problem zbog toga što se sve potrebno za rad nalazi na Linuxu.

ROS *(engl. Robot Operating System)* je *open-source* sustav koji pomaže istraživačima i developerima stvarati i iskorištavati gotove kodove za robotske aplikacije. Budući da je *open source*, dostupan je svima te omogućuje suradnju među ljudima te poboljšanje same robotike. Sama činjenica da ga tvrtke poput ABB-a, OTTO Motors-a i mnogih drugih danas otvoreno koriste dokazuje da ROS pokreće budućnost robotike u industriji.

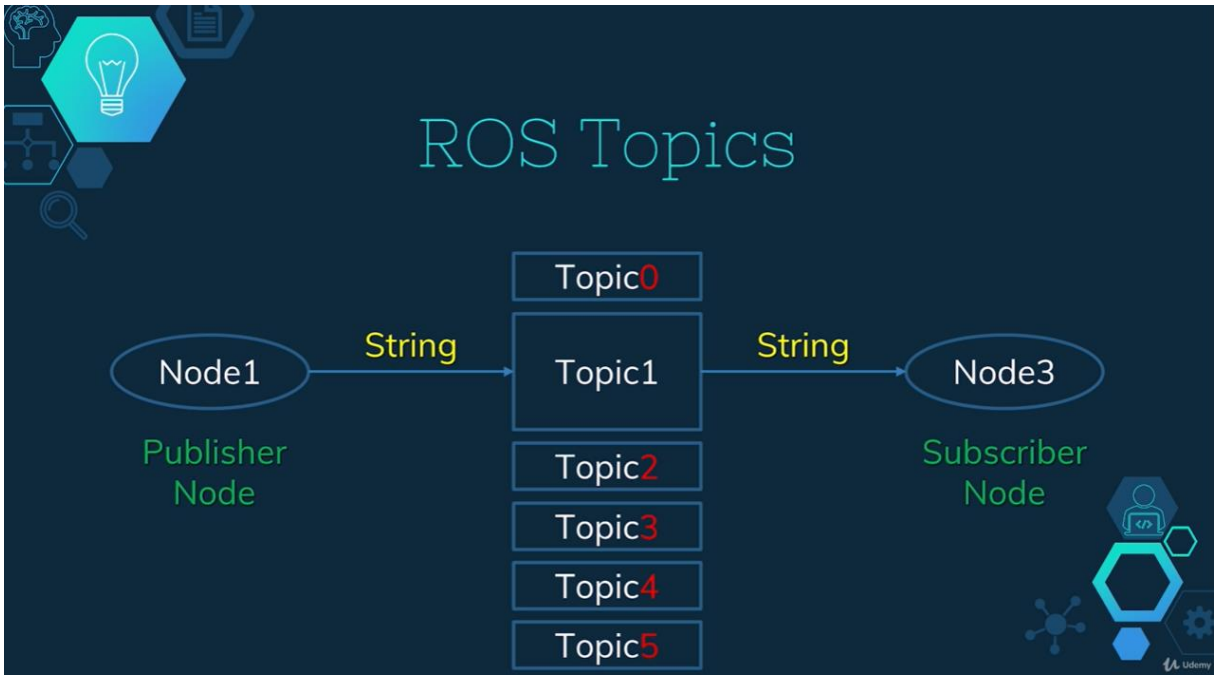
Glavni razlozi za korištenje ROS-a su:

- Sprječava "ponovni izum kotača". Budući da je građen sa suradnjom među ljudima na umu, osnovna znanja i kodovi su dostupni svima i mogu se primijeniti na sve robotske platforme. Sadrži pakete za sve, od izračuna trajektorije robota, vladanja SLAM algoritmima do implementacije daljinskog upravljanja.
- Omogućuje komunikaciju s robotima pomoću Python i C++ jezika, dok sadrži knjižnice koje omogućuju i korištenje većine drugih jezika.
- Svakodnevni razvitak i održavanje. ROS zajednica postoji i raste od 2007. godine te pokazuje sve naznake da će njegova primjena opstati te s godinama postajati još bolja.
- Omogućuje simulaciju robota u bilo kojem okruženju prije primjene u stvarnom svijetu pomoću alata kao što je Gazebo.

U ovom radu će biti korištena najnovija verzija ROS-a, ROS Noetic. Ta verzija ima oznaku *Long Term Support* što znači da je njeno održavanje predviđeno za 5 godina (od svibnja 2020. do svibnja 2025.). Osmišljena je za rad na Ubuntu 20.04 operacijskom sustavu koji je prethodno instaliran te je ujedno i posljednje i konačno službeno izdanje ROS-a 1. U budućnosti, sav rad i održavanje će biti usmjereno u razvitak ROS-a 2 koji je glavna obrada ROS-a 1.

3.1. Kako ROS funkcionira?

Glavna ideja rada u ROS-u je postojanje komunikacije između *node-a* (izvršna jedinica na ROS mreži), odnosno slanje i primanje informacija. Komunikacija uključuje prvi *node* koji zapisuje poruku, odnosno daje informaciju na *topic* koji se ponaša kao tema razgovora te drugi node koji čita tu poruku. U ROS terminologiji, process zapisivanja i čitanja informacija se zove *publishing* i *subscribing,* dok se node nazivaju *publisher* i *subscriber.* Nadalje, u mnogim procesima može postojati više *topica*, no komunikacija između *node-a* će se odvijati samo ako su povezani na isti *topic*. Također, komunikacija između *node-a* se mora odvijati pomoću istog tipa poruke. Na primjer, ako *publisher* objavljuje *string* tip poruke na *topic*, *subscriber* također mora čitati samo taj tip poruke. Međutim, samo postojanje *publisher-a*, *subscriber-a* i *topic-a* nije dovoljno. Budući da se u sustavu mogu odvijati radnje više *node-a* odjednom i kako bi se znalo koje node su pokrenute te kako se komunikacija mora odvijati, u pozadini mora biti pokrenut ROS Master koji pruža tu značajku. ROS Master prati sve *node* pokrenute u sustavu te tako omogućuje da određena node-a pratiti prisutnost druge. ROS Master se pokreće u terminalu pomoću naredbe *roscore.*

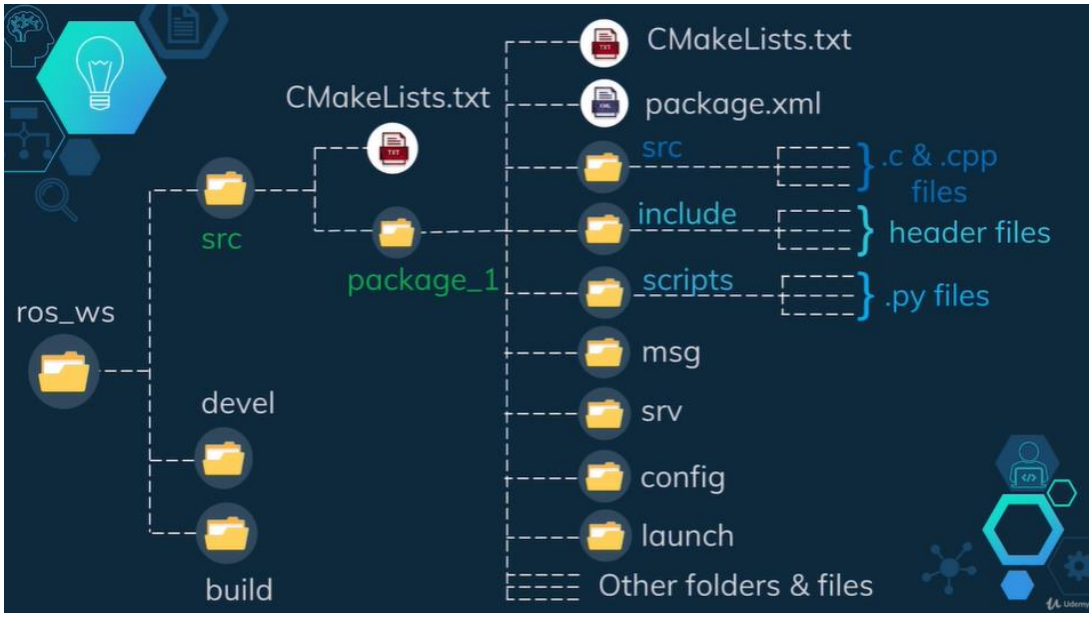


Slika 7. Komunikacija unutar ROS-a [7]

Međutim, kako bi se gotovi paketi preuzeti s interneta ili paketi koje korisnik gradi mogli koristiti, te kako bi komunikacija između *node-a* koje se nalaze unutar paketa bila moguća, mora se postaviti ROS radno područje *(engl. workspace)*. Paketi su osnovna jedinica organizacije unutar ROS-a te osim *node-a* mogu sadržavati knjižnice, konfiguracijske datoteke i sl. Detaljan opis povezanosti ROS radnog područja s drugim područjima vezanim uz ROS može se pronaći na službenoj stranici ROS-a. Sistematizacija unutar radnog područja objašnjena je u nastavku.

3.1.1. Postavljanje radnog područja

Prvi korak prema korištenju ROS-a je njegova instalacija čija se detaljna uputa može pronaći na službenoj stranici. Kako je navedeno u uvodnom dijelu trećeg poglavlja, odabrana verzija je ROS *Noetic*. Nakon instalacije, potrebno je postaviti radno područje prema u kojem će se nalaziti svi paketi potrebni za rad. Sistematizacija unutar radnog područja je sljedeća. Radno područje unutar sebe sadrži tri direktorija: *build*, *devel* i *src*. *Build* i *devel* se stvaraju automatski dok *src* stvara korisnik. Unutar *src* direktorija se nalaze svi izvorni kodovi/paketi koje korisnik piše/gradi ili preuzima s interneta. U *build* direktoriju se nalaze datoteke koje generira naredba *cmake*, a u *devel* direktoriju se nalaze kompilirane datoteke stvorene nakon izvršavanja naredbe *catkin_make* koju je potrebno izvršiti u terminalu nakon svakog stvaranja ili preuzimanja paketa. Detaljna sistematizacija izgleda ovako:



Slika 8. Sistematizacija unutar ROS radnog područja [7]

3.2. Simulacijski i vizualizacijski alati ROS-a

3.2.1. Gazebo

Gazebo je samostalan 3D simulator kojem je u integraciji s ROS-om glavni cilj simulirati korisnikovog robota te prikazati njegovo približno ponašanje u stvarnom svijetu. Omogućuje realističan prikaz samog okruženja uključujući visokokvalitetno osvjetljenje, sjene i teksture. U sebi ima integrirane mašine poput ODE-a (*engl. Open Dynamics Engine*), integrira OpenGL *renderiranje* te podržava kodove za simulaciju senzora i regulaciju aktuatora.

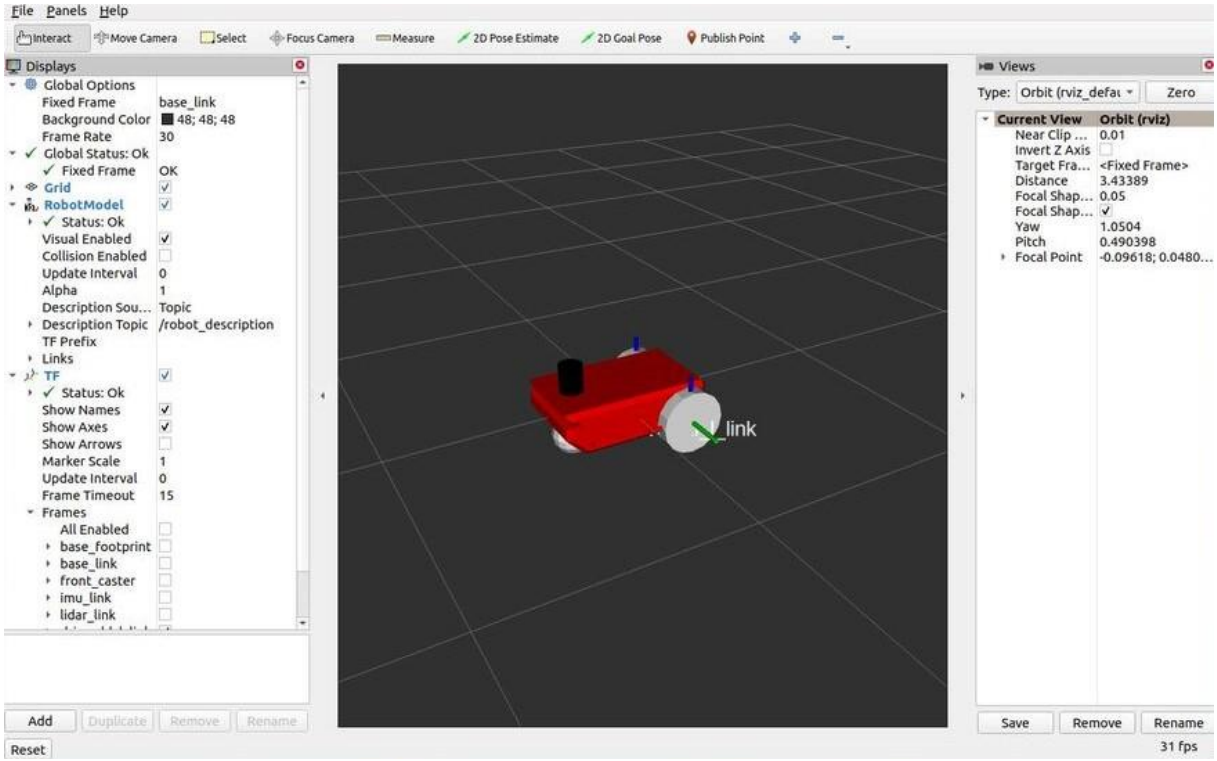
Integraciju s ROS-om postiže pomoću skupa paketa "*gazebo_ros_pkgs"* koji pružaju potrebna sučelja za simulaciju robota u Gazebo-u koristeći ROS poruke i određene značajke samih paketa. Ako se provela "Desktop-Full" instalacija ROS-a prema uputama na internetu, simulator i paketi se automatski instaliraju. Inače, instalaciju i preuzimanje paketa treba provesti zasebno.

Slika 9. Izgled Gazebo 3D simulatora [8]

3.2.2. RVIZ

RVIZ je ROS-ovo grafičko sučelje (*node*) koje omogućuje vizualizaciju informacija koje se dobivaju s robota pomoću *subscribe-a* na aktivne *topice*. Koristi se za vizualizaciju procesa poput lokalizacije, navigacije i mapiranja. Ako se provela "Desktop" instalacija ROS-a prema uputama na internetu, RVIZ se automatski preuzima. Inače, preuzimanje treba provesti zasebno.

S lijeve strane sučelja slike 10. se nalaze segmenti koji se mogu dodati u RVIZ, a najbitniji za ovaj rad su *TF* (vizualizira transformacije između koordinatnih sustava) i *RobotModel* (prikazuje robota). Bitna globalna postavka "Fixed Frame" definira glavni koordinatni sustav s kojim se prikazuju ostali generirani koordinatni sustavi pomoću čvora *tf* (čvor za primjenu matrica transformacija).



Slika 10. Prikaz RVIZ sučelja [9]

Na kraju, jednostavno se može reći da RVIZ prikazuje informacije te pokazuje što robot *"misli"* da se događa nakon primanja informacija s robota u Gazebo-u ili u stvarnoj situaciji dok Gazebo prikazuje što se zapravo događa.

4. KVADRATNA TRAJEKTORIJA

UMBMark (detaljno objašnjena kasnije) je standardna metoda za ispitivanje točnosti mobilnih robota i njihovu kalibraciju te se primjenjuje na kvadratnu putanju dimenzija 4x4 m. Stoga, prvi korak pri osiguravanju što točnije estimacije odometrije robota je zadavanje navedene referentne trajektorije. Kako bi se to provelo, potrebno je napraviti čvor u ROS-u za generiranje i slanje iste mobilnom robotu. Nadalje, zbog ograničenosti prostora u kojem se fizička testiranja odvijaju, putanja je smanjena na 1x1 m.

4.1. Vizualizacija u RVIZ-u

Kao što je navedeno, RVIZ grafičko sučelje omogućuje vizualizaciju informacija koje se dobivaju s robota pomoću *subscribe-a* na aktivne *topice*. U slučaju ovog rada informacije koje će biti vizualizirane su robotovi koordinatni sustavi; transformacije u odnosu na *"Fixed Frame" – odom (predstavlja početni položaj robota)* koji se automatski stvaraju povezivanjem robota s RVIZ-om; zasebno generirani koordinatni sustav (objašnjen u nastavku) te robotove transformaciju u odnosu na njega.

4.2. Istraživački mobilni robot Regionalnog centra izvrsnosti za robotske tehnologije

Prije nego se počne objašnjavati proces pisanja koda, potrebno je navesti robota namijenjenog za primjenu istog. Robot je diferencijalnog pogona te posjeduje dva usporedno pogonjena i upravljiva kotača koji omogućuju kretanje te dva slobodno rotirajuća dodanih radi stabilnosti i osiguravanja kretanja po ravnoj podlozi. Karakteristike robota su prikazane u tablici 2.

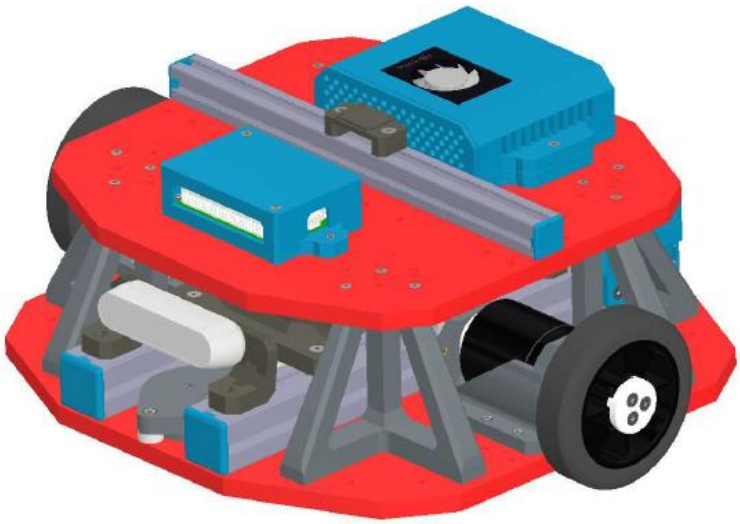
Tablica 2. Karakteristike mobilnog robota

Broj pogonskih kotača	2
Vrsta pogonskih kotača	Ispunjena guma
Promjer pogonskih kotača [mm]	95
Širina pogonskih kotača [mm]	24
Broj slobodno rotirajućih kotača	2
Vrsta slobodno rotirajućih kotača	Aluminijska kuglica
Promjer slobodno rotirajućih kotača [mm]	19
Prijenosni omjer	1:1
Broj enkodera	2
Međusobni razmak kotača [mm]	365

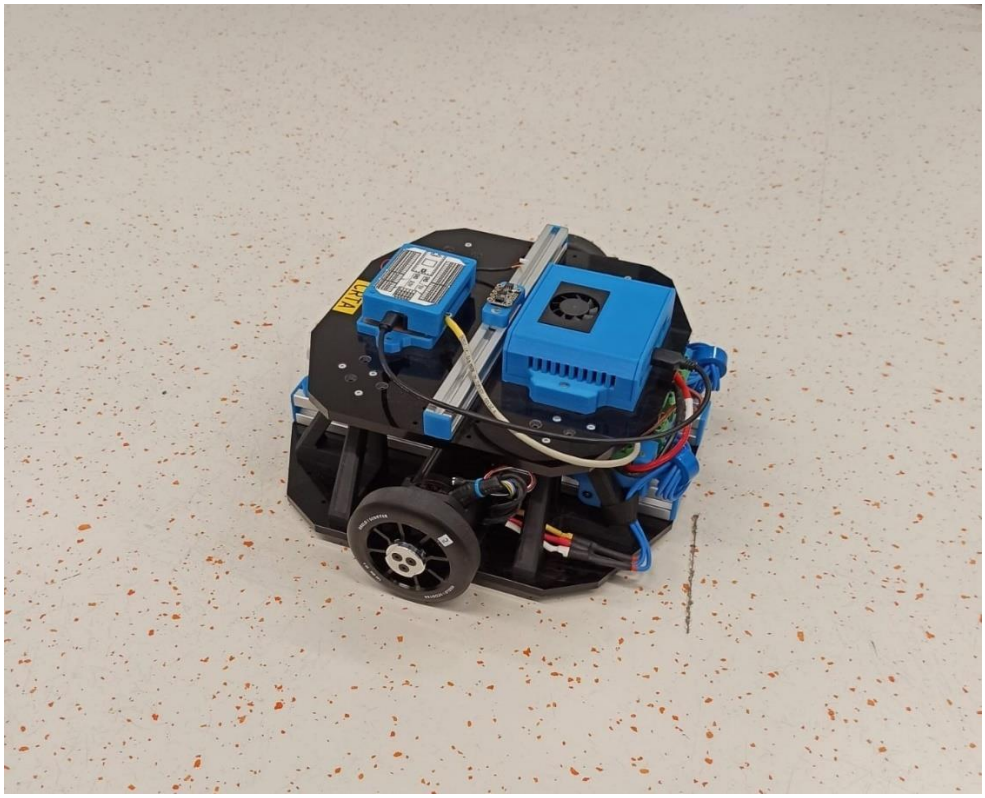
Kotači se pogone pomoću dva motora bez četkica kompanije *Alien Power System* koji pojedinačno postižu moment od 8.5 Nm pri 33.6 V. Također, za određivanje pozicije se koriste *CUI AMT10E2-V* inkrementalni enkoderi. Gustoća im je 20480 tikova po okretaju te su postavljeni na slobodni kraj osovine pogonskih motora.

Mobilni robot posjeduje *NVIDIA Jetson Xavier NX* mikroprocesorsko računalo za provođenje *high level* upravljanja koje posjeduje Linux *Ubuntu 18.04 – Bionic Beaver* te je na njega instaliran ROS *Melodic Morenia.* Algoritmi izračuna referenci položaja i potrebne brzine pogonskih kotača koje se prosljeđuju *low level* kontroleru se odvijaju na istom. Kako je spomenuto, robot posjeduje i mikrokontroler *Teensy 4.0* zaduženog za rad s aktuatorima te se na njemu provodi *low level* upravljanje, odnosno upravljanje motorima i očitavanja trenutnog položaja s enkodera. Povezivanje s mikroprocesorom *Jetson* je ostvareno pomoću USB *port-a* i serijske komunikacije. Također, robot posjeduje i *Odrive* uređaj namijenjen za precizno upravljanje motorima bez četkica pomoću petlje povratne veze. Informacija o trenutnom položaju ili brzini očitanim pomoću enkodera je ulazni parametar algoritma *Odrive-a*, dok se reference prema kojima se zakreću motori dobivaju kao izlazni podaci. Komunikacija s *high level* upravljačkim sustavom se provodi korištenjem *UART* protokola (*engl. universal asynchronous receiver/transmitter).*

Povezivanje s robotom se omogućuje pomoću WiFi-ja u ROS Master konfiguraciji. Na kraju, mobilni robot prima informacije o brzini preko *cmd_vel topic-a* (objašnjeno u nastavku). Prikaz njegovog CAD modela i fizičkog izgleda se nalazi na slikama 11. i 12.



Slika 11. CAD model mobilnog robota



Slika 12. Mobilni robot

Kako bi se omogućila komunikacija s robotom preko WiFi-ja u ROS Master konfiguraciji, potrebno je urediti *.bashrc* file kako bi se ROS na robotu povezao s radnim prostorom na osobnom računalu. Važno je napomenuti da osobno računalo i mobilni robot moraju biti povezani na istu mrežu. Upisane linije u *.bashrc* datoteku su:

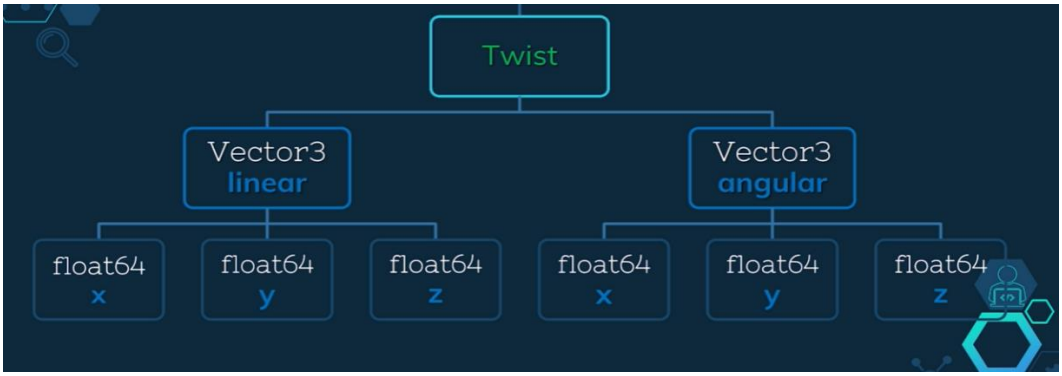
1. export ROS_MASTER_URI = http://192.168.1.78:11311
2. export ROS_HOSTNAME = 192.168.1.26
3. export ROS_IP = 192.168.1.26
4. echo "ROS_HOSTNAME: "$ROS_HOSTNAME
5. echo "ROS_IP: "$ROS_IP
6. echo "ROS_MASTER_URI: "$ROS_MASTER_URI

4.3. Čvor za generiranje i slanje putanje mobilnom robotu

Budući da je čvor pisan u Python programskom jeziku, potrebno je napraviti *.py* datoteku u ROS radnom prostoru prema shemi objašnjenoj u "*Postavljanje radnog područja"* te *importati*

sve zadane pakete. Funkcija čvora je *"slanje"* kvadratne putanje (informacije o kretanju) mobilnom robotu. Stoga, čvor vrši funkciju *publisher-a* dok mobilni robot vrši funkciju *subscriber-a.* Nadalje, kako bi komunikacija bila moguća, poslana informacija mora biti spremljena na *topic*. *Cmd_vel* je *topic* koji služi za razmjenu informacija o brzinama koje robot mora ostvariti te se preko njega ostvaruje kretanje robota. Tip poruke koji se zapisuje i čita s *cmd_vel-a* je *Twist.*

Definicija *Twist-a* izgleda ovako:



Slika 13. Twist tip poruke [7]

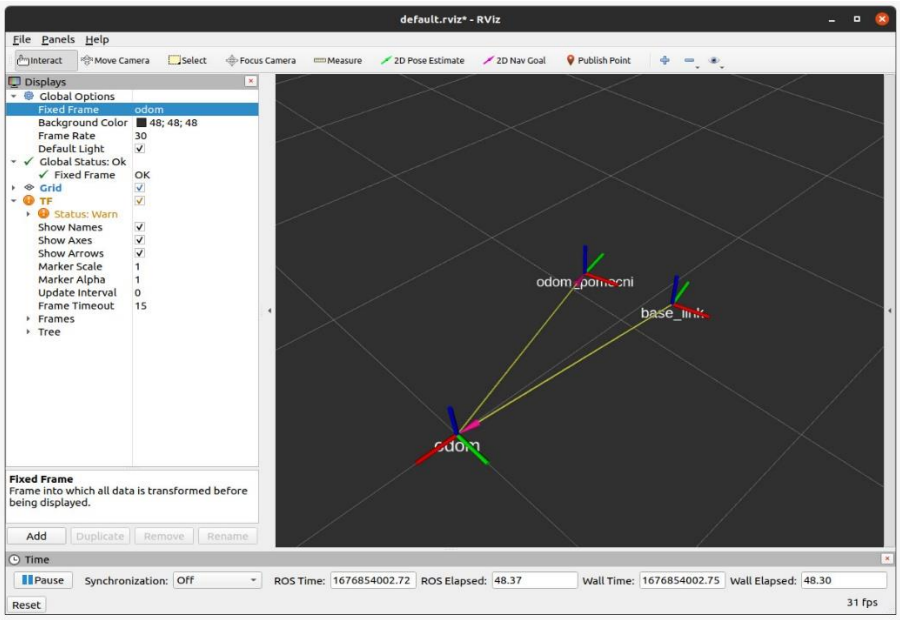
Stoga, ako se želi ostvariti pravocrtno gibanje robota u smjeru osi x, naredba koja će to omogućiti je *twist.linear.x = <željena brzina u m/s>*.

U nastavku će se objašnjavati *"tok misli"* prilikom izrade čvora za generiranje i slanje putanje robotu. Cijeli kod se može pronaći u prilogu koji se nalazi na kraju ovog rada.

Samo zadavanje brzine nije dovoljno kako bi se ostvarila željena kvadratna putanja. Korak od velike važnosti je odabir koordinatnog sustava na temelju kojeg će robot pomoću matrica transformacija, odnosno *tf* čvora određivati svoj položaj, odnosno udaljenost i rotaciju od istog. Načelno, tome može poslužiti *Fixed Frame - odom.* Međutim, UMBMark metoda ispitivanja točnosti i kalibracije nalaže da se provođenje putanje mora odviti pet puta u smjeru kazaljke na satu te pet puta u smjeru obrnutom od kazaljke na satu. Problem koji se javlja ako se koristi *Fixed Frame – odom* je sljedeći.

Kada se fizičkom robotu zada kvadratna putanja, zbog sustavnih i nesustavnih grešaka koje će biti objašnjene u poglavlju *UMBMark kalibracija mobilnog robota,* njegova konačna pozicija neće biti jednaka početnoj, odnosno lokaciji *Fixed Frame-a – odom* (koji je s vremenom uvijek na istom mjestu). Budući da je kretanje robota temeljeno na udaljenosti od referentnog koordinatnog sustava, njegov pomak u idućem krugu će biti veći, odnosno manji od 1 m ovisno

o lokaciji gdje je robot završio prethodni krug. Stoga, potrebno je napraviti čvor koji će u težištu robota generirati koordinatni sustav prije početka njegovog kretanja te koji će se resetirati nakon svakog kruga i ponovno pojaviti u robotovom težištu kako bi robot mogao ponovno započeti kretanje s odgovarajućom duljinom kraka. Navedeno je ostvareno u Python skripti *odom_pomocni.py* koja se pokreće preko glavne skripte *kvadrat2.py* (u kojoj se nalaze sve ostale funkcije) pomoću funkcije *Odom_pomocni()* te je također prikazana u prilogu. Prije nego se objasni način na koji je sustav generiran, važno je napomenuti da robotov glavni i jedini koordinatni sustav koji se koristi je *base_link* zato što se nalazi u samom težištu robota.



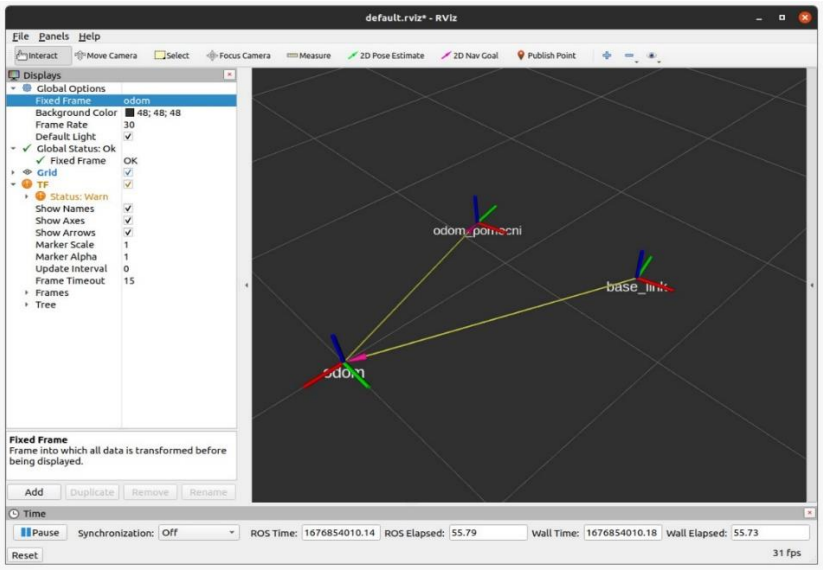
Slika 14. Generirani koordinatni sustav odom_pomocni

Način na koji se *odom_pomocni* generira je sljedeći. U funkciji *odom_pomocni()* skripte *odom_pomocni.py* se dobivaju informacije o trenutnoj transformaciji između robotovog koordinatnog sustava *base_link* i fiksiranog sustava *odom*. Nakon toga, na trenutnoj statičnoj poziciji *base_link-a* u odnosu na *odom* se stvara novi koordinatni sustav *odom_pomocni* te se tamo zadržava do terminacije spomenute skripte.

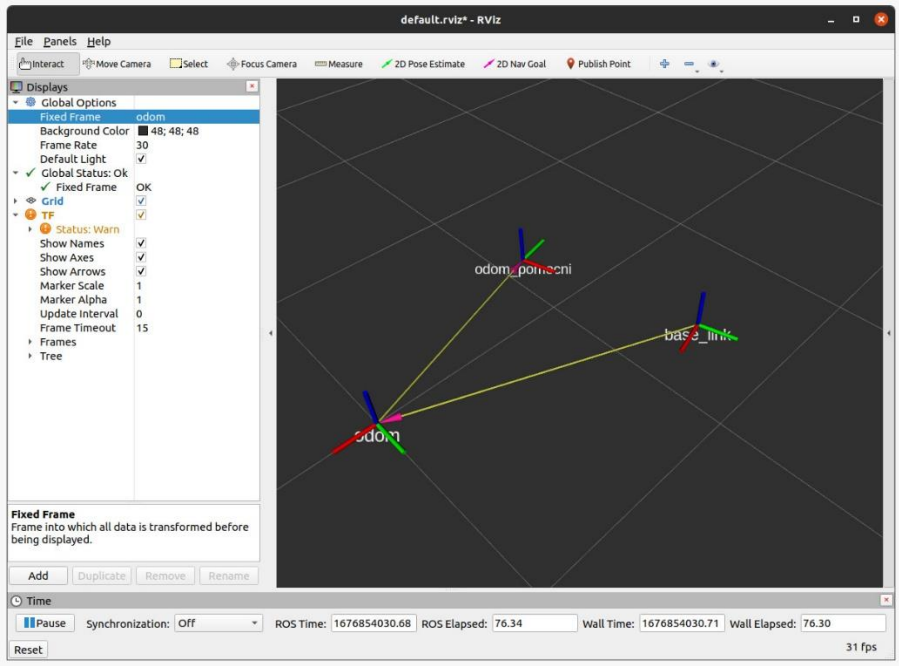
Budući da je koordinatni sustav u odnosu na kojeg se robot udaljava određen, potrebno je napraviti funkciju koja mjeri udaljenost od početne pozicije i trenutne pozicije robota te je to ostvareno pomoću funkcije *getABSDistance()*. Funkcija koja vraća trenutne pozicije robota u odnosu na referentni koordinatni sustav, odnosno matricu transformacija između *base_link-a* i *odoma_pomocnog* je *getCurrentPosition()*.

Napokon, potrebni segmenti za slanje informacija o brzinama su određeni. Navedeno se ostvaruje funkcijama *straight() i turn()* koje su ujedno i regulatori brzine. Logika iza tih funkcija je sljedeća. U funkciji *straight()* se pozivaju pozicije iz funkcije *getCurrentPosition()* te funkcija za računanje udaljenosti između početne i trenutne pozicije *getABSDistance()*. Također, funkcija *straight()* sadrži parametar *dx* koji predstavlja duljinu stranice kvadratne putanje od 1 m. Definira se *publisher* te se postavljaju uvjeti. Ako je udaljenost između koordinatnih sustava manja od razlike parametra *dx* i prethodno definiranog parametra preciznosti, na *cmd_vel Topic* se šalje informacija o pravocrtnoj brzini u smjeru osi x robota. Brzina je definirana funkcijom tangensa hiperbolnog kako bi se povećanjem razlike između sustava brzina smanjivala. Drugi uvjet je: ako je razlika u udaljenosti veća od *dx – DIST_PRECISION – 0.02,* pravocrtno gibanje se drastično smanje te se robot kreće tom brzinom sve dok razlika parametra *dx* i udaljenosti nije manja od željene preciznosti. Kada se zadnji uvjet postigne poziva se funkcija *stopAllMotors()* koja sve pravocrtne i kutna brzine izjednačava s nulom te zaustavlja robota nakon obavljenog puta od 1 m. Tada se pokreće funkcija *turn()* koja sadrži parametre *a , b i c.* Parametar *a* predstavlja kut zakreta, parametar *b* definira kutnu brzinu oko osi *z* čiji je predznak za različite smjerove putanje drugačiji dok parametar *c* vrši istu funkciju samo s drastično manjom brzinom pri zatvaranja kuta od 90°. Dok je razlika između zadanog kuta i trenutnog kuta zakreta veća od željene preciznosti, na *cmd_vel* se šalje informacija o brzini oko osi *z* nakon čega se poziva funkcija *stopAllMotors()* te se robot zaustavlja.

Nakon pozivanja funkcija *straight() i turn()* u *main-u* četiri puta respektabilno, robot je obišao kvadratnu putanju.

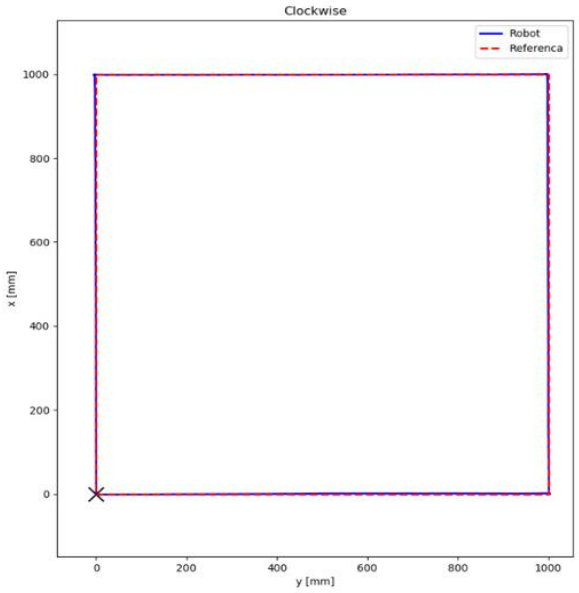


Slika 15. Prijeđen put od 1 m u odnosu na *odom_pomocni*



Slika 16. Kut zakreta od 90° u odnosu na *odom_pomocni*

Nakon robotovog obilaženja kvadratne putanje 1x1 m te spremanjem podataka transformacija koordinatnih sustava između robotovog *base_link-a* i *odoma_pomocnog* u *.txt* file, pomoću funkcije *Plotting()* je moguće grafički prikazati rezultate kretanja u odnosu na referentu kvadratnu trajektoriju.



Slika 17. Prikaz robotove kvadratne putanje u odnosu na referencu

Kao što se može vidjeti iz slike 17., čvor za generiranje i slanje kvadratne putanje mobilnom robotu radi ispravno. Međutim, promatranjem robota se može uočiti kako robot kvadratnu putanju, zbog sustavnih i nesustavnih grešaka, ne obiđe kako je prikazano.

5. MJERENJE ODSTUPANJA

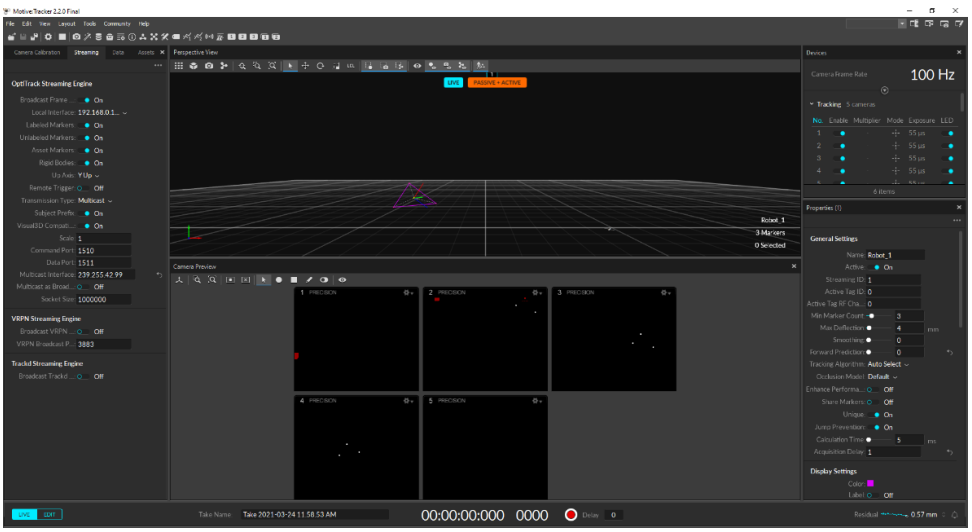
5.1. OptiTrack

Budući da je robotovo odstupanje od referentne kvadratne putanje primjetno, odstupanja je potrebno mjeriti kako bi se na njih moglo utjecati. Za praćenje robotovog kretanja i mjerenje odstupanja odabran je OptiTrack.

OptiTrack je vanjski vizijski sustav koji se sastoji od softvera za upravljanje sustava za snimanje pokreta te kamera velikih brzina za praćenje markera. Ne proizvodi pozicijske greške veće od 0.3 mm i rotacijske greške od 0.05° te je odličan izbor za praćenje industrijskih i mobilnih robota zbog niske latentnosti i visoke preciznosti praćenja unutar šest stupnjeva slobode. Kako bi se vizijski sustav mogao koristiti, potrebno je prvo postaviti radno područje, zatim instalirati softver Motive, izvršiti kalibraciju sustava te ga povezati s ROS-om.

5.1.1. Motive

Motive je softver koji služi za upravljanje sustava za snimanje pokreta. Uz pružanje mogućnosti kalibracije i konfiguracije sustava, Motive također pruža korisnička sučelja za snimanje i obradu 3D podataka koji također mogu biti prenošeni uživo. Motive dobiva 3D informacije putem *rekonstrukcije*, procesa kompiliranja većeg broja 2D slika markera za dobivanje 3D koordinata. Koristeći 3D koordinate praćenih markera, Motive prikuplja podatke šest stupnjeva slobode (3D pozicija i orijentacija) krutih tijela te omogućava praćenje kompleksnih kretnji u 3D prostoru.



Slika 18. Motive sučelje [10]

5.1.2. Hardver i postavljanje radnog područja

U postavljanju radnog područja za snimanje pokreta uključeno je osam kamera te ostale komponente čija ispravna postava ima značajan utjecaj na kvalitetu prikupljenih podataka. Kamere koje se koriste u sustavu su *OptiTrack Prime^{X}13* te je njihova specifikacija navedena u nastavku.

Slika 19. OptiTrack PrimeX13 kamera [10]

Tablica 3. Svojstva PrimeX 13 kamere [10]

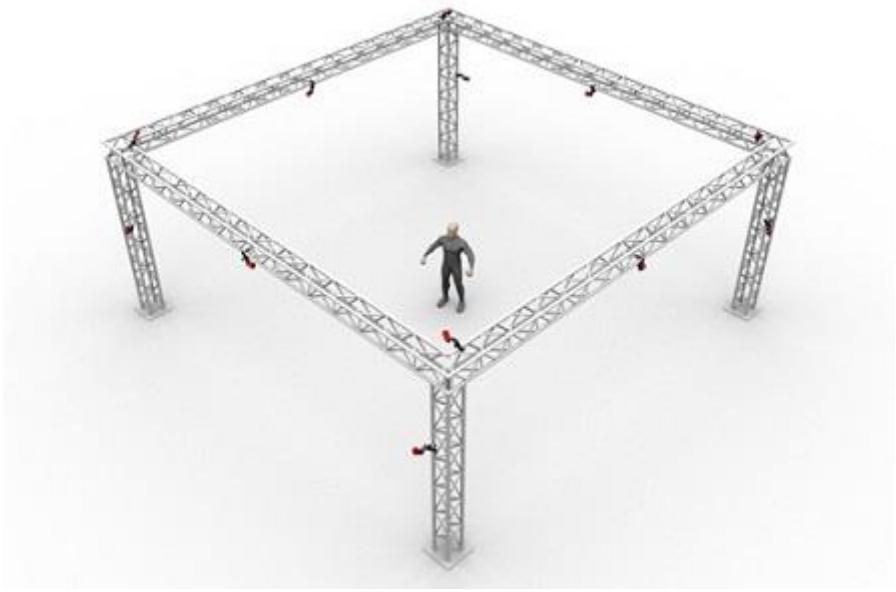
Senzor	1280x1024 *Rezolucija*	240 Hz *Frame rate*	4.2 ms *Latentnost*
Svojstva i domet praćenja	+/- 0.20 mm *Preciznost*	16 m *Pasivni markeri*	25 m *Aktivni markeri*
Kompatibilni softveri	Motive	Camera SDK	NatNet SDK
Obrada slike	Raw Grayscale	MJPEG Grayscale	Segment objekta
Globalni zatvarač	0.25 ms *Zadana brzina*	0.01 ms *Minimalna brzina*	3.9 ms pri 240 fps-a *Maksimalna brzina*
Povezivanje	GigE/PoE data port, Ethernet camera sync		
Leće i filter	Prilagodljiv fokus	850 nm *band-pass* filter	800 nm (IR)/ 700 nm (vidljivi) prekidač filter
Veličina i težina	6.86x5.46x2.7 cm		0.32 kg

Pri izračunu 3D lokacije markera, praćeni markeri moraju biti istovremeno snimani s barem dvije sinkronizirane kamere u sustavu. Ako se snimanje 2D pozicije obavlja s nedovoljnim brojem kamera, marker neće biti prisutan u prikupljenim podacima. Prema tome, snimljena trajektorija markera će sadržavati praznine te će preciznost snimke biti smanjena. Stoga, vidljivost markera tijekom cijelog snimanja je bitan faktor koji utječe na kvalitetu praćenja te bi kamere trebale snimati s različitih položaja kako bi se okluzije markera svele na minimum.

Preporuke prilikom postavljanja kamera i određivanja radnog područja su:

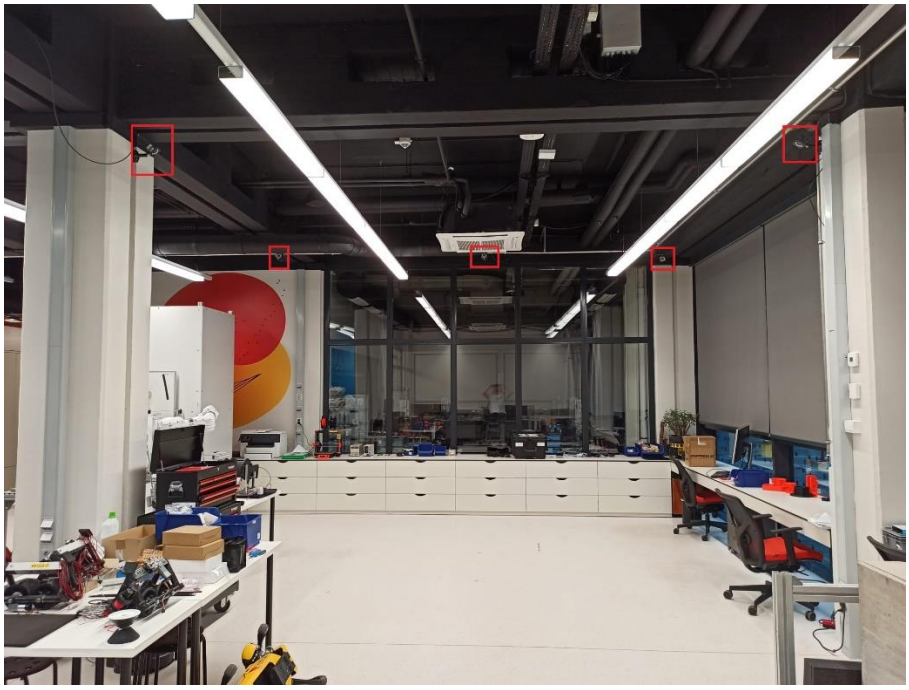
- *Radno područje.* Pri praćenju krutog tijela te prikupljanju podataka o njegovoj poziciji i orijentaciji, preporuča se postavljanje kamera na periferijama radnog područja unutar kojeg će predmet biti praćen kako bi se predmet mogao pratiti s prednje i stražnje strane.
- *Visina kamera.* Preporuča se postavljanje kamera na visoke uzvisine jer se time maksimizira pokrivenost cijelog volumena predmeta. Također, ako su kamere postavljene na niske uzvisine i usmjere jedne prema drugoj, sinkronizirana IR zračenja obje kamere će biti detektirana te će se morati maskirati. Međutim, postavljanjem kamera na različite uzvisine doprinosi pokrivenosti iz različitih kuteva gledanja.
- *Udaljenost kamere od kamere.* Odvajanjem kamera konstantnom udaljenosti te izbjegavanjem bliskog postavljanja se zaobilazi snimanje sličnih slika praćenog objekta i povećanje računalnog opterećenja prilikom kalibracije. Također, bliska i neujednačena postava kamera povećava šanse okluzija markera pri uvođenju prepreka u okoliš.
- *Udaljenost kamera od objekta.* Udaljenost ovisi o svrsi snimanja. Veća udaljenost kamere od predmeta osigurava veću pokrivenost. Međutim, snimanje na malim udaljenostima će omogućiti preciznija mjerenja.
- *Tlo.* Treba izbjegavati reflektirajuće podloge. IR zrake kamera se od njih mogu reflektirati te ometati praćenje. Također, fleksibilne i deformabilne podloge nisu preporučljive.
- *Smanjenje IR smetnji.* Kamere za snimanje pokreta prate markere pomoću otkrivanja reflektirane IR zrake i svako vanjsko IR svjetlo koje se nalazi unutar radnog prostora može ometati praćenje.
- *Prepreke.* Sve nepotrebne prepreke se trebaju ukloniti budući da mogu blokirati pogled kamere prema markerima.

Slika 20. Prikazuje preporučenu konfiguraciju radnog prostora te postavu kamera.



Slika 20. Preporučena konfiguracija prostora i kamera [10]

Nakon praćenja danih uputa, radni prostor unutar kojeg će se snimati odstupanje robota od zadane kvadratne trajektorije izgleda ovako:



Slika 21. Stvarna konfiguracija prostora i kamera (odostraga 3 kamere kao na kraju prostora)

Nakon ožičavanja i povezivanja komponenti te instalacije softvera Motive na osobno računalo koje je detaljno opisano u [10], prelazi se na kalibraciju sustava.

5.1.3. Kalibracija sustava

Tijekom kalibracije, sustav izračunava poziciju i orijentaciju svake kamere te količinu izobličenja u snimljenim slikama, a oni se koriste za konstruiranje 3D prostora snimanja u Motive-u. To se postiže promatranjem 2D slika s više sinkroniziranih kamera te pridruživanjem pozicije poznatih kalibracijskih markera sa svake kamere putem triangulacije. Treba imati na umu da ako se postava kamera promijeni tijekom snimanja, proces kalibracije se treba ponoviti. Također, kalibracijska preciznost tijekom vremena može opasti zbog raznih faktora radnog okruženja poput promjene temperature. Stoga se preporuča povremena rekalibracija sustava.

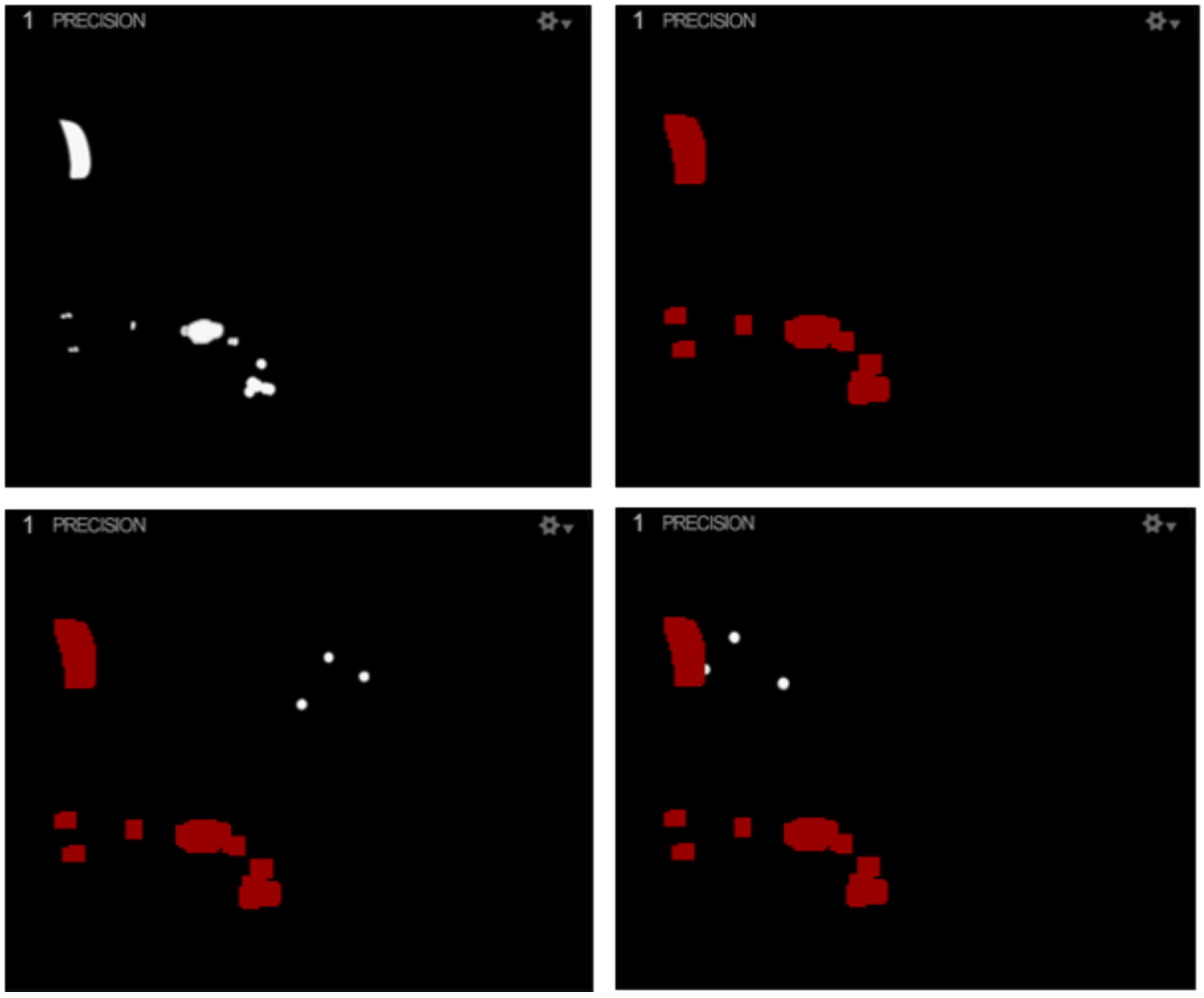
Osnovni koraci pri kalibraciji su:

- Priprema radnog prostora u kojem će se objekt snimati. (objašnjeno u 5.1.2.)
- Maskiranje postojećih refleksija IR zraka u pogledu kamere.
- Prikupljanje kalibracijskih uzoraka pomoću kalibracijskog štapa.
- Pregled rezultata i primjena kalibracije
- Postavljanje ravnine tla kako bi se kalibracija završila.

5.1.3.1. Maskiranje

U idealnom slučaju, sve refleksije i nepotrebni markeri su uklonjeni prije kalibracije. U slučajevima gdje to nije moguće, njihovo prekrivanje se ostvaruje pomoću *MaskingTool-a.* Taj alat primjenjuje crvene maske preko detektiranih refleksija u 2D pogledu kamere nakon čega su svi pikseli u maskiranim područjima ignorirani. *MaskVisible* značajka automatski detektira sve postojeće refleksije te provodi proces maskiranja. Međutim, maskiranje se može obaviti i ručno označavanjem kvadratnih područja klikom na predviđenu tipku.

Kao što je već spomenuto, maskirani pikseli su poptupno ignorirani te podaci u maskiranim područjima neće biti prikupljeni prilikom izračuna 3D podataka. Stoga, pretjerana upotreba maskiranja rezultira gubljenjem podataka i učestaloj okluziji markera. Upravo iz tog razloga je bitno utjecati na sve faktore refleksije prije kalibracije.

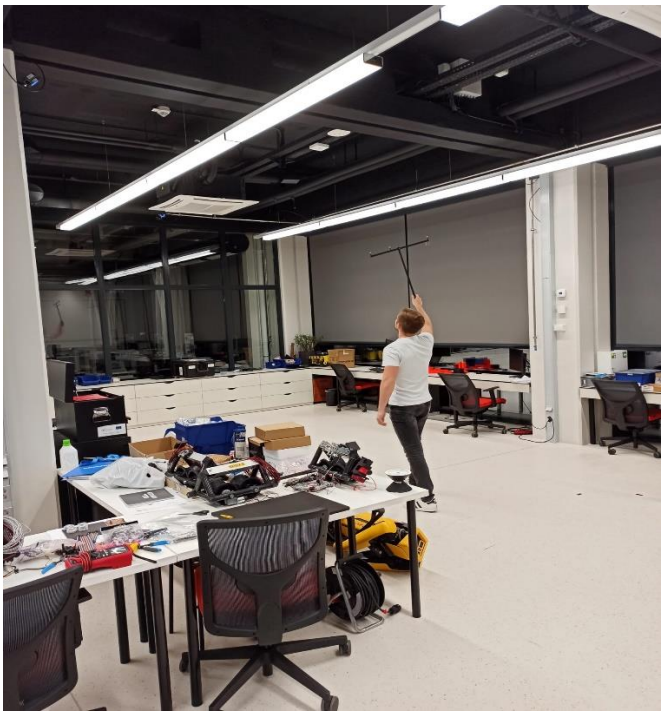


Slika 22. Maskiranje reflektirajućih područja i okluzija markera [10]

5.1.3.2. Prikupljanje kalibracijskih uzoraka pomoću kalibracijskog štapa

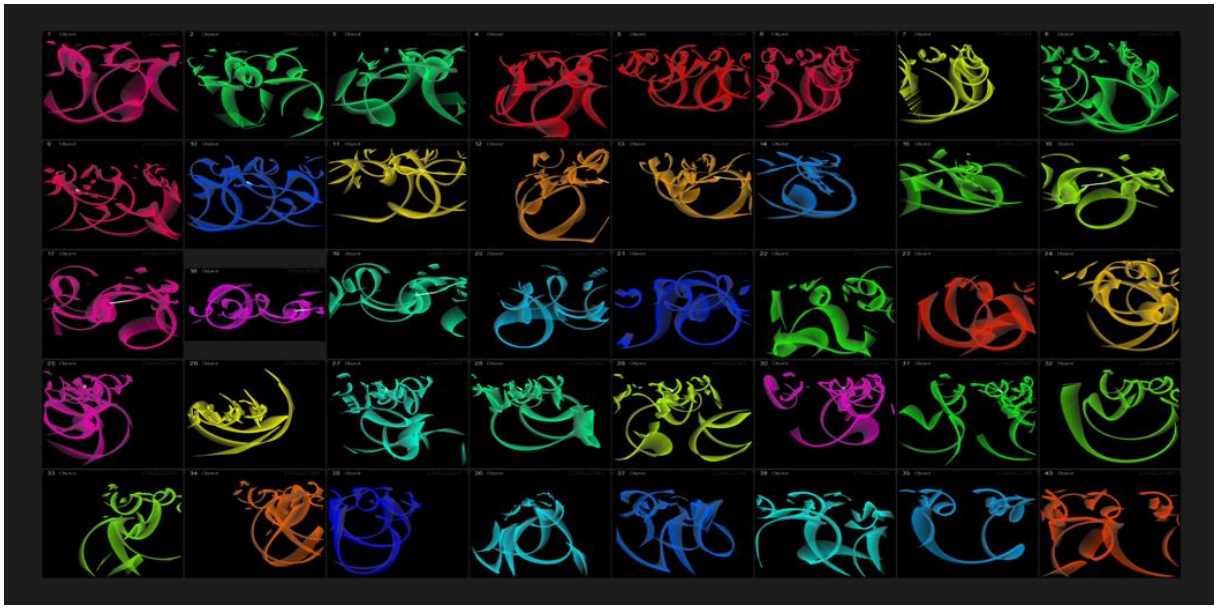
Proces prikupljanja uzoraka je sljedeći. Kalibracijskim štapom, koji na sebi ima pričvršćene markere, neprestano se maše ispred kamera kako bi kamere mogle detektirati markere. Tijekom ovog procesa, svaka kamera prikuplja snimljene uzorke markera kako bi se mogla izračunati njihova odgovarajuća pozicija i orijentacija u 3D prostoru. Prvi korak pri prikupljanju uzoraka je specificiranje modela štapa koji se koristi u *OptiWand* opciji. Nakon toga, određuje se vrsta kalibracije u *Calibration Type* opciji. Ako se kalibracija vrši za novi radni prostor u kojem će

objekt biti sniman, odabire se opcija *Full Calibration.* Tada, mahanjem kalibracijskim štapom ispred kamera unutar radnog prostora se prikupljaju podaci. Mahanje štapa se vrši u obliku broja osam kako bi se prikupili uzorci pri različitim orijentacijama. Također, potrebno je pokriti što je više moguće prostora kako bi se prikupio odgovarajući broj uzoraka.



Slika 23. Prikupljanje uzoraka

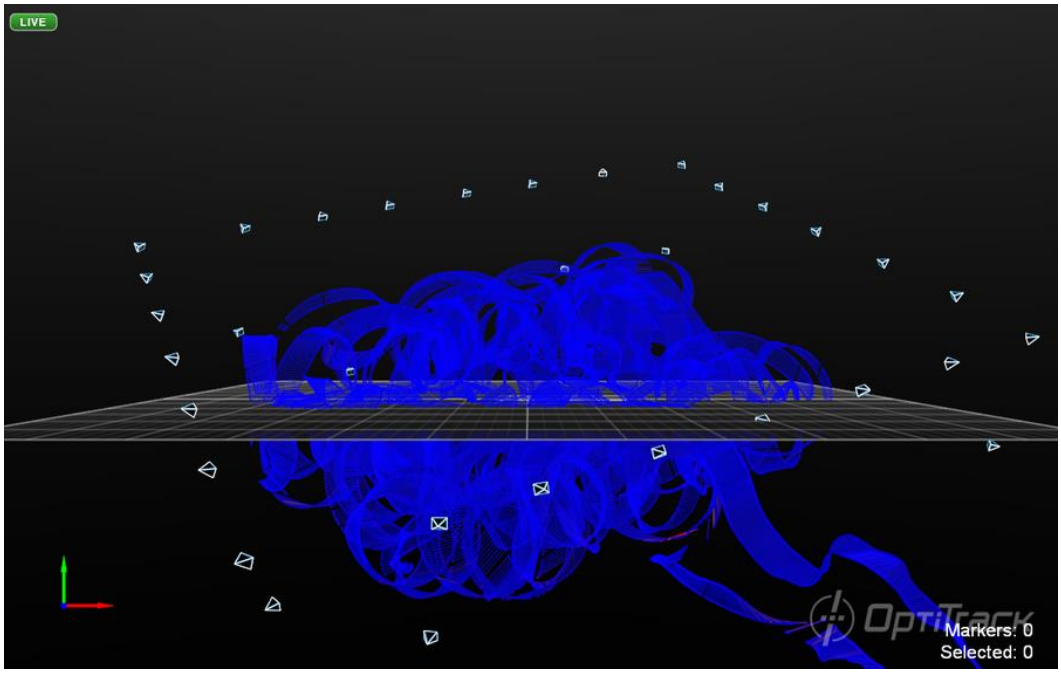
Nakon prikupljanja uzoraka po cijelom prostoru, potrebno je provjeriti 2D pogled u *Camera Preview Pane-u* kako bi se procijenila pokrivenost svake kamere. Svaka kamera bi trebala biti temeljito pokrivena uzorcima te količina uzoraka bi trebala varirati između 2000 – 5000. Prikupljanje preko 10000 uzoraka po kameri bi se trebalo izbjegavati jer pretjeran broj uzoraka može negativno utjecati na kalibraciju. Pritiskom na tipku *Calculate* nakon prikupljanja odgovarajućeg broja uzoraka se vrši kalibracija. Prikaz prikupljenih uzoraka se nalazi na slici 24.



Slika 24. Prikaz prikupljenih uzoraka u 2D Camera Preview Pane-u [10]

5.1.3.3. Postavljanje ravnine tla i ishodišta

Nakon izračuna će se prikazati kamere zajedno sa snimljenim radnim prostorom u Motive *3D View Pane-u*. Međutim, radni prostor neće biti usklađen s koordinatnim sustavom ravnine.



Slika 25. Prikaz rezultata kalibracije [10]

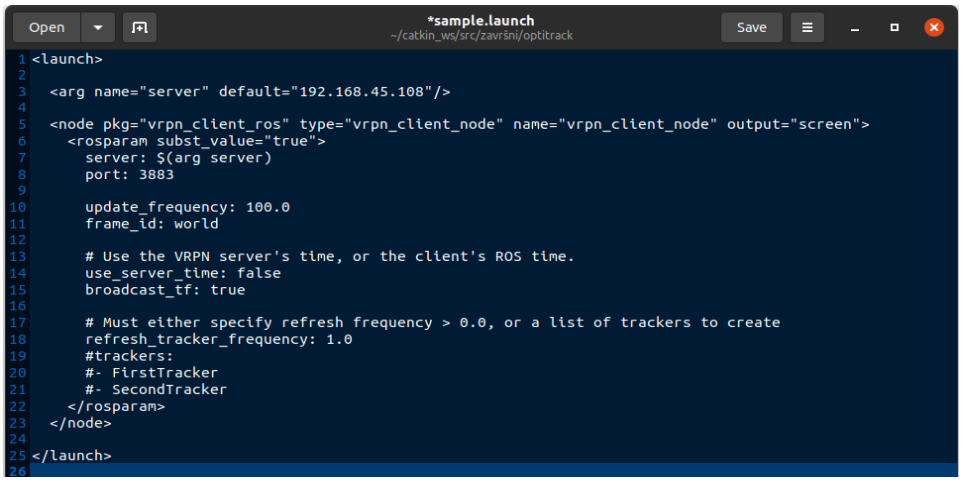
Zadnji korak kalibracijskog procesa je postavljanje ravnine tla i ishodišta sustava. To se postiže postavljanjem kalibracijskog kvadrata na tlo radnog područja gdje bi zamišljeno ishodište koordinatnog sustava trebalo biti. Kraća noga kalibracijskog smjera predstavlja pozitivan smjer X osi, a dulja predstavlja pozitivan smjer Z osi koordinatnog sustava. Prema tome, pozitivan smjer Y osi će se automatski postaviti prema gore prateći pravilo desne ruke. Također, potrebno je unijeti veličinu *Vertical offset* parametra koji predstavlja udaljenost između centra markera koji se nalaze na kalibracijskom kvadratu i tla na kojem se kvadrat nalazi. Kalibracijski kvadrat je postavljen na sredinu radnog područja.

Slika 26. Kalibracijski kvadrat

Nakon postave kalibracijskog kvadrata i označavanja markera u *3D Perspective View-u,* pritiskom na tipku *Set Ground Plane* u *Calibration Pane-u* te spremanjem kalibracijskih podataka se završava proces kalibracije OptiTrack vizijskog sustava. Tada, OptiTrack je spreman za povezivanje s ROS-om.

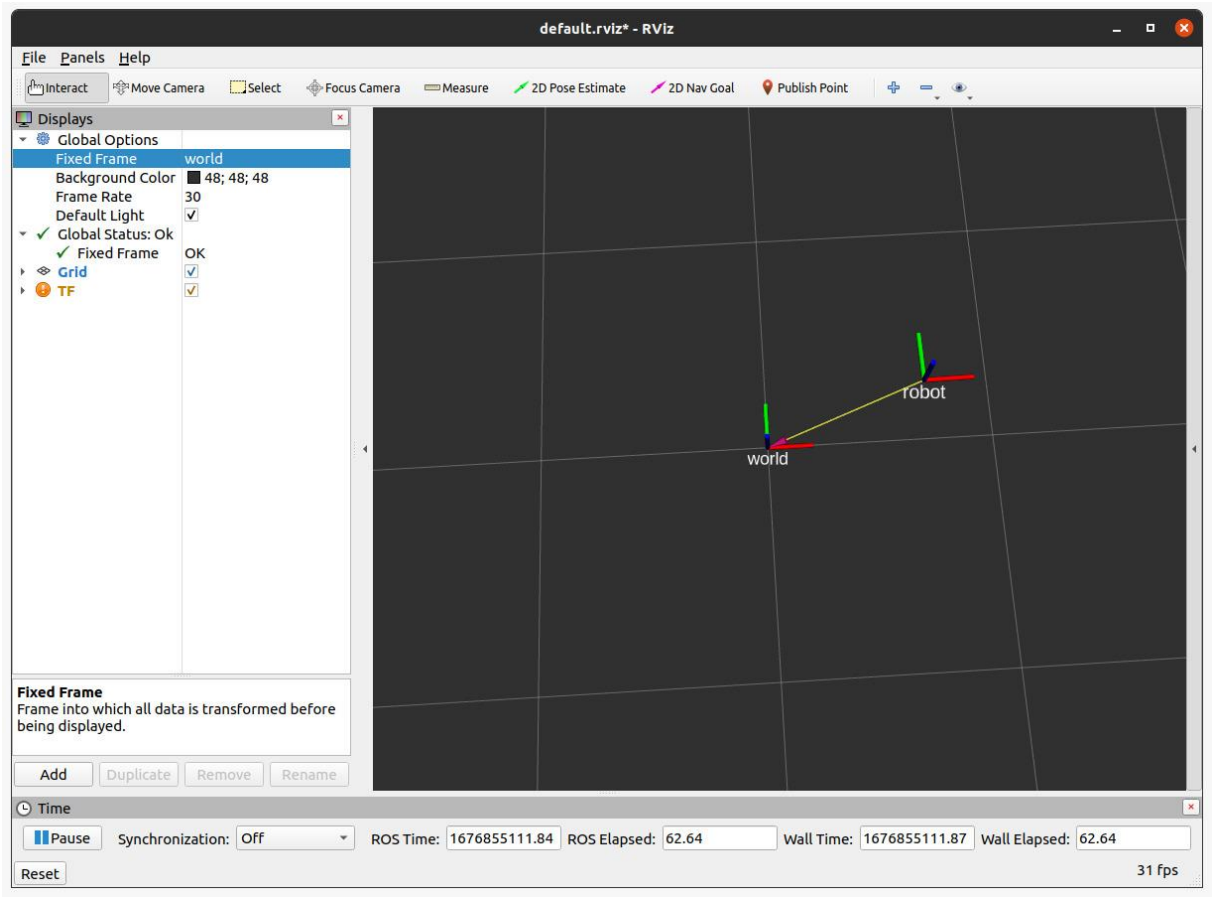
5.1.4. Povezivanje OptiTrack-a s ROS-om

Iako se pomoću OptiTrack-a snima pozicija i orijentacija markera koji se nalaze na mobilnom robotu, OptiTrack mora biti povezan s ROS-om kako bi se snimljeni podaci mogli koristiti za lokalizaciju robota u odnosu na definirani koordinatni sustav (OptiTrack-ov *world* koji je vezan za postavljeno ishodište radnog prostora ili drugi generirani koordinatni sustav povezan s *world-om*) pomoću matrica transformacija, odnosno *tf* čvora. Komunikacija između OptiTrack-a i ROS-a se ostvaruje pomoću VRPN paketa. Detaljan opis VRPN-a se može pronaći na [11]. Nakon instalacije VRPN-a, stvara se paket na osobnom radnom području *"optitrack"* u kojem se kreira *.launch* datoteka *sample.launch* preko koje se pozivaju čvorovi koji omogućuju komunikaciju OptiTrack-a i ROS-a. Izgled *sample.launch* je standardan te se u njega upisuje IP adresa računala na kojem se nalazi OptiTrack. Nakon pokretanja naredbe *catkin_make* u radnom području paket je kreiran te je *sample.launch* datoteka spremna za pozivanje.



Slika 27. Izgled sample.launch datoteke

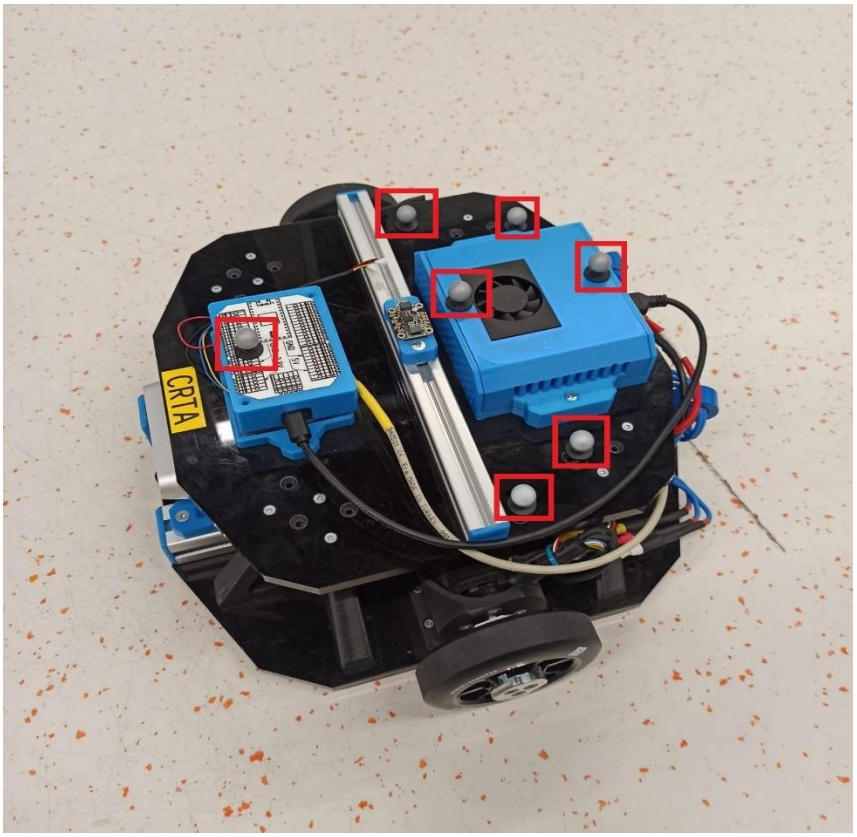
Pokretanjem *sample.launch* datoteke se ostvaruje komunikacija između OptiTrack-a i ROS-a. Kako bi se uvjerili u ostvarenu komunikaciju, pokretanjem RVIZ-a se u sučelju pojavljuju novi koordinatni sustavi: *world* i *robot* koji će biti objašnjeni u sljedećem poglavlju. Pri završetku ovog koraka se može započeti sa snimanjem pokreta mobilnog robota.



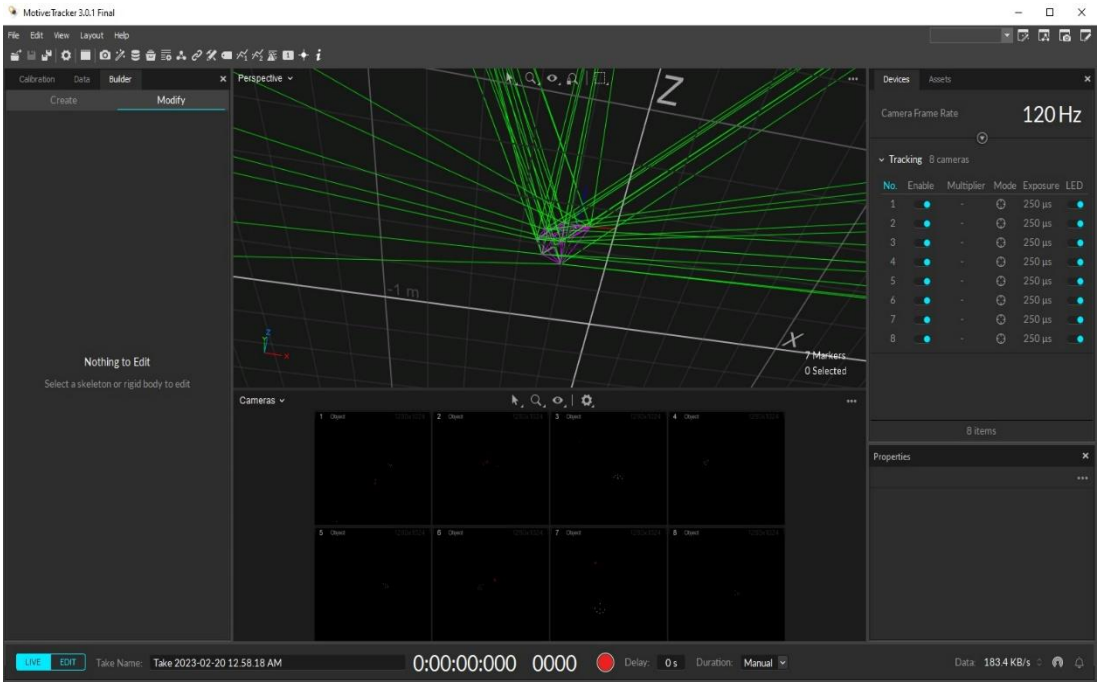
Slika 28. Povezanost ROS-a i OptiTrack-a

5.2. Praćenje pokreta mobilnog robota i mjerenje odstupanja

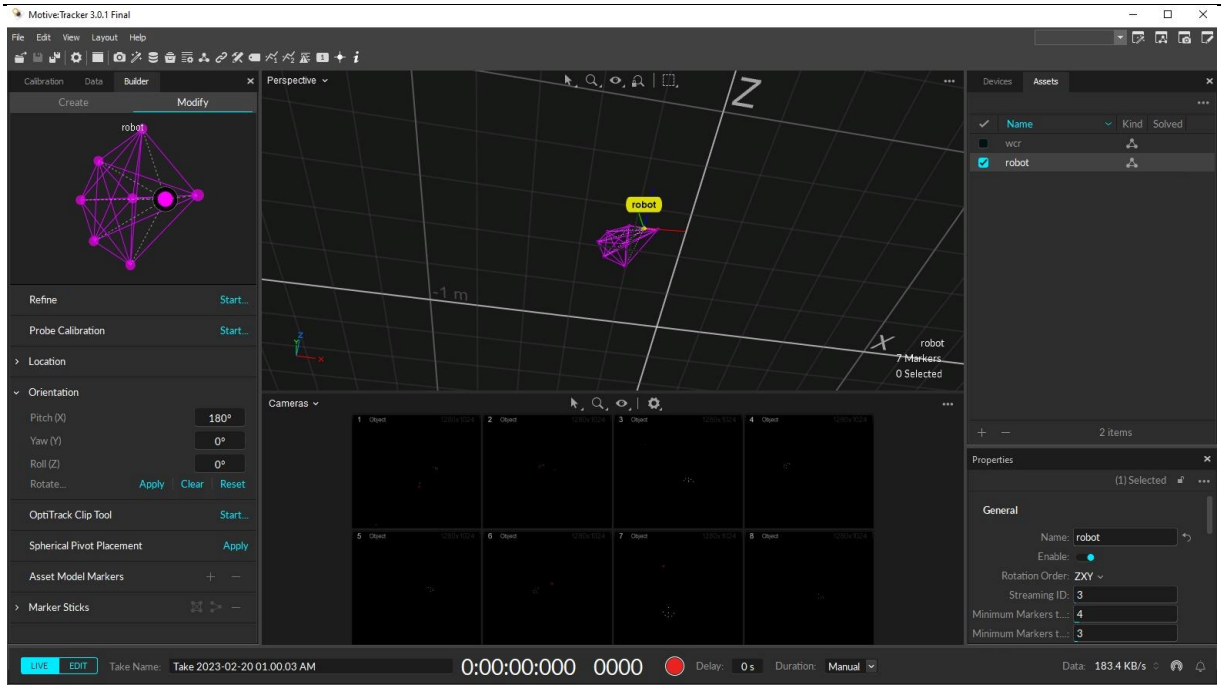
Kako bi praćenje mobilnog robota bilo ostvarivo, potrebno je na njega postaviti markere te ga položiti u radno područje snimanja. Nakon što kamere prepoznaju markere, potrebno ih je definirati kao *Rigid Body* (kruto tijelo) u Motive-u kako bi se izračunalo težište označenog tijela te u njega postavilo koordinatni sustav. Kruto tijelo se definira obuhvaćanjem i označavanjem markera te pritiskom na + u *Assets* odjeljku se odabire *RigidBody*. Novom krutom tijelu se dodjeljuje ime te označavanjem *VRPN Enable on* u *Streaming* odjeljku u postavkama se omogućuje prenošenje informacija ROS-u.



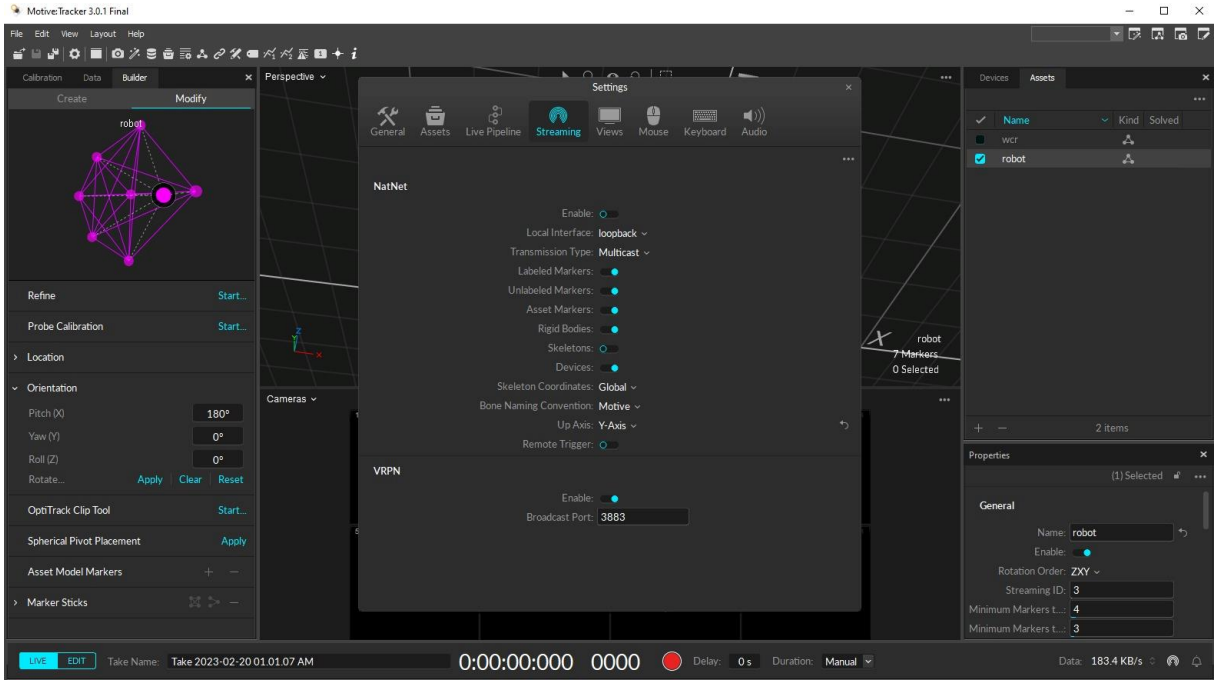
Slika 29. Postavljanje markera na mobilnog robota



Slika 30. Prepoznati i označeni markeri u Motive-u



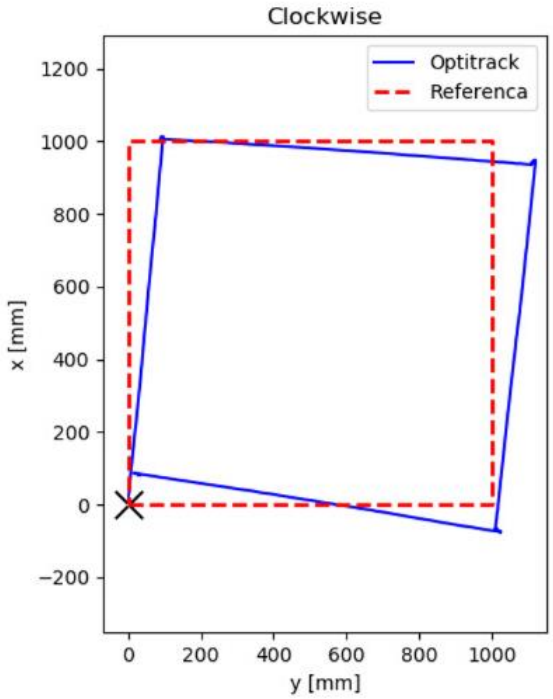
Slika 31. Stvoreno kruto tijelo *robot*



Slika 32. Omogućavanje slanja informacija ROS-u

Pokretanjem *sample.launch* datoteke se primaju informacije o ishodištu radnog prostora snimanja te položaja *robot* koordinatnog sustava u odnosu na njega. Međutim, točne vrijednosti pozicije i orijentacije robota u odnosu na zadani koordinatni sustav koje se mogu koristiti se dobivaju preko *tf* čvora. Stoga, u Python skripti *kvadrat2.py* se nalazi funkcija *getOptitrackPosition()* koja upravo služi tome. No, za razliku od *odom-a* objašnjenog u prethodnim poglavljima, *world* koordinatni sustav ne predstavlja početnu poziciju *robota* već jednostavno ishodište radnog područja snimanja. Iz tog razloga, potrebno je generirati novi koordinatni sustav koji će se stvarati u položaju *robot* koordinatnog sustava te se gasiti i ponovno stvarati nakon prvog obilaska kvadratne putanje kako bi se dobili stvarni početni i konačni položaji robota. Slično kao kod generiranja *odom_pomocnog*, u funkciji *odom_optitrack()* skripte *odom_optitrack.py* se "sluša" trenutna transformacija između koordinatnog sustava *robot* i fiksiranog sustava *world*. Nakon toga, na trenutnoj statičnoj poziciji *robota* u odnosu na *world* se stvara novi koordinatni sustav *odom_optitrack* te se tamo zadržava do terminacije spomenute skripte. Pozivanje i terminacija skripte *odom_optitrack.py* se odvija unutar skripte *kvadrat2.py* pomoću funkcije *Odom_optitrack()*. Na kraju, može se reći da povezivanjem robota s radnim prostorom na osobnom računalu preko *.bashrc* datoteke se generiraju koordinatni sustavi *base_link* i *odom* preko kojih se dobiva *odom_pomocni* te oni služe pri određivanju kvadratne putanje putem regulacije brzine koja je uvijek savršena (jedno *tf stablo*), dok se povezivanjem OptiTrack-a s ROS-om generiraju *robot* i *world* preko kojih se dobiva *odom_optitrack* te oni služe za praćenje stvarne kvadratne putanje, bilježenje početnih i krajnjih položaja robota te izračun kalibracijskih parametara robota (drugo *tf* stablo).
Napokon, pokretanjem skripte *kvadrat2.py* robot obilazi kvadratnu putanju pet puta u smjeru kazaljke na satu i pet puta u smjeru obrnutom od kazaljke na satu dok se za to vrijeme obavlja snimanje stvarnog položaja robota. Položaj robota se sprema u *.txt* file koji se poziva u funkciji *Plotting()* te se dobiva grafički prikaz rezultata.

Iz grafičkog prikaza rezultata na slici 33. se vidi koliko robot odstupa od referentne kvadratne putanje nakon jednog obilaska te se zbog toga mora provesti izračun kalibracijskih parametara mobilnog robota nakon čega će se njihovom primjenom kretanje približiti referenci.



Slika 33. Prikaz stvarne kvadratne putanje nakon jednog obilaska

Međutim, treba uzeti u obzir da težište *Rigid Body-ja* nije u centru rotacije robota zbog nasumičnosti i nesimetričnosti položaja markera. Iz tog razloga, prilikom kretanja, transformacija koordinatnih sustava i na kraju plotanja rezultata, robot pri rotaciji opisuje dio kružnice umjesto točke na kutevima trajektorije kako je prikazano na slici 33. Stoga je potrebno pronaći centar rotacije te težište *Rigid Body-ja* pomaknuti u tu točku. Kako bi se to ostvarilo, robotove translacijske brzine se izjednačavaju s nulom dok se zadaje određena rotacijska brzina prilikom koje se vrši transformacija između težišta *Rigid Body-ja* i *world* koordinatnog sustava. Položaji koji približno opisuju kružnicu se spremaju u *.txt* dokument te se pomoću dobivenih podataka traži odgovarajuća kružnica s definiranim radijusom. Budući da je položajima markera približno pogođena Y os, za dobivenu vrijednost radijusa se translatira položaj *Rigid Body-ja* u smjeru X osi. Slika 31. pokazuje definiran robotov koordinatni sustav s već izvršenom translacijom. Vrijednosti radijusa za različite brzine su prikazane u Tablici 4. u nastavku.

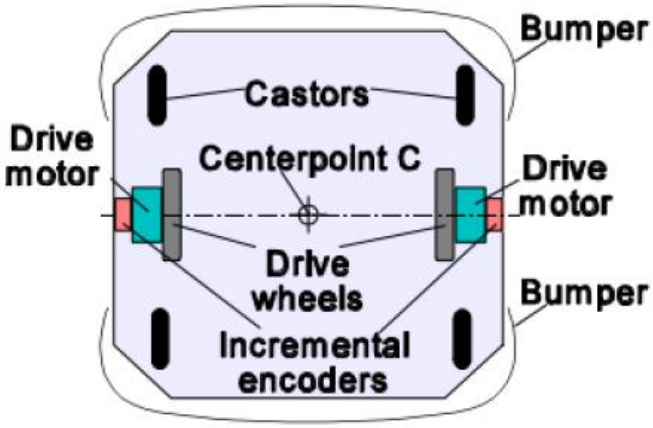
Tablica 4. Radijusi pronađene kružnice za odabranu kutnu brzinu

Kutna brzina (rad/s)	r (mm)
0.1	63.6
0.2	64.2
0.3	64.4
0.4	64.3
0.5	64.5
0.6	64.7
0.7	64.1
0.8	63.8
0.9	63.5
1	63.8
1.5	61.0

Iz priloženih podataka u Tablici 4. je vidljivo kako radijus konvergira na vrijednosti od 64 mm te vrijednost koja je odabrana za translaciju je r = 64.1 mm pri rotacijskoj brzini od 0.7 rad/s.

6. UMBMARK KALIBRACIJA MOBILNOG ROBOTA

Sposobnost kretanja po prostoru je osnovna karakteristika mobilnih robota te da bi kretanje bilo što kvalitetnije, potrebno je osigurati što točniju estimaciju odometrije. U konstrukciji mobilnih robota diferencijalne kinematike, inkrementalni enkoderi su postavljeni na pogonske motore kako bi brojali okretaje kotača. Koristeći jednostavne geometrijske jednadžbe, trenutna pozicija robota u odnosu na početnu poziciju se lako izračunava. Međutim, na izračun odometrije utječu sustavne i nesustavne greške.



Slika 34. Osnovni oblik mobilnog robota diferencijalne kinematike [12]

6.1. Nesustavne greške

Nesustavne greške su greške uzrokovane interakcijom robota s nepredvidivim svojstvima okoliša. Neke od nesustavnih grešaka su:

- Neujednačena površina (ispupčenja, pukotine, prašina)
- Prelazak preko neočekivanih objekata na površini
- Proklizivanje kotača zbog skliske podloge, prevelike akceleracije, naglog skretanja te vanjskih (interakcija s vanjskim objektima) i unutarnjih sila

Nesustavne greške predstavljaju značajan problem jer se na njih teško utječe. Međutim, pri glatkim površinama zatvorenih prostora, sustavne greške daleko više utječu na estimaciju odometrije.

6.2. Sustavne greške

Sustavne greške su poprilično ozbiljne budući da se konstantno gomilaju. Najčešće su uzrokovane konstrukcijskim nesavršenostima i implementacijom komponenata mobilnog robota. Sustavne greške su:

- Nejednaki promjeri kotača
- Prosjek promjera kotača se razlikuje od nominalnog promjera
- Neusklađenost kotača
- Nesigurnost u razmaku kotača
- Ograničena rezolucija enkodera
- Ograničena frekvencija uzorkovanje enkodera

Međutim, najkobnije greške od navedenih su *nejednak promjer kotača* i *nesigurnost u razmaku između kotača* te su iz tog razloga detaljno pojašnjeni. Za ostale će se pretpostaviti da imaju malen utjecaj ili da su lako ispravljive te ne treba provoditi eksperimente za njihove eliminacije.

6.2.1. Nejednaki promjeri kotača

Budući da je proklizavanje nimalo nije poželjno, kotače je potrebno proizvoditi od gume koja s većinom podloga povećava trenje. No, proizvodnja gumenih kotača s potpuno istim promjerom je poprilično zahtjevna. Nadalje, zbog elastičnih svojstva gume, gumeni kotači se pod utjecajem nesimetričnog opterećenje različito tlače što također utječe na nejednak promjer kotača. Bilo koji od navedenih uzroka stvara značajne greške u odometriji mobilnog robota. Međutim, budući da se na tu grešku može utjecati, definira se izraz:

$$E_d = \frac{D_R}{D_L} \tag{10}$$

gdje je D_R stvarni promjer desnog kotača, a D_L stvarni promjer lijevog kotača. E_d pokazuje omjer desnog i lijevog kotača te je idealan omjer 1.

6.2.2. Nesigurnost u razmaku između kotača

Razmak između kotača se definira kao udaljenost dodirnih točaka dvaju kotača diferencijalnog mobilnog robota s podlogom. Nesigurnost u točan razmak kotača je uzrokovana činjenicom da gumeni kotači ne dodiruju podlogu u dodirnoj točki već u dodirnoj površini. Stoga, u komercijalnim mobilnim robotima razlika između stvarne i konstrukcijske udaljenosti varira oko 1%. Kao i na nejednake promjere kotača, na razmak između kotača se također može utjecati. Dakle, izraz za tu grešku je:

$$E_b = \frac{b_a}{b_n} \tag{11}$$

gdje je b_a stvarni, a b_n nominalni razmak među kotačima. Idealan omjer je također 1.
Na kraju, važno je napomenuti da greška E_b utječe samo na rotaciju, dok greška E_d utječe na pravocrtno gibanje. Također, obje veličine su bezdimenzijske.

6.3. Mjerenje sustavnih grešaka

Kao što je već spomenuto, sustavne greške su vezane za samog mobilnog robota te se obično ne mijenjaju tijekom kretanja. Stoga, na osiguravanje što točnije odometrije se može utjecati njihovim mjerenjem te im se može suprotstaviti njihovom primjenom u softveru. Postoje dva načina mjerenja sustavnih grešaka. Prvi je "*The uni-directional square path experiment" (hrv. eksperiment jednosmjerne kvadratne putanje),* dok je drugi "*The bi-directional square path experiment" (hrv. eksperiment dvosmjerne kvadratne putanje)* koji se također naziva i UMBMark te su oba objašnjena u nastavku.

6.3.1. The uni-directional square path experiment

Eksperiment se provodi zadavanjem jednosmjerne kvadratne putanje 4x4 m mobilnom robotu. Robot kreće iz početne pozicije x_0, y_0, θ_0; (0, 0, 0). Međutim, nakon obilaženja putanje, zbog odometrijskih grešaka se ne vraća točno u početnu točku. Nakon toga, mjere se odstupanja od početne pozicije.

Nakon usporedbe s početnom pozicijom, dobiva se rezultat greške u poziciji i orijentaciji te su definirani kao:

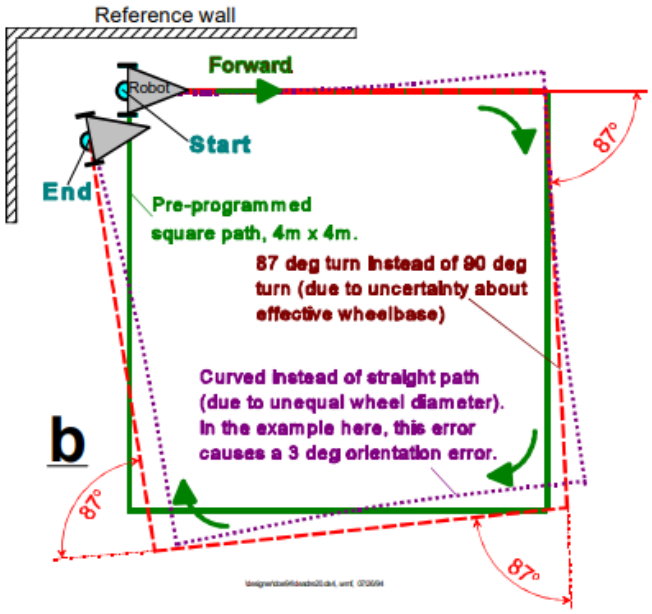
$$\varepsilon x = x_{abs} - x_{calc} \tag{12}$$

$$\varepsilon y = y_{abs} - y_{calc} \tag{13}$$

$$\varepsilon \theta = \theta_{abs} - \theta_{calc} \tag{14}$$

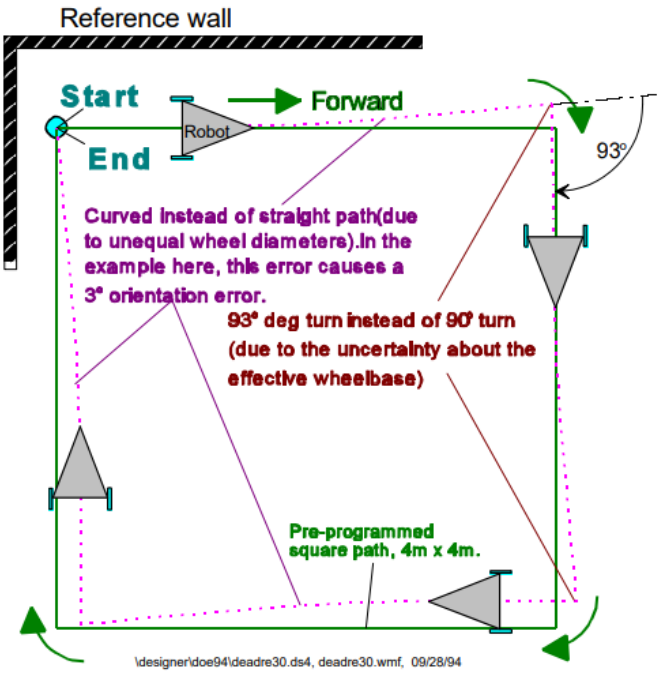
Gdje su:

- εx, εy, $\varepsilon \theta$ – greške nastale zbog odometrije
- x_{abs}, y_{abs}, θ_{abs} – apsolutna pozicija i orijentacija robota (u ovom slučaju početna pozicija)
- x_{calc}, y_{calc}, θ_{calc} – pozicija i orijentacija robota dobivena iz odometrije (konačna pozicija)



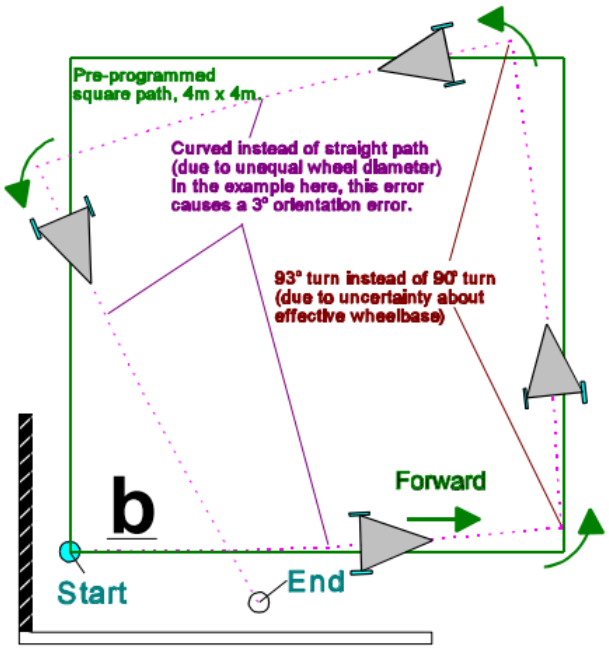
Slika 35. Utjecaj sustavnih grešaka E_b ili E_d na robotovo kretanje koje mogu prouzrokovati istu grešku konačnog položaja [12]

Nakon analiziranja rezultata eksperimenta mogu se zaključiti dvije stvari. Greška u položaju je uzrokovana E_d parametrom kao što pokazuje istočkana linija na slici 35; ili, greška u položaju je uzrokovana parametrom E_b što pokazuje iscrtkana linija. Kao što je prikazano, u oba slučaja se može proizvesti jednaka greška položaja. Činjenica da dva različita mehanizma stvaranja greške stvaraju istu grešku može dovesti do pogreške ispravljanja samo jednog parametra. Ozbiljnost ove greške je dokazana činjenicom da se naizgled dobivaju zadovoljavajući rezultati koji su prikazani na slici 36.



Slika 36. Podešavanje putanje nakon ispravljanja samo jednog parametra [12]

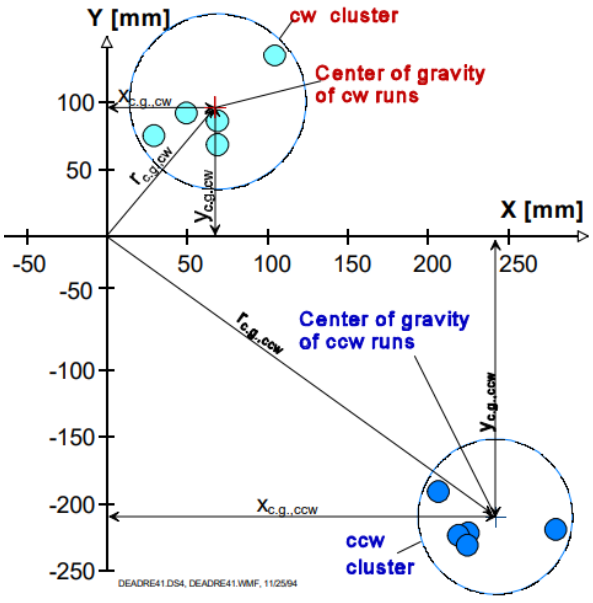
Međutim, skrivena greška pogrešnog zaključka se otkriva ako robot prođe kvadratnu putanju u suprotnom smjeru. Slika 37. upravo to pokazuje. Međusobno kompenziranje parametara nakon prolaska putanje u jednom smjeru te ispravljanje samo jednoga uzrokuje nadodavanje sustavnih grešaka i povećava ukupnu grešku u suprotnom smjeru. Stoga, zaključuje se da je *eksperiment jednosmjerne kvadratne putanje* neprikladan za osiguravanje što točnije estimacije odometrije mobilnog robota. Iz tog razloga se uvodi *eksperiment dvosmjerne kvadratne putanje – UMBMark.*



Slika 37. Prolazak putanje u suprotnom smjeru nakon nadodavanja sustavnih grešaka [12]

6.3.2. The bi-directional square path experiment - UMBMark

Slično kao i u jednosmjernom eksperimentu, robotu se zadaje kvadratna putanja 4x4 m te kreće iz početnog položaja x_0, y_0, θ_0; (0, 0, 0). Međutim, za razliku od prethodnog eksperimenta, putanja se izvodi pet puta u smjeru kazaljke na satu – *cw*, te pet puta u smjeru suprotnom od kazaljke na satu - *ccw*. Nakon izvođenja, rezultati izgledaju ovako:



Slika 38. Prikaz rezultata nakon UMBMark eksperimenta [12]

Slika 38. pokazuje krajnje položaje grupirane prema smjeru obilaženja kvadratne putanje. Iako je distribucija krajnjih položaja također rezultat i nesustavnih grešaka, utjecaj sustavnih grešaka na nekalibriranog robota voženog po glatkoj podlozi je znatno veći. Kako bi se utjecaj nesustavnih grešaka smanjio, potrebno je odrediti težišta dvaju klastera koji se dobivaju sljedećim jednadžbama:

$$x_{c.g.,cw/ccw} = \frac{1}{n}\sum_{i=1}^{n} \varepsilon x_{i,cw/ccw} \tag{15}$$

$$y_{c.g.,cw/ccw} = \frac{1}{n}\sum_{i=1}^{n} \varepsilon y_{i,cw/ccw} \tag{16}$$

gdje je *n* broj vožnji u svakom smjeru dok su ostali parametri jednaki parametrima iz jednadžbi (12) i (13). Također, apsolutne udaljenosti između ishodišta i oba težišta se izračunavaju kao:

$$r_{c.g.,cw/ccw} = \sqrt{(x_{c.g.,cw/ccw})^2 + (y_{c.g.,cw/ccw})^2} \tag{17}$$

Nakon čega se veća vrijednost uzima kao točnost odometrije te se koristi umjesto srednje vrijednosti dvaju težišta zato što je ta vrijednost, kao najveća moguća, korisnija za praktičnu primjenu. Također, orijentacija robota nije uzeta u obzir zato što sustavna greška orijentacije, zbog simetričnosti trajektorije s obje strane, direktno utječe na grešku pozicije te će navedeno biti objašnjeno u nastavku.

$$E_{max} = \max\left(r_{c.g.,cw}, r_{c.g.,ccw}\right) \tag{18}$$

6.3.3. Analiza sustavnih grešaka

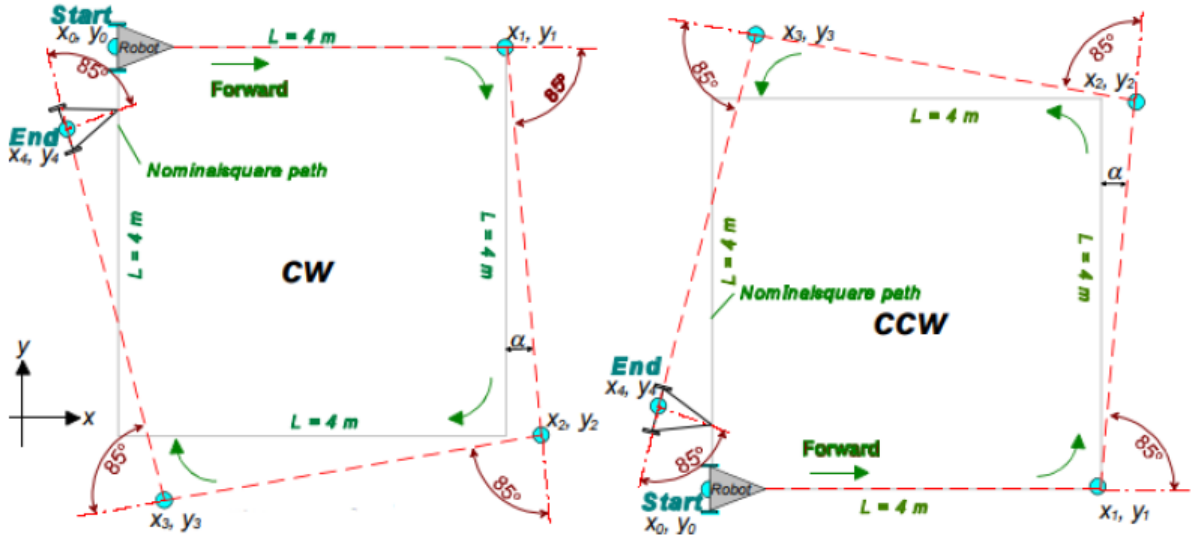
Nakon definiranja UMBMark metode, potrebno je analizirati tipove pogrešaka unutar istog te definirati matematičke modele koje ih određuju.

6.3.3.1. *Greška tipa A*

Greška tipa A je definirana kao orijentacijska greška koja smanjuje ili povećava ukupnu količinu rotacije robota tijekom izvođenja kvadratne putanje u smjeru kazaljke na satu, ali i u smjeru obrnutom od kazaljke na satu. Taj tip greške nastaje zbog razlike u stvarnoj i nazivnoj udaljenosti između kotača E_b. Ako se može primijetiti da u smjeru kazaljke na satu i u smjeru obrnutom od kazaljke na satu vrijedi:

$$\left|\theta_{ukupni,cw}\right| < \left|\theta_{nazivni,cw}\right| \text{ i } \left|\theta_{ukupni,ccw}\right| < \left|\theta_{nazivni,ccw}\right| \qquad (19)$$

gdje je vrijednost nazivnog kuta 90°, radi se o navedenom tipu greške koji je prikazan na slici 39.



Slika 39. Tip greške A u smjeru kazaljke na satu i smjeru obrnutom od kazaljke na satu [12]

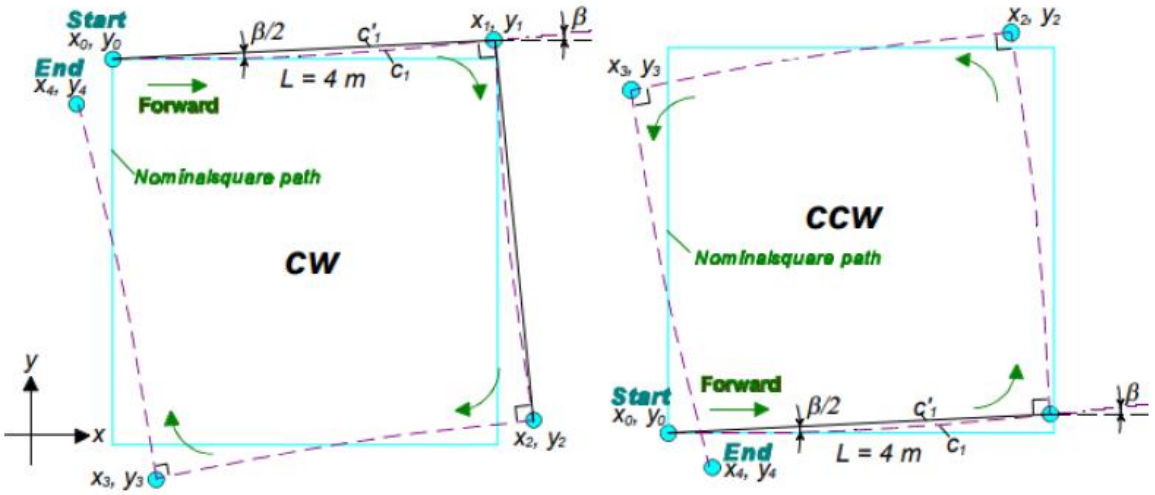
6.3.3.2. *Greška tipa B*

Greška tipa B je definirana kao orijentacijska greška koja smanjuje ili povećava ukupnu rotaciju robota u jednom smjeru dok suprotno tome povećava ili smanjuje ukupnu rotaciju robota u drugom smjeru tijekom izvođenja kvadratne putanje. Taj tip greške nastaje zbog razlike u promjeru kotača E_d.

Ako se može primijetiti da vrijedi:

$$\left|\theta_{ukupni,cw}\right| < \left|\theta_{nazivni,cw}\right| \text{ i } \left|\theta_{ukupni,ccw}\right| > \left|\theta_{nazivni,ccw}\right| \qquad (20)$$

i obrnuto, radi se o navedenom tipu greške prikazanom na slici 40.



Slika 40. Tip greške B u smjeru kazaljke na satu i smjeru obrnutom od kazaljke na satu [12]

6.3.3.3. Matematičke pretpostavke

Kako bi se pojednostavio matematički model grešaka, uvodi se nekoliko pretpostavki. Prva pretpostavka je da je početna pozicija robota (x_0, y_0) u $(0, 0)$. Nadalje, uvedene aproksimacije su valjanje za male kuteve grešaka koji su nastali uslijed izvođenja kvadratne putanje u oba smjera te one glase:

$$\begin{aligned} L\,sin\gamma &\approx L\gamma \\ L\,sin2\gamma &\approx 2L\gamma \\ L\,sin3\gamma &\approx 3L\gamma \end{aligned} \qquad (21)$$

$$\begin{aligned} L\,cos\gamma &\approx L \\ L\,cos2\gamma &\approx L \\ L\,cos3\gamma &\approx L \end{aligned} \qquad (22)$$

gdje su:

- L – duljina svake stranice kvadratne putanje
- γ – svaka inkrementalna greška uzrokovana $E_d$ili E_b

6.3.3.4. Matematički model grešaka tipa A i B

Izrazi za grešku tipa A u smjeru kazaljke na satu su sljedeći (sve oznake vidljive na slici 39.):

$$\begin{aligned} x_1 &= x_0 + L \\ y_1 &= y_0 \end{aligned} \tag{23}$$

$$\begin{aligned} x_2 &= x_1 + L\,sin\alpha \approx L + L\alpha \\ y_2 &= y_1 - L\,cos\alpha \approx -L \end{aligned} \tag{24}$$

$$\begin{aligned} x_3 &= x_2 - L\,cos2\alpha \approx L\alpha \\ y_3 &= y_2 - L\,sin2\alpha \approx -L - 2L\alpha \end{aligned} \tag{25}$$

$$\begin{aligned} x_4 &= x_3 - L\,sin3\alpha \approx -2L\alpha \\ y_4 &= y_3 + L\,cos3\alpha \approx -2L\alpha \end{aligned} \tag{26}$$

Izrazi za grešku tipa A u suprotnom smjeru od kazaljke na satu su sljedeći (sve oznake vidljive na slici 39):

$$\begin{aligned} x_1 &= x_0 + L \\ y_1 &= y_0 \end{aligned} \tag{27}$$

$$\begin{aligned} x_2 &= x_1 + L\,sin\alpha \approx L + L\alpha \\ y_2 &= y_1 + L\,cos\alpha \approx L \end{aligned} \tag{28}$$

$$\begin{aligned} x_3 &= x_2 - L\,cos2\alpha \approx L\alpha \\ y_3 &= y_2 + L\,sin2\alpha \approx L + 2L\alpha \end{aligned} \tag{29}$$

$$x_4 = x_3 - L\,sin3\alpha \approx -2L\alpha$$
$$y_4 = y_3 - L\,cos3\alpha \approx 2L\alpha \tag{30}$$

Za razliku od greške tipa A, greška tipa B uzrokuje lagano zakrivljenu putanju između dvije točke umjesto ravne linije te zbog toga nastaje orijentacijska greška β na kraju svakog pomaka. Nadalje, linija koja spaja kutne točke stvarne putanje c'_1 ima nagib od $\frac{\beta}{2}$ zbog paralelnosti s tangentom u središtu luka c_1. Sukladno tome proizlaze matematički izrazi za taj tip greške.

Izrazi za grešku tipa B u smjeru kazaljke na satu su sljedeći (sve oznake vidljive na slici 40.):

$$x_1 = x_0 + L\,cos\frac{\beta}{2} \approx L$$
$$y_1 = y_0 + L\,sin\frac{\beta}{2} \approx L\frac{\beta}{2} \tag{31}$$

$$x_2 = x_1 + L\,sin\frac{3\beta}{2} \approx L + 3L\frac{\beta}{2}$$
$$y_2 = y_1 - L\,cos\frac{3\beta}{2} \approx L\frac{\beta}{2} - L \tag{32}$$

$$x_3 = x_2 - L\,cos\frac{5\beta}{2} \approx 3L\frac{\beta}{2}$$
$$y_3 = y_2 - L\,sin\frac{5\beta}{2} \approx -L(2\beta + 1) \tag{33}$$

$$x_4 = x_3 - L\,sin\frac{7\beta}{2} \approx -2L\beta$$
$$y_4 = y_3 - L\,cos\frac{7\beta}{2} \approx -2L\beta \tag{34}$$

Izrazi za grešku tipa B u suprotnom smjeru od kazaljke na satu su sljedeći (sve oznake vidljive na slici 40.):

$$x_1 = x_0 + L\cos\frac{\beta}{2} \approx L$$
$$y_1 = y_0 + L\sin\frac{\beta}{2} \approx L\frac{\beta}{2} \tag{35}$$

$$x_2 = x_1 - L\sin\frac{3\beta}{2} \approx L - \frac{3\beta}{2}$$
$$y_2 = y_1 + L\cos\frac{3\beta}{2} \approx L\frac{\beta}{2} + L \tag{36}$$

$$x_3 = x_2 - L\cos\frac{5\beta}{2} \approx -3L\frac{\beta}{2}$$
$$y_3 = y_2 - L\sin\frac{5\beta}{2} \approx -2L\beta + L \tag{37}$$

$$x_4 = x_3 + L\sin\frac{7\beta}{2} \approx 2L\beta$$
$$y_4 = y_3 - L\cos\frac{7\beta}{2} \approx -2L\beta \tag{38}$$

Superponiranjem grešaka tipa A i tipa B za kretanje robota u smjeru osi *x* i smjeru kazaljke na satu dobiju se izrazi:

$$x_{c.g.,cw} = -2L(\alpha + \beta) \tag{39}$$
$$x_{c.g.,ccw} = -2L(\alpha - \beta) \tag{40}$$

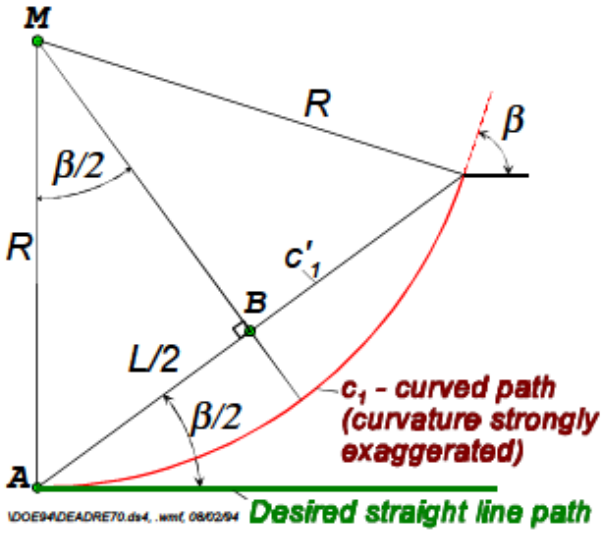
čijim se oduzimanjem (40) od (39) i sređivanjem dobije izraz za kut β u stupnjevima:

$$\beta = \frac{x_{c.g.,cw} - x_{c.g.,ccw}}{-4L} * \frac{180°}{\pi} \tag{41}$$

Također, slični rezultati vrijede i za pomake po osi *y*:

$$\beta = \frac{y_{c.g.,cw} - y_{c.g.,ccw}}{-4L} * \frac{180°}{\pi} \tag{42}$$

Nadalje, polumjer krivulje *R* prikazan na slici 40. može se izračunati pomoću jednostavnih geometrijskih odnosa i ABM trokuta prikazanog na slici 41. Također, zakrivljenje je jako naglašeno radi preglednosti.



Slika 41. ABM trokut za izračun polumjera zakrivljenosti kretnje robota [12]

Izraz za polumjer zakrivljenosti glasi:

$$R = \frac{L/2}{\sin\left(\frac{\beta}{2}\right)} \tag{43}$$

Nakon što je polumjer izračunat, omjer promjera lijevog i desnog kotača E_d, koji je bio uzrok robotovog zakrivljenog kretanja, se jednostavno dobije iz izraza:

$$E_d = \frac{D_R}{D_L} = \frac{R + \frac{b}{2}}{R - \frac{b}{2}} \tag{44}$$

Odnos jednadžbi (39) i (40) također može poslužiti definiranju kuta α. Stoga, njihovim zbrajanjem i sređivanjem, izraz za kut α u stupnjevima za kretanje u smjeru osi *x* glasi:

$$\alpha = \frac{x_{c.g.,cw} - x_{c.g.,ccw}}{-4L} * \frac{180°}{\pi} \tag{45}$$

Dok za pomake u smjeru osi *y*:

$$\alpha = \frac{y_{c.g.,cw} - y_{c.g.,ccw}}{-4L} * \frac{180°}{\pi} \tag{46}$$

Pomoću izračunatog kuta α je moguće izračunati grešku u omjeru nazivne i stvarne udaljenosti kotača E_b. Uzevši u obzir da je razmak između kotača *b* direktno proporcionalan stvarnoj količini rotacije, može se zapisati:

$$\frac{b_a}{90°} = \frac{b_n}{90° - \alpha} \tag{47}$$

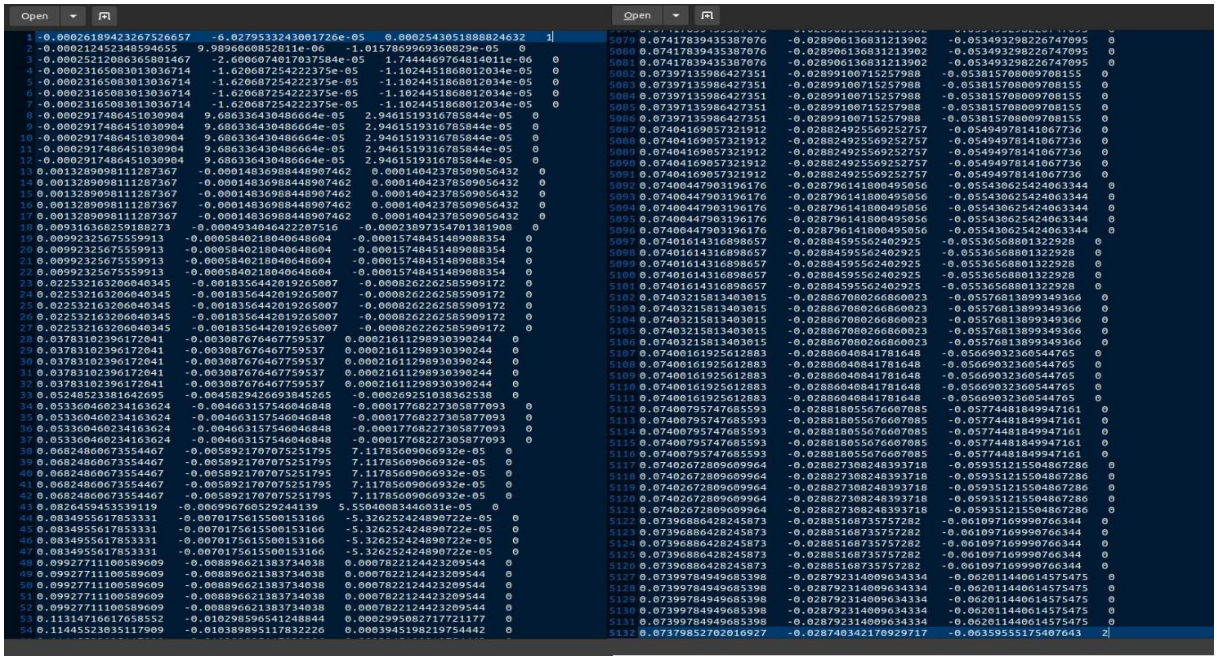
Sređivanjem izraza (47), dobije se E_b:

$$E_b = \frac{b_a}{b_n} = \frac{90°}{90° - \alpha} \tag{48}$$

7. AUTOMATSKA KALIBRACIJA MOBILNOG ROBOTA

7.1. Izračun kalibracijskih parametara

Nakon pokretanja i povezivanja vanjskog vizijskog sustava OptiTrack s mobilnim robotom, pokretanjem skripte *kvadrat2.py* se pokreće robot koji izvršava kvadratnu trajektoriju pet puta u smjeru kazaljke na satu i pet puta u smjeru obrnutom od kazaljke na satu. Tijekom izvođenja trajektorija se obavlja snimanje pozicija robota pomoću OptiTrack-a tako da se u svakom trenutku vrši transformacija između koordinatnog sustava *robot* vezanog za robota koji se kreće i fiksnog koordinatnog sustava *odom_optitrack* u odnosu na kojeg se vrši transformacija te se dobivaju podaci položaja u X, Y ravnini. Ti podaci se spremaju u *.txt* datoteke *opt_cw.txt* za smjer kazaljke na satu i *opt_ccw.txt* za smjer suprotnom od kazaljke na satu. Budući da se nakon svakog kruga koordinatni sustav *odom_optitrack* resetira kako je definirano u *loop* petlji skripte i pojavljuje na drugom mjestu (mjesto gdje je robot završio prethodnu trajektoriju) unutar svijeta (*world* koordinatni sustav) te se ponovno vrši snimanje u odnosu na novo generirani *odom_optitrack*, svaka datoteka sadrži pet početnih pozicija koje uvijek imaju koordinate (0, 0) jer robot uvijek kreće iz pozicije gdje je generiran *odom_optitrack* i pet krajnjih pozicija gdje je kretanje završeno. Podaci su spremljeni tako da svaka početna pozicija na kraju retka ima dodan element “1”, a krajnja pozicija element “2”. Pozicije između imaju dodan element “0” te nakon obilaženja trajektorija *.txt* document izgleda kako je prikazano na slici 42.



Slika 42. Zapis snimljenog položaja robota u .txt datoteci

Nakon ispisa svih podataka, mogu se odrediti odstupanja od ishodišta za kretanje robota u smjeru kazaljke na satu i u smjeru obrnutom od kazaljke na satu koji su prikazani u tablicama 5 i 6.

Tablica 5. Odstupanja od ishodišta - cw smjer kretanja

Broj snimanja	Greška u smjeru osi X	Greška u smjeru osi Y
1.	108.341 mm	-17.993 mm
2.	76.375 mm	-121.292 mm
3.	100.16 mm	-91.131 mm
4.	90.865 mm	-38.128 mm
5.	154.67 mm	-52.025 mm

Tablica 6. Odstupanja od ishodišta – ccw smjer kretanja

Broj snimanja	Greška u smjeru osi X	Greška u smjeru osi Y
1.	26.9 mm	29.09 mm
2.	31.963 mm	33.037 mm
3.	97.429 mm	57.545 mm
4.	95.04 mm	39.053 mm
5.	17.185 mm	-31.888 mm

S izvršenim kvadratnim trajektorijama pet puta u oba smjera te snimljenim i zapisanim pozicijama robota, u Python skripti se poziva funkcija *Calibration()* koja prikuplja podatke iz onih redaka u *.txt* dokumentima koji završavaju s elementom "1" te ih definira kao početne pozicije i onih koji završavaju s elementom "2" koje definira kao krajnje pozicije. Pomoću svih jednadžbi opisanih u šestom poglavlju koje su implementirane u funkciji *Calibration()* i prikupljenih podataka iz *.txt* dokumenata izračunavaju se sve potrebne veličine redom navedenim u nastavku te na kraju i kalibracijski parametri.

Koordinate težišta klastera za kretanje u oba smjera:

$$\begin{aligned} x_{c.g.,cw} &= -106.082\,mm \\ y_{c.g.,cw} &= 64.114\,mm \\ x_{c.g.,ccw} &= -53.704\,mm \\ y_{c.g.,ccw} &= -25.367\,mm \end{aligned} \tag{49}$$

Dobiveni kutevi α i β:

$$\alpha = 2.2887° \tag{50}$$

$$\beta = 0.7503° \tag{51}$$

Polumjer zakrivljenosti putanje *R*:

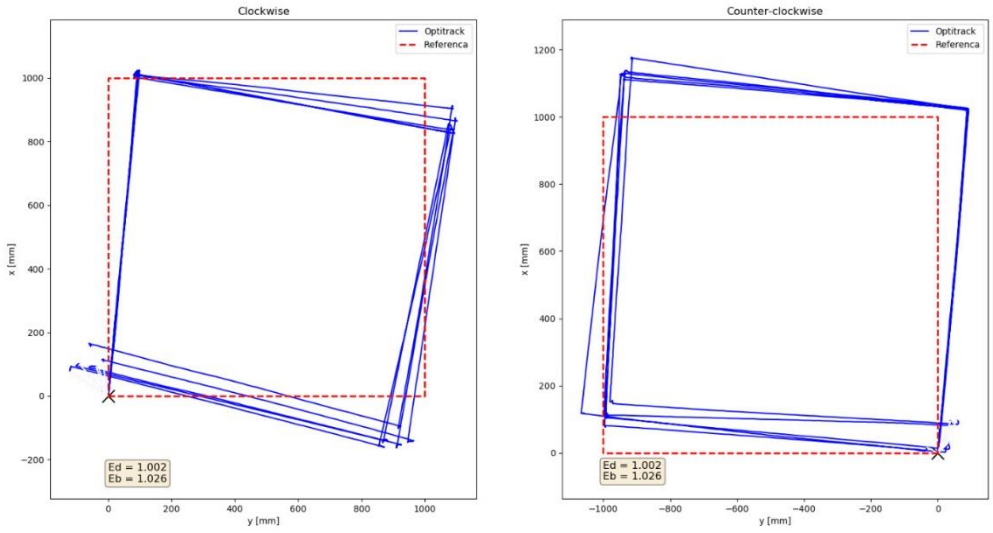
$$R = 76.364\,m \tag{52}$$

Te na kraju, kalibracijski parametri E_d i E_b:

$$E_d = 1.002 \tag{53}$$

$$E_b = 1.026 \tag{54}$$

Da bi se prikazale snimljene kretnje robota i izračunati kalibracijski parametri kako bi korisnik dobio vizualni pregled i povrat informacija, nakon izvršavanja funkcije *Calibration()* u Python skripti se pokreće nova funkcija *Plotting()* koja obavlja navedeno tako da poziva sve pozicije iz *.txt* dokumenata za kretanje u oba smjera i kalibracijske parametre iz funkcije *Calibration()* te ih iscrtava zajedno s referentnom kvadratnom putanjom radi usporedbe. Rezultat navedene funkcije je prikazan na slici 43.



Slika 43. Prikaz stvarnih kvadratnih putanja koje je robot obišao

Kao što je prikazano, s lijeve strane slike 43. se nalazi prikaz kretnje robota (označeno plavom linijom) u smjeru kazaljke na satu dok se s desne strane nalazi prikaz kretnje u smjeru obrnutom od kazaljke na satu. Referentna kvadratna putanja (veličine 1x1 m) je prikazana iscrtkanom crvenom linijom dok su kalibracijski parametri E_b i E_d prikazani na oba dijela u donjem lijevom kutu. Nakon izračuna kalibracijskih parametara slijedi njihova primjena u svrhu približavanja robotove putanje referentnoj. Međutim, implementacija istih i provjera optimizirane kretnje robota nije provedena zbog nemogućnosti unošenja parametara u kontroler zaduženog za *low level* upravljanje.

8. ZAKLJUČAK

Sposobnost kretanja po prostoru je osnovna karakteristika mobilnih robota. Da bi kretanje bilo što kvalitetnije, potrebno je osigurati što točniju estimaciju odometrije. Osigurati istu je izazovan zadatak zbog utjecaja sustavnih i nesustavnih grešaka. Dok su nesustavne greške uzrokovane interakcijom robota s nepredvidivim svojstvima okoliša te se na njih teško utječe, za utjecaj sustavnih grešaka, uzrokovanih konstrukcijskim nesavršenostima i implementacijom komponenata mobilnog robota te s daleko većim utjecajem na estimaciju odometrije, postoji način kompenzacije. Stoga se provodi UMBMark metoda kalibracije koja praćenjem kvadratne trajektorije robota, mjerenjem odstupanja od iste te izračunom kalibracijskih parametara kompenzira najkobnije među njima: nejednak promjer kotača i nesigurnost u razmaku između kotača. Također, implementacija algoritama kretanja pomoću ROS-a i lokalizacija robota pomoću vanjskog vizijskog sustava OptiTrack omogućuju da se proces kalibracije odvija automatski. Automatizacija kalibracije je smislena i korisna odluka zbog jednostavnije primjene i brže izvedivosti iste. Na kraju, primjenom izračunatih kalibracijskih parametara na niskoj razini kontrolera mobilnog robota se očekuje vjerodostojno praćenje referentne trajektorije te podudaranje početne i krajnje pozicije prilikom završetka kretanja.

LITERATURA

[1] https://builtin.com/robotics

[2] https://industrytoday.com/challenges-encountered-in-the-implementation-of-robots/

[3] https://www.automationmag.com/9277-mobile-industrial-robots-adds-autonomous-mobile-robot-for-1000-kg-loads/

[4] Siegwart R., Nourbakhsh I. R. – Introduction to Autonomous Mobile Robots; 2004

[5] http://wiki.ros.org/ROS/Introduction

[6] https://ubuntu.com/robotics/what-is-ros

[7] https://www.udemy.com/course/ros-tutorials/learn/lecture/20523364?start=0#overview

[8] https://gazebosim.org

[9] https://automaticaddison.com

[10] https://optitrack.com

[11] https://github.com/vrpn/vrpn/wiki

[12] Borenstein, J., Feng, L.: UMBmark – A Method for Measuring, Comparing and Correcting Dead-reckoning Errors in Mobile Robots

[13] https://journals.sagepub.com/doi/pdf/10.1177/1729881419839596

[14] https://www.ndt.net/article/ndtce2022/paper/65966_manuscript.pdf

PRILOG: KOD ZA UPRAVLJANJE ROBOTOM

I. Programski kodovi - https://github.com/KL17/Automatic-mobile-robot-calibration