Lokalizacija i navigacija mobilnog robota za dekontaminaciju

Smirčić, Dino

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 702

LOKALIZACIJA I NAVIGACIJA MOBILNOG ROBOTA ZA DEKONTAMINACIJU

Dino Smirčić

Zagreb, veljača 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 702

LOKALIZACIJA I NAVIGACIJA MOBILNOG ROBOTA ZA DEKONTAMINACIJU

Dino Smirčić

Zagreb, veljača 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 30. rujna 2024.

DIPLOMSKI ZADATAK br. 702

Pristupnik:	**Dino Smirčić (0036524601)**
Studij:	Računarstvo
Profil:	Računalno inženjerstvo
Mentor:	izv. prof. dr. sc. Matko Orsag

Zadatak: **Lokalizacija i navigacija mobilnog robota za dekontaminaciju**

Opis zadatka:

U sklopu diplomskog zadatka potrebno je razviti programsko rješenje i pripadajuće elektromehaničke komponente kojim će se omogućiti lokalizacija i navigacija mobilnog robota u nepoznatom okruženju. Potrebno je pregledati i analizirati postojeću literaturu u navedenom području istraživanja. Odabranu metodu potrebno je prilagoditi dostupnom skupu senzora i aktuatora, vodeći računa o kinematičkim i dinamičkim ograničenjima te implementirati na laboratorijskom primjeru mobilnog robota. Sustav je potrebno ispitati u simulacijskom i laboratorijskom okruženju.

Rok za predaju rada: 14. veljače 2025.

Ovaj rad posvećujem babi Milki i majci Ani. Hvala vam na svemu što ste mi pružali tijekom mog studija.

Zahvaljujem svima koji su mi pomogli pri izradi ovoga rada svojim savjetima, a posebno mentoru, prof. dr. sc. Matku Orsagu i asistentu, budućem doktoru znanosti Marku Kozliku.

Sadržaj

1. Uvod

U ovom radu bit će opisane tehnologije i algoritmi potrebni za samostalnu navigaciju mobilnog robota. Cilj ovoga rada je uspješno lokalizirati i navoditi mobilnog robota nepoznatim prostorom koristeći tehnologiju opisanu u nastavku rada. Mobilnom robotu zadaje se krajnja točka u prostoru do koje se mora kretati izbjegavajući prepreke na putu. Mobilni robot opremljen je GPS sustavom za lokalizaciju u prostoru, magnetometrom za određivanje trenutačne putanje u odnosu na magnetski sjever, LiDAR sustavom, kamerom, Intel procesorom i vlastitom odometrijom za određivanje prijeđene udaljenosti i trenutačne brzine. Korištena tehnologija opisana je u poglavlju 2.3.2. Navedenim senzorima bit će potrebno uskladiti mjerenja i osmisliti algoritam koji će ih koristiti kako bi navodio mobilnog robota do krajnje točke. Potrebno je osmisliti i algoritam za učinkovitu segmentaciju okruženja mobilnog robota, točnije algoritam za prepoznavanje prepreka u prostoru s pomoću kamere opisanom u poglavlju 3.3.2. Nakon uspješnog prepoznavanja prepreka, bit će potrebno implementirati algoritam za označavanje prepreka u oblaku točaka (*pointcloudu*) što je opisano u poglavlju 3.5. Kada se odrede prepreke u okruženju mobilnog robota, trebat će implementirati algoritme za navođenje robota do krajnje točke izbjegavajući prepreke koje se nalaze na putu.

2. Pregled znanstvenog područja i tehnologija

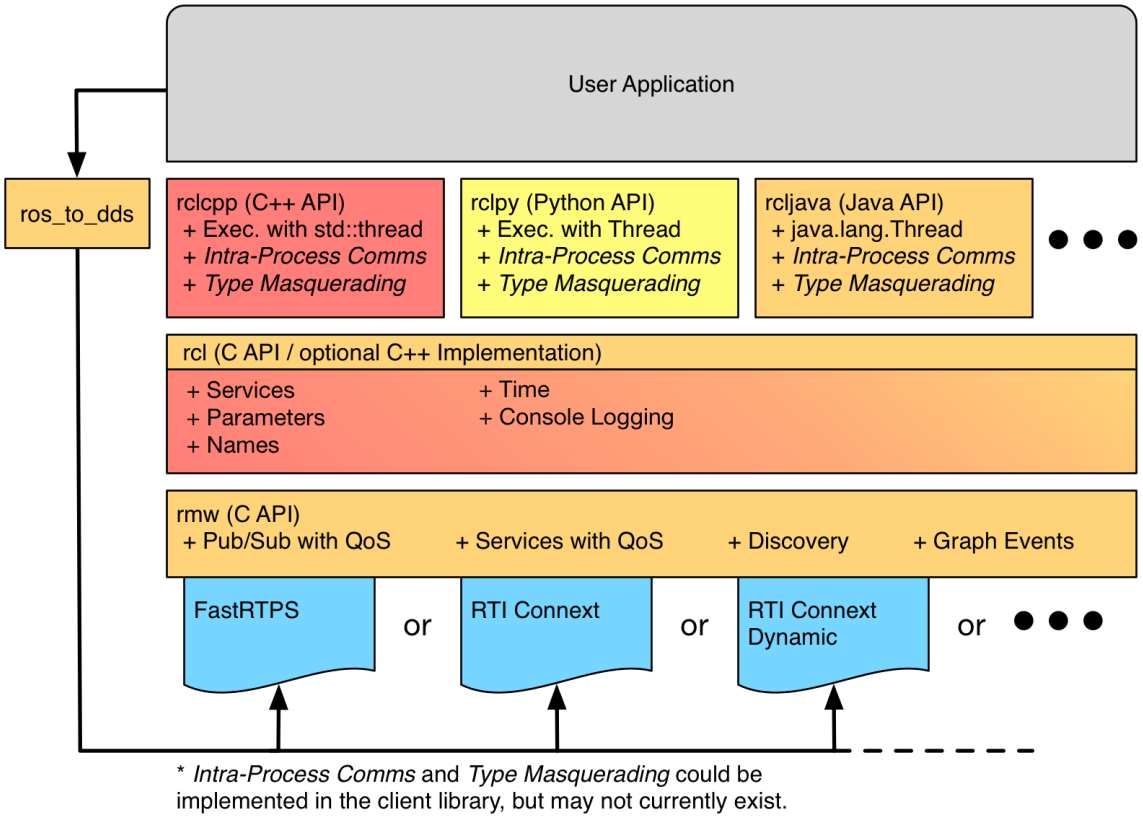
2.1. Pregled znanstvenog područja

Područje autonomne navigacije mobilnih robota je rastuće područje posljednjih nekoliko godina. Područje koje je ključno za ovaj rad je dekontaminacija kontaminiranih područja. Kontaminant kojeg je potrebno ukloniti može biti radioaktivnog ili kemijski otrovnog porijekla. Cilj je ostvariti samostalnu navigaciju mobilnog robota do kontaminanta kako bi ga mogao ukloniti manipulatorskom rukom umjesto čovjeka. Postoji velik broj sličnih grana u kojima se koriste algoritmi i tehnologije opisane u ovom radu, primjerice samoupravljajući automobili poput Teslinog *Autopilota* [1], Amazonovog robota *Kiva* [2] za rad u skladištima i Segwayeve *Nova Carter* [3] univerzalne autonomne platforme.

S pomoću kalibriranih senzorskih očitanja s mobilnog robota stvara se mapa okruženja. Mapom će se služiti algoritmi opisani u 4.2. i 4. kako bi mogli odrediti optimalnu putanju kretanja mobilnog robota u prostoru. Senzorska očitanja se kalibriraju korištenjem transformacija u *ROS-u*, 2.5.1.

2.2. *ROS*

ROS (*Robot Operating System*) je platforma za razvoj programskih rješenja u robotskim sustavima. Modularan je i nadogradiv, zbog čega je vrlo jednostavno koristiti ga u svrhu razvoja programskih rješenja u robotici, [4]. U ovom radu će se koristiti *ROS2*, verzija *humble* koja je nadogradnja na *ROS1*. Arhitektura *ROS2* sustava prikazana je slikom 2.1. Programski kod strukturira se u čvorove (*nodes*), gdje je svakom čvoru dodijeljen zadatak koji izvršava. Čvorovi komuniciraju DDS protokolom, opisanim u 2.3.1.

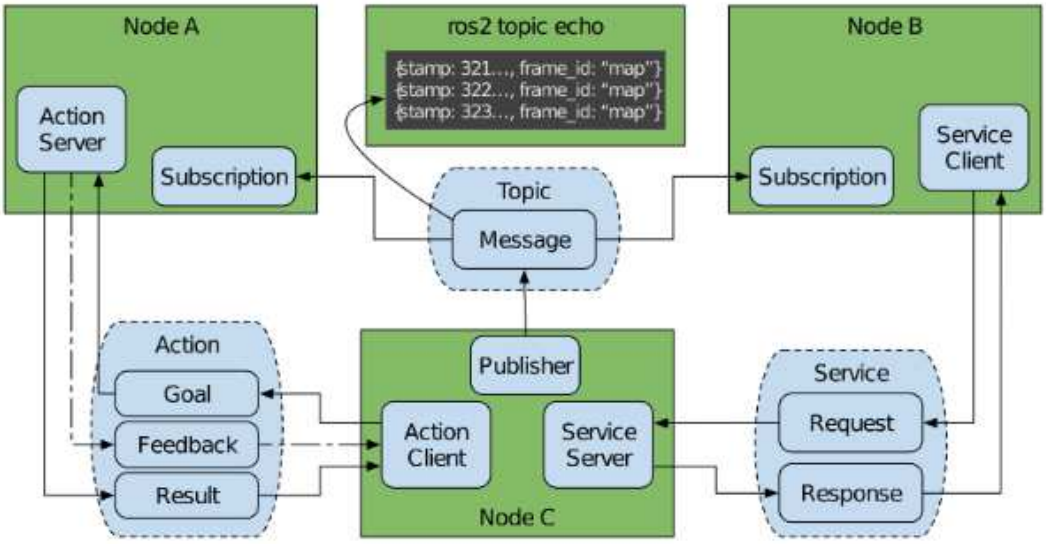


Slika 2.1. Arhitektura ROS2 sustava, izvor: [5].

2.3. Komunikacija čvorova

ROS omogućava podjelu velikog zadatka u manje podzadatke, *„podijeli pa vladaj"* princip. Programski kod koji izvršava podzadatak zove se čvor. Čvorovi najčešće komuniciraju preko *ROS* tema. Komunikacija je asinkrona što svakom čvoru omogućava rad vlastitom frekvencijom, neovisno o radu drugih čvorova. Komunicira se modelom pretplate i objave, (*publish-subscribe*), gdje su poruke koje se objavljuju na temu strogo definiranog tipa, primjer poruke nalazi se u poglavlju 3.3.2. Model pretplate i objave omogućava komunikaciju više čvorova istodobno.

Osim tema, *ROS* nudi i komunikaciju preko servisa, koji su pogodni kada komunikacija mora biti sinkrona. Servisi se temelje na modelu zahtjeva i odgovora (*request-response*), prije izvršavanja zadatka šalje se zahtjev, nakon njegovog izvršavanja šalje se odgovor čime je osigurana sinkronost u izvođenju. Također, *ROS* omogućava da s klijentske strane u servisu izvršavanje ne bude blokirano dok se ne dobije odgovor sa serverske strane. Komunikacijski modeli servisa i tema prikazani su slikom 2.2.



Slika 2.2. Modeli tema i servisa, izvor: [4]

.

2.3.1. *DDS* protokol

DDS (*Data Distribution Service*) omogućava komunikaciju među čvorovima u *ROS* sustavu. Vrlo ga je lako primijeniti u raznim sustavima jer je komunikacija utemeljena na UDP protokolu, zbog čega se može primijeniti i u mikrokontrolerima. Transport poruka *DDS* protokolom ostvaren je na modelu pretplate i objave poruka (*publish-subscribe*). Moguće je i otkriti druge čvorove na mreži i izravno komunicirati bez glavnog čvora *ROS mastera*, kao što je bilo potrebno u sustavima koji koriste *ROS1*. Vrlo važna funkcionalnost uvedena je s mogućnošću definiranja kvalitete usluge (*QoS, Quality of Service*). Moguće je koristiti *DDS* za sve od sustava za rad u stvarnom vremenu do sustava u kojima je bitna tolerancija na dugoročne ispade čvorova promjenom *QoS* parametra, [4]. Prema [6], parametri kojima se specificira transport poruka su:

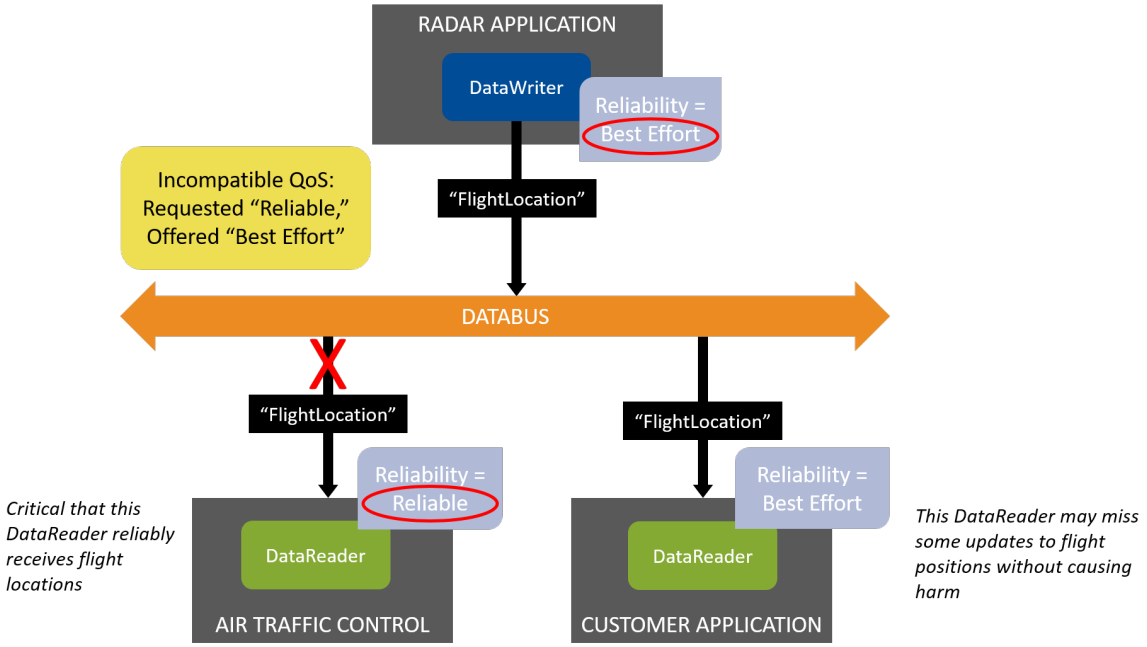
- pouzdanost (*reliability*) - treba li se dolazak svake poruke garantirati ili je dozvoljeno propustiti neke poruke
- povijest (*history*) - koliko pristiglih poruka je potrebno čuvati u međuspremniku
- memorija (*resource limits*) - koliko je memorijsko ograničenje reda za primanje i slanje poruka
- izdržljivost (*durability*) - trebaju li se poruke za slanje spremati prije nego što su

čvorovi koji ih čitaju u operativnom stanju

- rok isporuke (*deadline*) - jesu li podaci dovoljno brzo isporučeni od pošiljatelja do primatelja?

Primjer parametara u DDS komunikaciji

Parametar pouzdanosti može biti ili „pouzdan" (*reliable*) ili „najbolji mogući" (*best-effort*). Pošiljatelj nudi razinu pouzdanosti dok primatelj zahtijeva određenu razinu pouzdanosti. Ako je razina pouzdanosti usluge koju pošiljatelj nudi veća ili jednaka onoj koju primatelj zahtijeva, parametri pouzdanosti će se poklapati, [6]. Primjer u kojem se razine pouzdanosti ne poklapaju prikazan je slikom 2.3. Ako se parametri razine pouzdanosti ne poklapaju, nije moguće uspostaviti komunikaciju, točnije, poruke se odbacuju.



Slika 2.3. Pošiljatelj je konfigurirao razinu pouzdanosti na manju od zahtijevane, izvor [6]

.

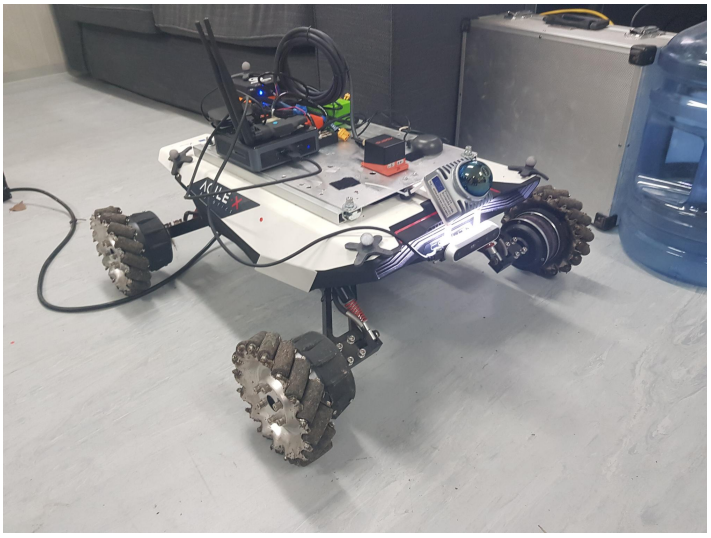
Razina pouzdanosti „najbolja moguća" označava da će se poruke slati s vrlo malim vremenskim razmakom među njima, periodički. Ako se poruka ne primi na drugoj strani, važnije je osigurati da se primi iduća poruka nego ponovno poslati izgubljenu poruku s pošiljateljeve strane.

Razina pouzdanosti „pouzdana" označava da, ako se poruka ne primi na primateljskoj strani, ponovno će se pokušati poslati s pošiljateljeve strane, uz neka ograničenja. Ako je parametar povijesti „zadrži sve poruke" (*keep all history*), sve izgubljene poruke

će se ponovno pokušati poslati primatelju, a ako je parametar povijesti „zadrži zadnju" (*keep last*) ponovno će se poslati samo definiranih *N* poruka primatelju.

2.3.2. Mobilni robot

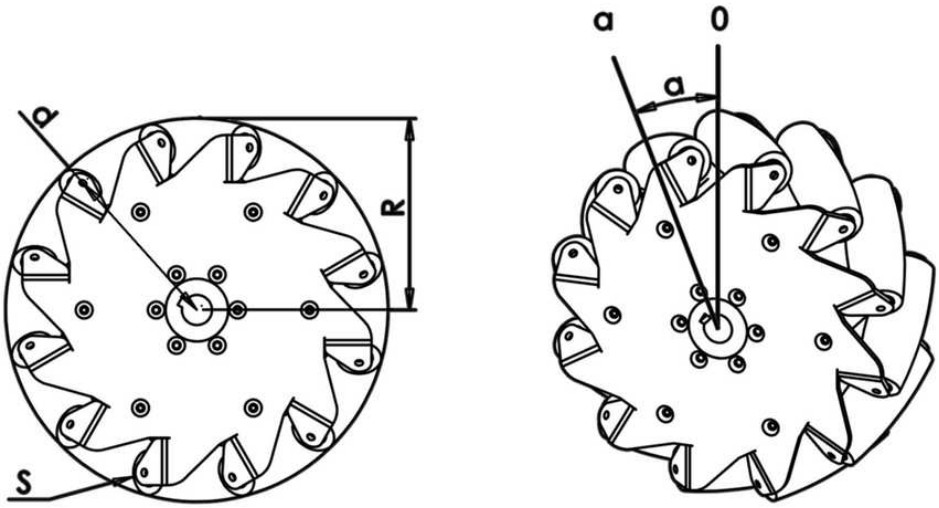
Mobilni robot koji se koristio za lokalizaciju i autonomnu navigaciju kroz prostor je proizvođača *Agilex*, model *Scout Mini 2.0*, prikazan slikom 2.4.



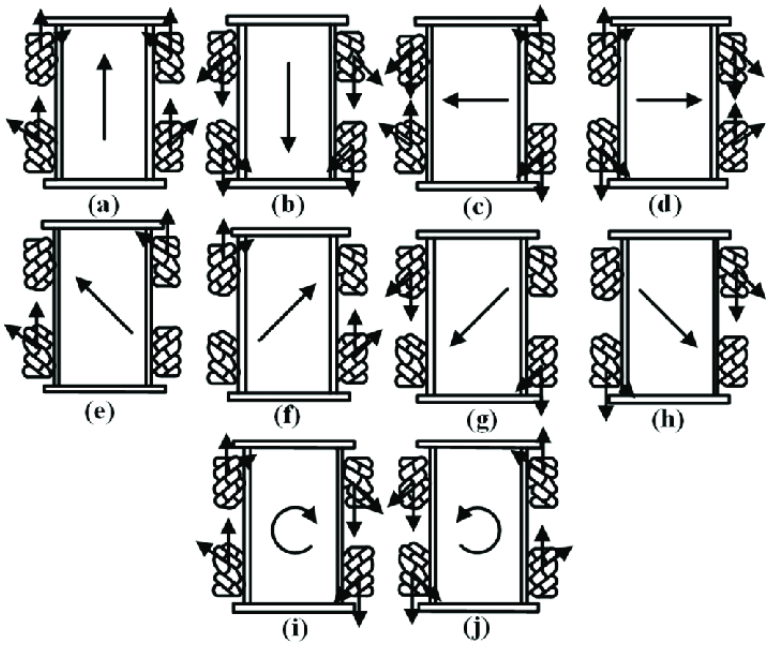
Slika 2.4. *Scout Mini 2.0.*

Opremljen je svestranim *mecanum omnidirecitonal* kotačima koji omogućavaju gibanje u svim smjerovima, prikazanim slikom 2.5. Dijagram gibanja prikazan je slikom 2.6. Svesmjerni kotači su optimalan izbor za autonomnu navigaciju uskim prostorima jer robotu omogućavaju horizontalnu translaciju i okretanje u mjestu. Nedostatak svesmjernih kotača je gibanje uzbrdo. U usporedbi s običnim kotačima, svesmjerni kotači imaju pokretne dijelove zbog kojih je trenje s podlogom smanjeno, čime je otežano gibanje na kosini. Robot ima dva načina upravljanja, ručno upravljanje putem upravljača i upravljanje putem *ROS* čvorova.

Svaki kotač na mobilnom robotu ima svoj elektromotor s kojim ugrađeno računalo mobilnog robota komunicira preko *CANBus* protokola. Kontrolom okretanja pojedinog elektromotora određuje se smjer gibanja mobilnog robota, prikazano slikom 2.6.



Slika 2.5. Prikaz svesmjernih (*mecanum omnidirectional*) kotača, izvor: [7]



Slika 2.6. Dijagram kretanja svesmjernim kotačima, izvor: [8]

2.4. Intel *Nuc* računalo

Osim ugrađenog računala unutar mobilnog robota, koristi se i vanjsko Intelovo računalo *Nuc* s operacijskim sustavom *Linux* na kojem se izvršavaju *ROS* paketi opisani u 3.3.2. i 3.5. Njegova zadaća je komuniciranje s vanjskim senzorima opisanim u poglavlju 2.5.1.

2.5. Senzori i detekcija

2.5.1. LiDAR

Kako bi se precizno mogle odrediti udaljenosti do predmeta u okruženju mobilnog robota, koristi se LiDAR (*Light Detection and Ranging*) proizvođača *Livox*, model *MID 360*, prikazan slikom 2.7. LiDAR s pomoću reflektiranih laserskih zraka precizno može odrediti udaljenost do svake točke u okruženju. U sklopu ovog rada, LiDAR je korišten zajedno s kamerom za lokalizaciju i stvaranje mape prostora. Kako bi mjerenja bila što točnija, potrebno je napraviti kalibraciju očitanja LiDAR-a i kamere.

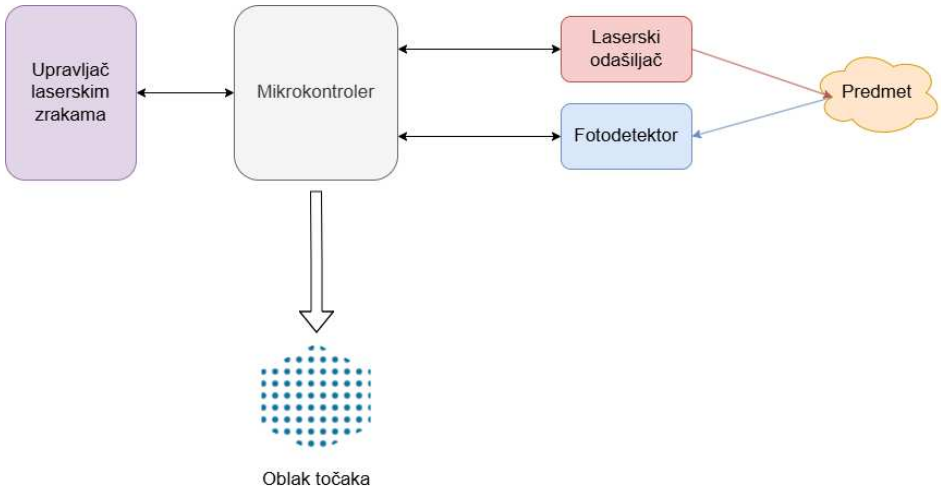
Slika 2.7. *Livox MID 360* LiDAR, izvor [9].

Za potrebe ovog rada dovoljno je ručno izmjeriti gdje se LiDAR i kamera nalaze u odnosu na *IMU* mobilnog robota i pri pokretanju čvorova za segmentaciju prostora izvršiti naredbe za transformaciju koordinatnih sustava LiDAR-a i kamere u odnosu na koordinatni sustav *IMU-a*. LiDAR je na robotu postavljen pod kutom od 45° kako bi istovremeno mogao snimati područje izravno ispred mobilnog robota i područje iznad mobilnog robota. Prilikom obrade oblaka točaka očitanja se transformiraju u horizontalnu ravninu. LiDAR tehnologija se koristi kako bi se dobile točne udaljenosti do svih točaka u okruženju. Svaki LiDAR emitira laserske zrake unutar definiranog kuta pregleda (*Field of View, FoV*). LiDARi koji snimaju prostor u dvije dimenzije odašilju samo jednu lasersku zraku u horizontalnom smjeru, dok LiDARi kao *Livox Mid 360* zrake koje se reflektiraju od *MEMS* zrcala u svim smjerovima kako bi se dobila slika prostora u tri

dimenzije. Laserske zrake reflektiraju se od predmeta i vraćaju do senzora u LiDAR-u, koji za svaku odašiljanu zraku mjeri vrijeme putovanja *(ToF, Time of Flight)*, [10]. LiDAR koji se koristio u praktičnom dijelu ovoga rada ima kut pregleda od 7° do 59° vertikalno, u koničnom obliku. Snima okruženje u 360° horizontalno, a domet mu je od 10 cm do 40 m. Udaljenost do točke na predmetu računa se s pomoću formule 2.1, gdje je c brzina svjetlosti, a t (ToF) izmjereno vrijeme u sekundama od početka emitiranja svjetlosnog signala do primitka svjetlosnog signala.

$$d = \frac{c \cdot t}{2} \tag{2.1}$$

Fotodetektori pretvaraju izmjereni intenzitet svjetlosti u električni signal koji se kasnije obrađuje kako bi se uklonile smetnje koje se mogu javiti prilikom refleksije svjetlosti zbog različitih materijala od kojih se svjetlost reflektira. U konačnici, LiDAR vraća sliku prostora u obliku oblaka točaka (*pointcloud*). Proces pretvorbe okruženja u niz točaka sa izmjerenim udaljenostima prikazan je slikom 2.8.



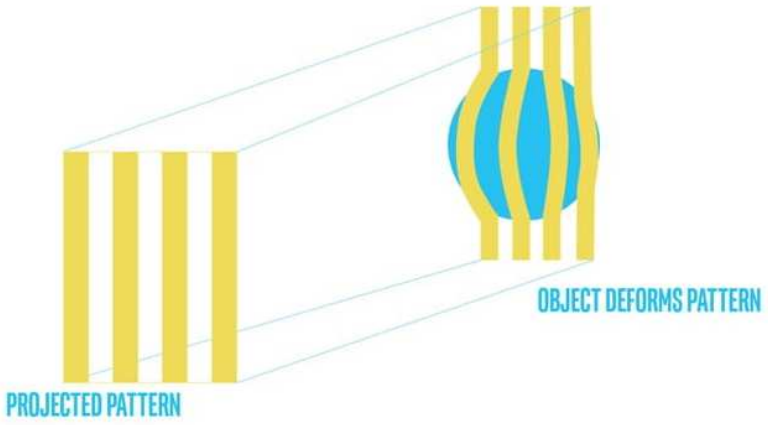
Slika 2.8. Pretvorba predmeta iz okruženja u oblak točaka.

2.5.2. Kamera

Korištenjem LiDAR-a se efikasno može stvoriti mapa okruženja mobilnog robota, ali njegovo ograničenje jest nemogućnost klasifikacije predmeta. U tu svrhu se uz LiDAR koristi i Intel *Realsense* kamera. Osim što snima sliku u RGB formatu, može mjeriti udaljenosti do predmeta u blizini. Uz LiDAR služi za stvaranje mape prostora u kojem se nalazi mobilni robot.

U svrhu klasifikacije prepreka i segmentacije okruženja mobilnog robota, koristi se Intel *Realsense D415* kamera prikazana slikom 2.9. Kamera odašilje poznate uzorke infracrvenim zrakama i razlikom između očekivane i izmjerene veličine uzorka i očekivanog i izmjerenog oblika uzorka, računa udaljenost do predmeta. Opisanim načinom može se precizno odrediti udaljenost do predmeta u zatvorenom prostoru, na manjim udaljenostima. Zbog toga što i Sunce zrači u infracrvenom spektru, pojavljuje se šum u mjerenju udaljenosti u otvorenom prostoru, kao što je opisano u radu [11].

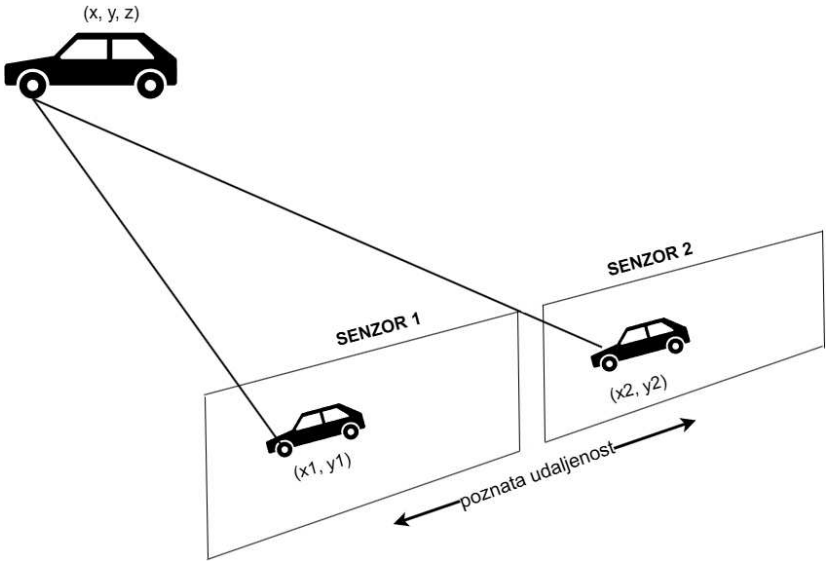
Slika 2.9. *Intel D415* kamera, izvor [12]



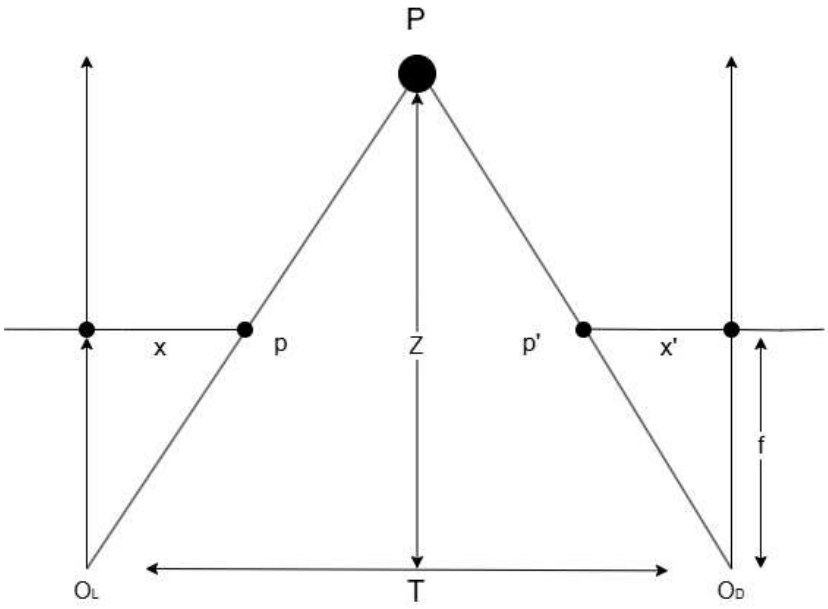
Slika 2.10. Projekcija poznatih laserskih uzoraka s kamere do predmeta, izvor: [13].

Nadogradnja na opisani pristup je uvođenje stereo senzora. Na isti način na koji ljudi percipiraju dubinu slike, s pomoću dva oka, Intelove *D400* kamere računaju udaljenost do predmeta. Na svakoj kameri nalaze se senzori za dubinu koji služe za računanje udaljenosti, opisano u nastavku, jedan odašiljač infracrvenih zraka i dodatna RGB kamera. Odašiljanjem infracrvenih zraka preciznije se mogu odrediti udaljenosti do predmeta s

malo vizualnih značajki, (npr. do bijelog zida, ravnog stola i sl.). Udaljenost između lijeve i desne kamere za dubinu je precizno izmjerena. Svaka kamera vidi prostor iz svog kuta. Procesorska jedinica unutar kamere uzima obje slike i s pomoću poznatih značajki sa slika objedinjuje dvije slike u jednu, izračunavajući udaljenost do predmeta prema formuli 2.3



Slika 2.11. Računanje udaljenosti do predmeta *D400* serijom.



Slika 2.12. Točka u prostoru snimljena stereo kamerom.

Z predstavlja udaljenost do predmeta, f predstavlja žarišnu duljinu leće, d predstavlja razliku u položaju snimljene točke u prostoru između slike na lijevoj i desnoj leći $(x-x')$, a T je unaprijed izmjerena udaljenost između lijeve i desne leće na stereo kameri [14].

Oznakama p i p' označene su točke u kojima reflektirana svjetlosna zraka ulazi u lijevu i desnu leću kamere, a O_L i O_D predstavljaju optička središta lijeve i desne leće kamere, točnije, mjesta u kojima se prikuplja reflektirana svjetlost iz okruženja. Dijagramom 2.12. je prikazan način na koji kamera snima okruženje. Postavljanjem jednadžbe za sličnost trokuta 2.2 dobiva se sljedeće:

$$\frac{Z}{T} = \frac{f}{d}, \tag{2.2}$$

iz čega se lako može izraziti Z i izračunati udaljenost do točke formulom 2.3

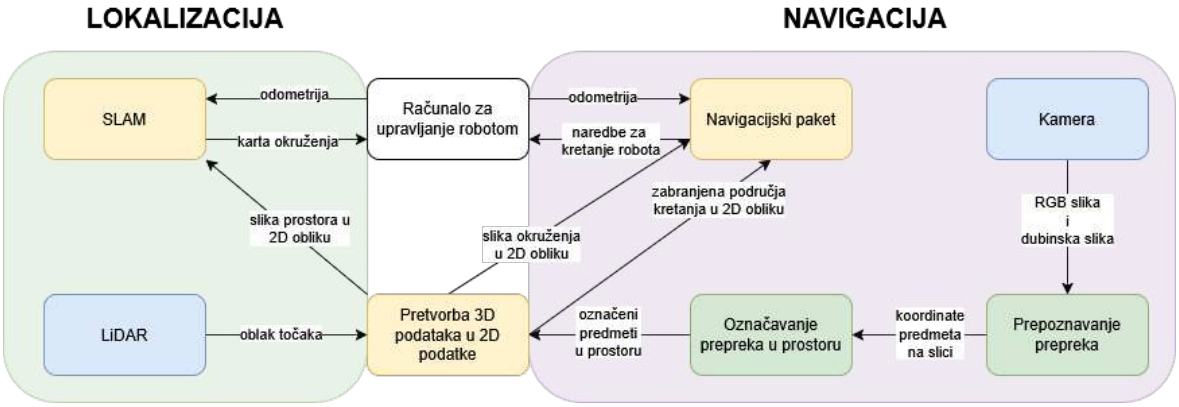
$$Z = f\frac{T}{d} \tag{2.3}$$

3. Postavljanje okruženja

U ovom poglavlju bit će opisani koraci koje je bilo potrebno izvršiti za uspješnu autonomnu navigaciju mobilnog robota kroz prostor. Prvo će biti opisana fizička priprema u odjeljku 3.2., zatim će biti opisana razvijena programska rješenja u odjeljcima 3.3., 3.4., 3.3.2. i 3.5.

3.1. Programska struktura praktičnog dijela rada

Za bolje daljnje razumijevanje opisana je struktura programskih čvorova koji su se koristili za ostvarivanje praktičnog dijela rada. Struktura je prikazana slikom 3.1.

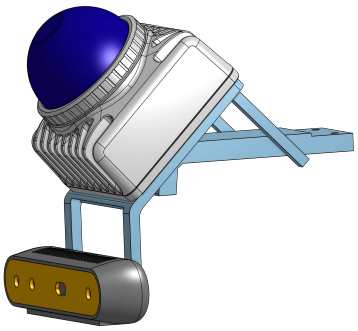


Slika 3.1. Komunikacija programskih čvorova i njihove odgovornosti.

Plavom bojom prikazani su senzori, zelenom bojom prikazani su programski čvorovi koji su razvijeni u sklopu ovoga rada, a žutom bojom su prikazani postojeći programski čvorovi koje je bilo potrebno prilagoditi za ostvarivanje zadanog cilja. Struktura je podijeljena u dvije velike cjeline, jedna kojoj je zadatak izvođenje algoritama za lokalizaciju mobilnog robota, a druga kojoj je cilj rezultatima lokalizacije i čvorovima razvijenim za segmentaciju prostora pronaći optimalnu putanju od ishodišta do odredišta. Svi čvorovi komuniciraju preko *ROS* tema koje će biti opisane u nastavku.

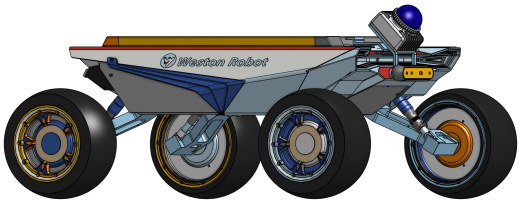
3.2. Dizajniranje nosača za LiDAR i kameru

Kako bi LiDAR i kamera mogli efikasno stvoriti mapu prostora u kojemu se nalazi mobilni robot, bilo je potrebno osmisliti nosač na kojemu će se nalaziti. Učvršćivanjem na nosač, LiDAR i kamera se nalaze u poznatoj prostornoj ovisnosti i moguće je odrediti transformaciju iz vidnog polja LiDAR-a u vidno polje kamere i obrnuto. Idealan položaj LiDAR-a je 45° u odnosu na horizontalnu ravninu, 30 *cm* vertikalno udaljenim od tla. Kada ne bi bilo nagiba u odnosu na vodoravnu ravninu, točnije, kada bi LiDAR bio postavljen paralelno s tlom, ne bi mogao snimati tlo izravno ispred mobilnog robota. Postavljanjem LiDAR-a u navedeni položaj, nastaje područje u duljini od 30 *cm* ispred mobilnog robota za koje se neće moći stvoriti mapa, ali je uzeto u obzir da LiDAR nije precizan pri skeniranju predmeta na manjim udaljenostima. Zamišljeno je da se kamera nalazi ispod LiDAR-a, na prednjoj strani robota. Kamera snima prostor i stvara mapu prostora koji se nalazi izravno ispred mobilnog robota pokrivajući i prostor koji LiDAR ne može pokriti zbog opisane orijentacije. Nosač zajedno s LiDAR-om i kamerom prikazan je na slici 3.2. Nosač je dizajniran u alatu *Onshape* koristeći studentsku licencu i isprintan korištenjem 3D printera. Osim navedenih ograničenja u pozicioniranju senzora, bilo je potrebno osmisliti na koji način se nosač pričvršćuje na mobilnog robota imajući u vidu da mora moći izdržati sile koje će biti prisutne pri kretanju mobilnog robota. Model nosača pričvršćenog na mobilnog robota prikazan je slikom 3.4. u obliku 3D modela i slikom 3.3. u fizičkoj izvedbi na mobilnom robotu.



Slika 3.2. Model nosača kamere i LiDAR-a.

Slika 3.3. Fizička izvedba nosača LiDAR-a i kamere na mobilnom robotu.



Slika 3.4. Model nosača pričvršćenog na mobilnog robota *Scout Mini 2.0*.

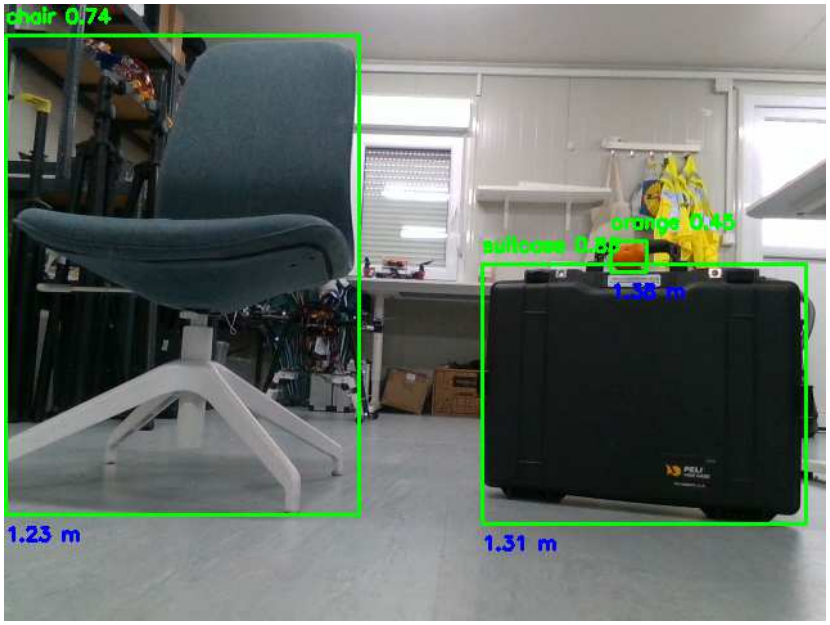
3.3. Prepoznavanje prepreka

Kako bi se prepreke na putu mogle prepoznati, koristi se model za prepoznavanje predmeta *YOLO v8-seg*. *YOLO v8-seg* osim prepoznavanja prepreka omogućava i segmentiranje prepreke u prostoru, što znači da može odrediti granice predmeta, opisano u 3.3.2. Uz prepoznavanje prepreka, mjere se i udaljenosti do njih, koristeći senzore s *Realsense* kamere opisane u 2.5.1.

3.3.1. *YOLO* model

Za svrhe ovoga rada odabran je *YOLO* model, [15] zbog svoje iznimne brzine i preciznosti prikladan je za primjenu u navigaciji mobilnog robota. Za precizno određivanje početnih i krajnjih koordinata predmeta odabran je model *YOLOv8-seg*, koji će detaljnije biti opisan u poglavlju 3.3.2. Na mobilnom robotu nema grafičke kartice, zbog čega je važno odabrati model koji može klasificirati objekte koristeći procesor. Koristi se najma-

nja *YOLOv8* neuronska mreža, (*YOLOv8 nano*). Prepoznavanje se izvršava u nekoliko koraka. Kada se slika podijeli na mrežu ćelija jednakih veličina, izvršava se prepoznavanje predmeta unutar svake ćelije. Svakom predmetu se dodjeljuju početne i krajnje koordinate pravokutnika koji ga opisuje i razina sigurnosti da pojedini predmet pripada određenoj klasi. Zbog mogućih preklapanja predmeta unutar iste ćelije, u konačnici se uzima predmet s najvećom razinom sigurnosti u ćeliji. Rezultati prepoznavanja svake ćelije se u završnom koraku spajaju u cjelovitu sliku s prepoznatim predmetima na slici. Uz snimanje slike u RGB formatu, *Realsense* kamera omogućava i mjerenje udaljenosti do svakog piksela slike na način opisan u 2.5.1. Klasifikacija predmeta s pripadnim udaljenostima prikazana je na slici 3.5.



Slika 3.5. Klasifikacija i izmjerene udaljenosti do predmeta.

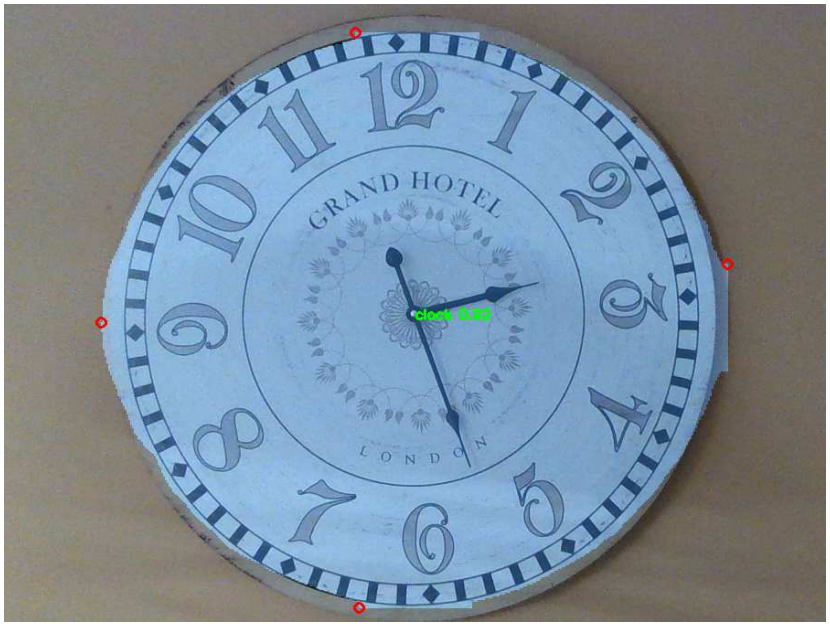
3.3.2. *ROS2* programski paket za prepoznavanje predmeta

Za svrhu ovoga rada osmišljen je i implementiran *ROS2* čvor za obradu slike u programskom jeziku *Python*. Kada se pokrene čvor koji upravlja *Realsense* kamerom, kamera objavljuje snimljenu sliku na *ROS* temu `/camera/camera/color/image_raw`. Objavljena slika je u RGB formatu. Slika se predaje *YOLO* modelu koji vraća listu prepoznatih predmeta na slici u poruci *BoundingBox.msg* osmišljenoj za prijenos podataka potrebnih za daljnji rad čvorova, prikazanoj u nastavku.

```
int32[2]    leftmost
int32[2]    rightmost
int32[2]    topmost
int32[2]    bottommost
int32       center_x
int32       center_y
float32     confidence
float32     depth
string      class_label
```

Poruka se objavljuje na *ROS* temu `/yolo/detect/bounding_box`. Svaka poruka sadrži koordinate prvog i posljednjeg piksela koji se pojavljuju s lijeva na desno (*leftmost* i *rightmost*) i koordinate prvog i posljednjeg piksela koji se pojavljuju od dolje prema gore (*bottommost* i *topmost*), razinu sigurnosti da se prepoznati objekt nalazi unutar navedenih koordinata, središnju x i y točku izračunatu po formulama 3.1, 3.2, 3.3, udaljenost do predmeta i ime klase predmeta. Kao udaljenost do predmeta uzima se najbliža točka od opisanih točaka. Razlog tome je što se time izbjegava mogućnost da se točka prepoznatog predmeta nalazi izvan granica predmeta, primjerice, kada bi predmet bio prstenastog oblika, a središnja točka bi se uzimala kao udaljenost do predmeta, izmjerena udaljenost ne bi bila točna. Odabran je model *YOLOv8-seg* zbog mogućnosti segmentacije slike. Segmentacija slike omogućava pronalazak točnih koordinata koje omeđuju predmet, dok modeli bez segmentacije vraćaju koordinate pravokutnika koje omeđuju predmet. Na slici 3.6. prikazan je rezultat čvora za segmentaciju nad slikom zidnog sata. *YOLO* model će prepoznati na kojim se pikselima u slici nalazi sat, nakon čega se računa koji pikseli se nalaze krajnje lijevo, desno, gore, dolje i u sredini. Na slici 3.6. crvenim

kružnicama su označene točke koje se u navedenoj poruci *BoundingBox.msg* objavljuju na *ROS* temu `/yolo/detect/bounding_box`. Kako su se koristili podaci iz opisane poruke bit će opisano u poglavlju 3.5.



Slika 3.6. Slika segmentiranog sata i točke koje ga omeđuju.

Formule za računanje središnje x i y koordinate u prostoru:

$$c_x = \frac{m_{10}}{m_{00}}, \tag{3.1}$$

$$c_y = \frac{m_{01}}{m_{00}}, \tag{3.2}$$

$$m_{ij} = \sum_x \sum_y I(x,y) \cdot x^j \cdot y^i. \tag{3.3}$$

- $I(x,y)$ - intenzitet piksela na koordinatama (x, y), za binarne slike 0 ili 1
- x - x koordinata piksela
- y - y koordinata piksela.

Navedene formule proizlaze iz fizike, a koriste se za računanje središta mase predmeta, prikazano jednadžbom 3.4, gdje je x_{cm} središte mase, umnožak $m_i \cdot x_i$ moment

pojedinog tijela, a M ukupna masa svih tijela za koja se računa središte mase. Pri računanju geometrijskog središta predmeta za moment pojedinog tijela (piksela) može se uzeti samo njegova udaljenost od središta x_i, jer se svi pikseli smatraju jednako važnima za računanje središta, točnije $m_i = 1$.

$$x_{cm} = \sum_i \frac{m_i \cdot x_i}{M} \tag{3.4}$$

Formulama 3.1, 3.2 i 3.3 kao argument se predaje binarna maska predmeta, koja se dobije s pomoću modela za segmentaciju slike. Kada je slika u binarnom obliku, m_{00} predstavlja zbroj, odnosno ukupnu površinu predmeta u pikselima, a m_{10} i m_{01} predstavljaju težinski zbroj koordinata u x i y smjeru [16]. Dijeljenjem s m_{00} dobiva se težinski prosjek za x, odnosno y koordinatu, čime se ujedno dobiva i središnja točka predmeta sa slike. Navedenim postupkom se osigurava da se dobiveno središte neće nalaziti izvan granica predmeta. Kada se ne bi koristio segmentacijski model prepoznavanja predmeta i kada ne bi bilo moguće odrediti točne krajnje koordinate predmeta, problem pri određivanju udaljenosti bi se javljao kod predmeta koji ravnomjerno ne zauzimaju prostor unutar pravokutnika koji ih omeđuje. Geometrijsko središte predmeta nalazilo bi se izvan granica predmeta, a udaljenost koja bi se izmjerila ne bi predstavljala točnu udaljenost do predmeta. Bez točne udaljenosti nije moguće odrediti položaj predmeta u prostoru.

3.4. Kalibracija kamere

Kako bi se dobila što preciznija transformacija između koordinatnih sustava kamere i LiDAR-a, prvo je potrebno odrediti intrinzične parametre leće na kameri, odnosno svojstva kamere, kako je opisano u [17]. Svojstva kamere se mogu prikazati matricom K (3.5).

$$\begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

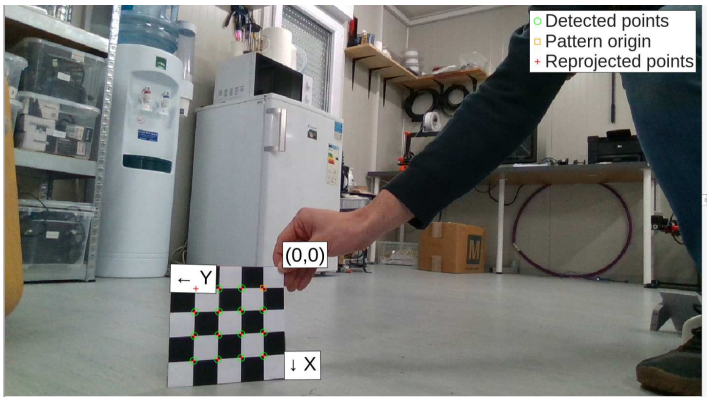
Matrica K, 3.5 sadrži sljedeće vrijednosti;

- f_x i f_y - predstavljaju žarišnu duljinu leće izraženom u broju piksela
- s - predstavlja distorziju piksela kojeg vidi kamera, računa se formulom $f_x \cdot tan\alpha$, gdje je α kut distorzije piksela koja se pojavljuje zbog nesavršene geometrije leće na kameri
- c_x i c_y - predstavljaju žarišnu točku leće.

Određivanje svojstvenih parametara kamere napravljeno je u alatu *MATLAB* koristeći šahovnicu s poznatim dimenzijama kvadrata. Korišteni alat za kalibraciju, *Camera Calibrator*, opisan je u [18]. Kako bi se točno odredili parametri, u alat je potrebno učitati 20 slika šahovnice u različitim orijentacijama. Jedna od slika s pomoću koje se provodila kalibracija prikazana je u 3.7.

Matrica K dobivena navedenim alatom za *Realsense* kameru navedena je u 3.6

$$\begin{bmatrix} 898.66 & 0 & 642.39 \\ 0 & 898.1 & 359.49 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$



Slika 3.7. Računanje svojstava *Realsense* kamere

.

3.5. Segmentacija trodimenzionalnog prostora

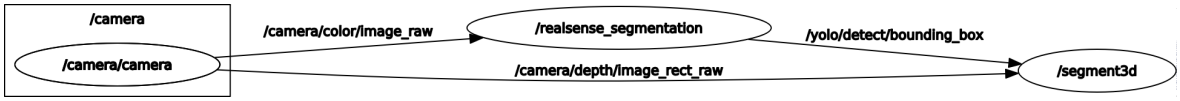
Kako bi se odredila zabranjena područja kretanja za mobilnog robota, potrebno je odrediti gdje se nalaze prepreke u trodimenzionalnom prostoru. Nakon što su određene koordinate prepreke na slici u dvodimenzionalnom prostoru, potrebno ih je pretvoriti u prostorne koordinate. Navedeni problem rješava osmišljeni *ROS* paket *pointcloud segmentation*. Dobivajući poruke opisane u poglavlju 3.3. mogu se izračunati trodimenzionalne koordinate predmeta formulama 3.7 i 3.8 prema [19]. Formule za pretvorbu dvodimenzionalnih u trodimenzionalne koordinate:

$$x_{3D} = d \cdot \frac{x_{2D} - c_x}{f_x}, \tag{3.7}$$

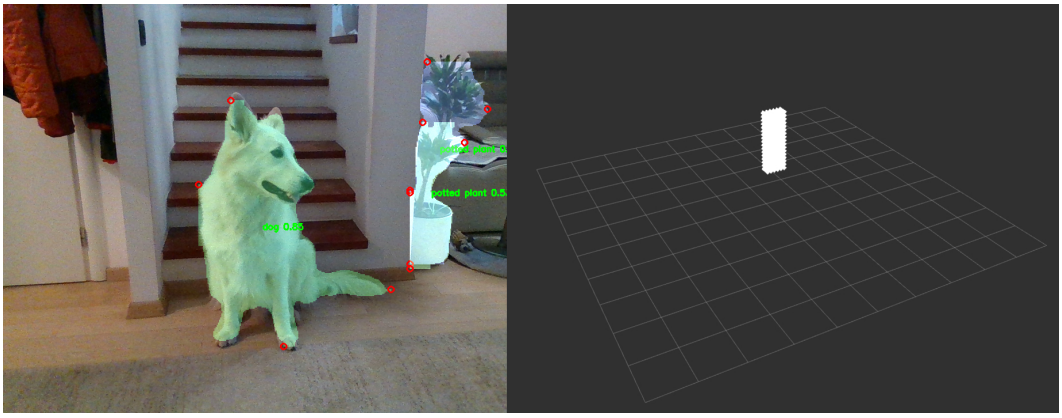
$$y_{3D} = d \cdot \frac{y_{2D} - c_y}{f_y}, \tag{3.8}$$

gdje su c_x, f_x, c_y i f_y svojstveni parametri kamere opisani u poglavlju 3.4. Parametar d predstavlja izmjerenu udaljenost do predmeta. Udaljenost do određenog piksela mjeri se pretplatom na *ROS* temu `/camera/camera/depth/image_rect_raw`, koja je prikazana slikom 3.8., a pretvorba dvodimenzionalnih koordinata u trodimenzionalne prikazana je na slici 3.9. Na navedenoj slici, s desne strane prikazan je položaj psa u prostoru, jer se sve ostale prepoznate klase predmeta ignoriraju. Kako kamera ne može vidjeti najdalju točku predmeta, za različite klase predmeta definirane su najveće moguće dubine predmeta, opisane u 3.5.1.

Komunikacija *ROS* čvorova za prepoznavanje predmeta u prostoru prikazana je slikom 3.8.

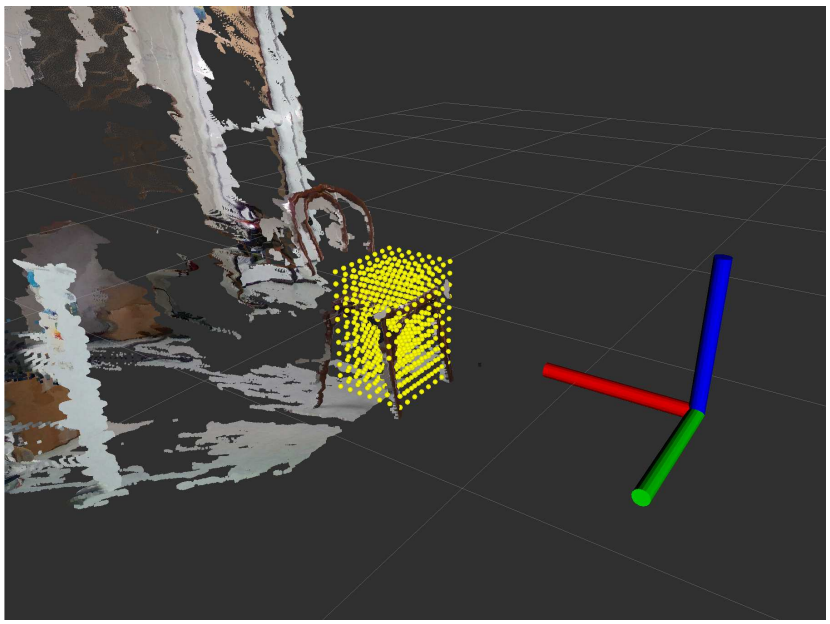


Slika 3.8. Komunikacija čvorova putem *ROS* tema za prepoznavanje predmeta u prostoru.



Slika 3.9. S lijeve strane prikazana je segmentacija u dvodimenzionalnom prostoru, s desne strane je prikazana segmentacija u trodimenzionalnom prostoru u alatu *RViz*.

Primjer segmentacije trodimenzionalnog prostora pri snimanju prostora izravno sa mobilnog robota prikazan je slikom 3.10. Zelenom, crvenom i plavom osi prikazano je mjesto na kojem se nalazi mobilni robot, ispred njega se nalazi drvena stolica oko koje je prikazano zabranjeno područje kretanja žutom bojom, u obliku oblaka točaka.



Slika 3.10. Prikaz zabranjenog prostora oko predmeta u obliku oblaka točaka.

3.5.1. Modularnost *ROS* paketa za segmentaciju prostora

Zbog brzine izvođenja prepoznavanja predmeta, algoritam je ograničen na prepoznavanje predmeta koji su važni za navigaciju mobilnog robota. Predmeti koji nisu navedeni u kodu, ignoriraju se prilikom prepoznavanja. Kako bi se program lako mogao nadograditi, korišten je oblikovni obrazac „strategija." Definirana je izvorna klasa za sve predmete, koja zahtijeva implementaciju metode koja vraća dubinu, odnosno najveću moguću udaljenost od najbliže do najdalje točke predmeta.

```
class AbstractObject(ABC):
    @abstractmethod
    def get_depth(self) -> float:
        return -1.0
```

Konkretna klasa predmeta mora implementirati metodu *get _depth()*, primjer je prikazan u nastavku.

```
class Person(AbstractObject):
    def get_depth(self):
        return 0.5
```

Bude li potrebno dodavati nove klase predmeta pri prepoznavanju, potrebno je napraviti novu klasu koja nasljeđuje *AbstractObject* i implementirati metodu *get _depth()* koja vraća najveću moguću dubinu predmeta. Kako bi se glavni kod mogao koristiti bez velikih promjena, implementirana je klasa *GetObject* koja je odgovorna za stvaranje objekata predmeta koji su važni za navigaciju robota.

```
class GetObject():
    def createObject(self, class_name: str) -> AbstractObject:
        if class_name == "person":
            return Person()
        elif class_name == "dog":
            return Dog()
        else:
            return None
```

Ovime je uklonjena odgovornost glavnog programa od poznavanja točne klase predmeta koji je rezultat prepoznavanja. Glavni program poziva metodu *createObject()* predavanjem argumenta s imenom klase predmeta.

```
objectClass = GetObject().createObject(bbox.class_label)
objectDepth: float = objectClass.get_depth()
```

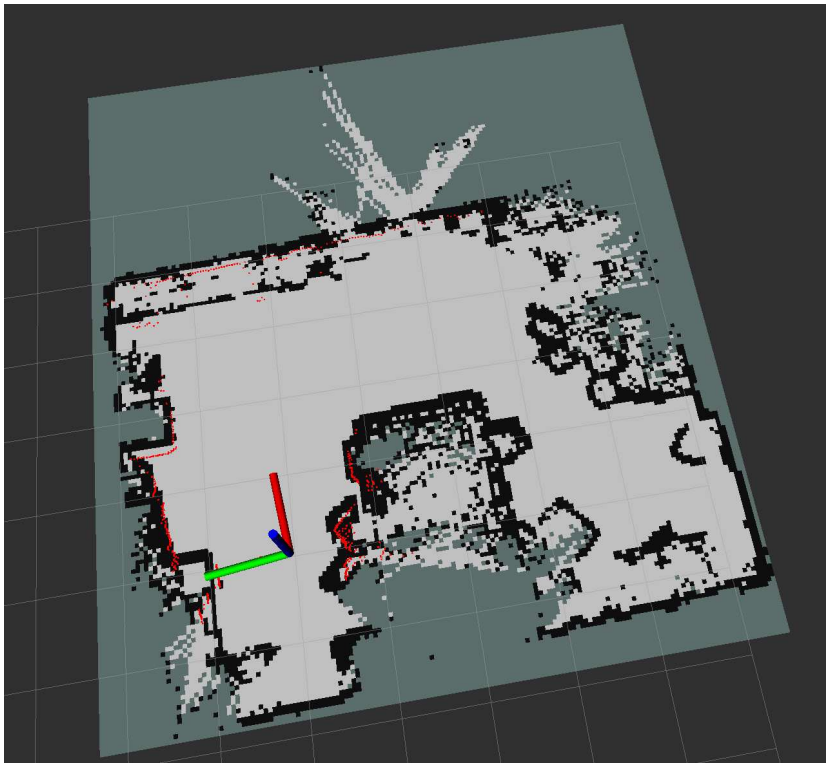
4. Lokalizacija i navigacija

Za autonomno upravljanje robotom prvo je potrebno odrediti njegov položaj u prostoru. Položaj i orijentacija se najefikasnije mogu izračunati korištenjem *SLAM* (*Simultaneous Localization and Mapping*) algoritama koji su opisani u 4.1. Kako bi navigacija ispravno radila, potrebno je napraviti nekoliko koraka. Prvo, trebaju postojati senzori kojima se može izvoditi *SLAM* lokalizacija, u ovom radu LiDAR i Intel *Realsense kamera*. LiDAR-om se stvara mapa šireg okruženja mobilnog robota, postupkom opisanim u 4.1. Kamerom se unutar bližeg okruženja prepoznaju predmeti od interesa i radi se korekcija LiDAR-skih očitanja, opisano u poglavlju 3.5.

4.1. Lokalizacija *SLAM* algoritmom

SLAM (*Simultaneous Localization and Mapping*) je proces kojim se istovremeno pokušava smjestiti robota unutar prostora i stvoriti točna mapa prostora. Najefikasnije se može izvesti korištenjem vlastite odometrije, LiDAR-a i kamera koje mogu mjeriti udaljenost do predmeta, opisanih u 2.5.1. Cijelo vrijeme se uz položaj bilježi i orijentacija robota. Najčešće korišteni algoritmi unutar *SLAM-a* su vizualna ekstrakcija karakteristika prostora (kamerom) i ekstrakcija karakteristika iz dvodimenzionalnih laserskih očitanja (LiDAR-om), [20]. Lokalizacija kamerom s pomoću poznatih značajki sa slike određuje vlastiti položaj u odnosu na položaj izmjerene značajke. Potrebno je prikupiti velik broj redundantnih vizualnih informacija iz okruženja kako bi se mogla napraviti lokalizacija RGB-D ili stereo kamerom, pritom je vizualna lokalizacija uglavnom ograničena na zatvorene prostore zbog karakteristika kamere. Lokalizacija korištenjem laserskih očitanja s dvodimenzionalnih ili trodimenzionalnih LiDAR-a je znatno učinkovitija. LiDAR stvara mapu prostora, a na temelju izmjerene udaljenosti do poznatih prostornih značajki (npr. kutovi predmeta) može se izračunati vlastiti prostorni položaj i orijenta-

cija. U ovom radu se koristi LiDAR za lokalizaciju u prostoru zbog mogućnosti stvaranja mape većeg dijela okruženja i brzine stvaranja mape. Primjer stvorene mape prikazan je slikom 4.1.



Slika 4.1. *SLAM* mapa okruženja robota.

Alat koji se koristio za izvođenje *SLAM* algoritma, *SLAM Toolbox* [21], omogućava tri načina rada. Sinkrono stvaranje mape, asinkrono stvaranje mape i „čista" lokalizacija. Kada je potrebno stvoriti što precizniju sliku prostora, najbolje je stvarati mapu sinkrono jer se koristi više senzorskih mjerenja koja se preklapanjem spajaju u jedno. Sinkronim stvaranjem mape se osigurava da će se iskoristiti sva senzorska mjerenja za stvaranje slike, ali u zamjenu za brzinu stvaranja slike. Asinkrono stvaranje mape, koje je korišteno u sklopu ovoga rada, koristi samo posljednje očitanje sa senzora. Nakon što je ispunjen uvjet za uzimanje zadnjeg očitanja, novim se očitanjem proširuje postojeća slika prostora. „Čista" lokalizacija (*pure localization*) je pogodna kada je okruženje za koje se stvara mapa promjenjivo. Očitanja sa senzora se spremaju u klizni međuspremnik koji se cijelo vrijeme prepisuje novim podatcima. Tako se osigurava da će u svakom trenutku na *SLAM* mapi biti prikazani samo podatci koji su nedavno pristigli sa senzora. Time se omogućava fleksibilnost u stvaranju slike prostora.

4.1.1. Primjena *SLAM* algoritma

Na slici 4.1. prikazana je *SLAM* mapa laboratorija na Borongaju u kojem se nalazio mobilni robot. Mobilni robot prikazan je osima u donjem lijevom dijelu slike. Prepreke su prikazane crnom bojom, a prazni prostor svijetlo sivom. Na preprekama u bliskom okruženju oko robota se mogu vidjeti crvene točkice, koje predstavljaju očitanja s LiDAR-a, pretvorena iz trodimenzionalnih u dvodimenzionalna očitanja. Kako bi *SLAM* algoritam, [21], mogao koristiti očitanja s LiDAR-a, potrebno ih je pretvoriti u dvodimenzionalne podatke. U tu svrhu koristio se *ROS2* paket *Pointcloud to Laserscan*, opisan u [22]. Čvoru za pretvorbu trodimenzionalnih u dvodimenzionalna očitanja predaju se parametri kojima se određuje kako se okruženje interpretira. Zadaje se najveća visina laserskog očitanja koja se smatra važnom za stvaranje mape prostora, početni i krajnji kutovi iz kojih se uzimaju mjerenja i najveća i najmanja udaljenost od senzora do predmeta. Najvažniji dio konfiguracije prikazan je u nastavku:

```
'min_height': 0.0,
'max_height': 0.5,
'angle_min': -3.14,
'angle_max': 3.14,
'range_min': 0.2,
'range_max': 8.0,
'use_inf': True,
```

Parametrima za najmanju i najveću vertikalnu udaljenost od senzora do predmeta (*min_height* i *max_height*) se određuje da se senzorska očitanja prostora iznad 0.5 *m* ignoriraju pri stvaranju mape okruženja robota. Parametrima za najmanji i najveći kut između senzora i predmeta (*angle_min* i *angle_max*) određuje se da se svi predmeti u vidokrugu senzora (360°) uzimaju u obzir pri stvaranju slike prostora, a parametrima za najmanju i najveću udaljenost (*range_min* i *range_max*) određuju se najmanja i najveća moguća udaljenost od senzora do predmeta. Parametrom *use_inf* definira se da se dio prostora u kojem ne postoji predmet unutar definiranih granica smatra praznim.

4.2. Navigacija

Za navigaciju se koristi *ROS2* programski paket *Nav2*, [23]. Korištenjem dvaju senzora za navigaciju (kamera i LiDAR) dobiva se redundancija senzorskih očitanja i povećava se točnost u prepoznavanju i zaobilaženju prepreka. Kamerom se točno mogu odrediti područja unutar kojih se mobilni robot ne bi smio kretati. Nakon što se ispravno podese senzori za stvaranje *SLAM* mape okruženja mobilnog robota, potrebno je definirati parametre potrebne za navigaciju mobilnog robota.

4.3. Struktura programskog paketa *Nav2*

Postoji nekoliko čvorova koji se istovremeno izvode za ispravan rad programskog paketa *Nav2*. Uvođenjem koncepta stabala ponašanja (*behaviour trees*) definiraju se opća ponašanja robota koja se mogu koristiti u navigaciji u prostoru. Primjerice, kreći se unaprijed, kreći se unatrag, kreći se lijevo ili desno su opća ponašanja kojima se u daljnjim slojevima navigacijskog paketa mogu koristiti čvorovi. Stabla ponašanja omogućavaju bolji pristup složenim zadacima poput navigacije nego konačni automati, zbog fleksibilnosti u pristupu. Kada bi se u navigaciji koristili konačni automati, za svaki specifični slučaj bi se trebalo definirati točno ponašanje koje se očekuje. Pristupom u kojem se može definirati opće ponašanje omogućava se prilagodljivost određenom problemu, jer rješenje ne mora biti jednoznačno određeno, [24]. Kako bi se preciznije mogla definirati ili izmijeniti ponašanja robota, vrlo lako se mogu dodati dodatci kodu. Velik broj dodataka dolazi uz programski paket *Nav2*, a mogu se razviti i bilo kakvi drugi dodaci koji su potrebni izvršavanju na vrlo jednostavan način. Najvažniji dodaci bit će opisani u nastavku.

4.3.1. Navigacijski planeri (*planner server*)

Zadatak navigacijskog planera je računanje putanje kojom se mobilni robot treba kretati kroz okruženje kako bi stigao do odredišta. Postoje tri moguće zadaće planera, računanje najkraćeg mogućeg puta, računanje putanje za obilazak cijelog okruženja mobilnog robota i računanje putanje po definiranoj ruti. Glavni zadatak je računanje važeće i potencijalno optimalne putanje do odredišta [24]. Posebni dodatak kodu (*plugin*) koristi se za provjeru dolaska robota do cilja (*general goal checker*) kojem je potrebno definirati maksimalno prihvatljivo odstupanje robota od krajnjeg položaja u metrima i orijentacije

u stupnjevima. Za svrhe ovog rada maksimalno odstupanje stvarnog krajnjeg položaja robota od zahtijevanog krajnjeg položaja u metrima iznosi 0.25 m, a maksimalno odstupanje stvarne krajnje orijentacije robota od zahtijevane orijentacije u radijanima 0.25 rad, točnije 14.5°.

4.3.2. Navigacijski upravljači (*controller server*)

Navigacijski upravljači se služe putanjom koju izračuna navigacijski planer opisan u prethodnom potpoglavlju. Zadatak upravljača je izvesti kretnje baze robota i pritom pratiti putanju koju određuje navigacijski planer. Upravljač ima pristup senzorskim očitanjima iz neposrednog okruženja baze robota kako bi izračunao optimalne kretnje pri zaobilaženju prepreka.

4.3.3. Upravljači ponašanja (*behaviour server*)

Cilj upravljača ponašanja je izračunati kretnje kojima će se robot dovesti iz neželjenog u željeno stanje. Primjerice, ako se ispred robota nađe čovjek, robot će se kretati unatrag ili će se pokušati okrenuti kako bi pronašao drugu putanju kojom može doći do definiranog cilja.

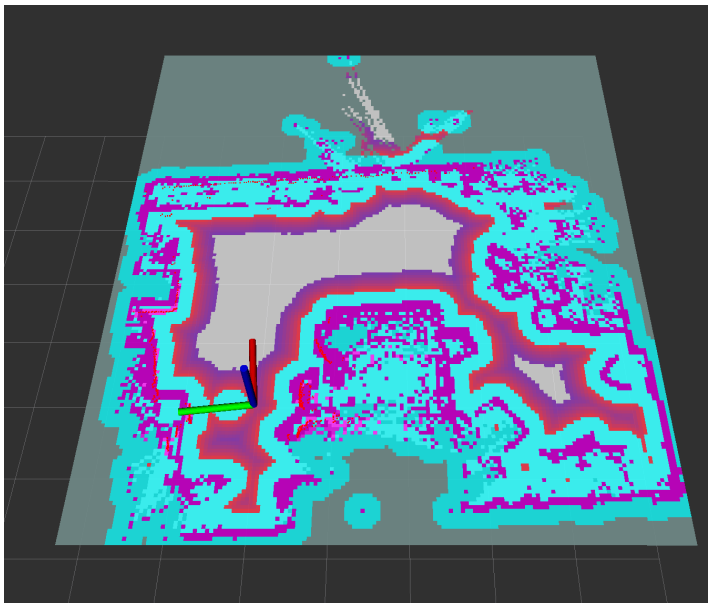
4.3.4. „Zaglađivači" putanje (*smoother server*)

Vrlo često će navigacijski planeri izračunati putanju do odredišta koja ima nagle rotacije pri skretanju. Razlog tome je što pri stvaranju putanje od ishodišta do odredišta, putanje mogu sadržavati oštra skretanja, što bi robotu otežavalo kretanje kroz prostor. U tu svrhu su uvedeni „zaglađivači" kojima je zadatak ukloniti nagla skretanja robota pri praćenju putanje, točnije, unaprijediti postojeću putanju.

4.4. Konfiguracija programskog paketa *Nav2*

Programski paket *Nav2* navigaciju razlaže u dvije cjeline. Globalna navigacija korištenjem globalne mape i lokalna navigacija korištenjem lokalne mape. Koristi se upravljačima i planerima opisanim u 4.3.1., 4.3.2., 4.3.3. i 4.3.4. Za ispravno stvaranje mape za navigaciju može se iskoristiti *SLAM* mapa prikazana u 4.1. Na slici 4.2. čvrste prepreke prikazane su svijetlo ljubičastom bojom, područja sa zabranom kretanja prikazana su

svijetlo plavom bojom, dok su područja u kojima se mobilni robot smije kretati uz smanjenu brzinu prikazana crvenom i tamno ljubičastom bojom.



Slika 4.2. Prikaz mape korištene za navigaciju u zatvorenom prostoru.

Širina područja sa zabranom kretanja ovisi o dimenzijama koje se definiraju u konfiguracijskoj datoteci *nav2_bringup.yaml* prikazanoj u nastavku.

```
global_costmap:
  global_costmap:
    ros__parameters:
      update_frequency: 1.0
      publish_frequency: 1.0
      global_frame: map
      robot_base_frame: base_link
      robot_radius: 0.3
        ...
```

Globalna mapa podijeljena je u nekoliko slojeva. Statički i inflacijski sloj i sloj s preprekama. U statičkom sloju nalazi se *SLAM* mapa iz koje se stvara dio navigacijske mape s čvrstim preprekama (zidovima, prikazanim svijetlo ljubičastom bojom na slici 4.2.). Inflacijski sloj se nalazi uz statički sloj i stvara područja sa zabranom kretanja ovisno o širini robota (svijetlo plavo područje na slici 4.2.). Sloj s preprekama sadrži čvrste prepreke (osim zidova) koje su se u trenutku stvaranja *SLAM* mape nalazile u prostoru. Prepreke

se iz sloja s preprekama mogu ukloniti ako se senzorskim očitanjima utvrdi da ih više nema, točnije, lokalna navigacijska mapa može izmijeniti sloj s preprekama globalne navigacijske mape. Globalna mapa se rjeđe osvježava i služi kao predložak za navigaciju u prostoru, dok lokalna mapa služi za dinamičko izbjegavanje prepreka. Lokalna navigacijska mapa sastoji se od vokselskog i statičkog sloja i sloja s preprekama. Statički sloj se nasljeđuje iz globalne navigacijske mape, dok se vokselski sloj i sloj s preprekama osvježavaju velikom frekvencijom. Vokselski sloj služi za obradu dvodimenzionalnih i trodimenzionalnih mjerenja sa senzora, oblaci točaka i laserska očitanja.

4.4.1. Prilagodba programskom paketu *Nav2*

Vokselski sloj unutar paketa *Nav2* može obrađivati dvodimenzionalne i trodimenzionalne podatke, ali se u radu s Intel *Realsense* kamerom mogu pojaviti greške. Nakon što se objave podatci u trodimenzionalnom obliku na *ROS* temu, može se dogoditi da trajno ostanu u lokalnoj mapi. Zbog greške u implementaciji vokselskog sloja, nakon što se objave podatci na temu, neće se očistiti kada se objavi prazan skup podataka. Točnije, prazan skup podataka se ne interpretira kao da prepreke ne postoje. Zbog toga je potrebno pretvoriti objavljene podatke o preprekama u prostoru s *ROS* teme `/no/go/zones` u dvodimenzionalne podatke, jer će onda prazan skup senzorskih očitanja označavati da nema prepreka.

5. Eksperimenti

Kako bi mobilni samostalno mogao doći do odredišta koje mu se odredi, potrebno je izvesti sljedeće korake. Prvo je potrebno pokrenuti senzorske čvorove, kameru i LiDAR. Nakon toga programske čvorove za prepoznavanje predmeta u dvodimenzionalnom i trodimenzionalnom prostoru opisane u 3.3.2. i 3.5. Kako je za ispravan rad lokalizacijskih i navigacijskih algoritama opisanih u poglavljima 4. i 4.2. potrebno koristiti dvodimenzionalne podatke, pokreće se čvor za pretvorbu trodimenzionalnih u dvodimenzionalna očitanja. Nakon toga se mogu pokrenuti programski paketi za *SLAM*, *Slam Toolbox* i za navigaciju, *Nav2*. Nakon pokretanja svih potrebnih čvorova, u programskom alatu *RViz2* mogu se vidjeti svi podatci koje vraćaju opisani programski paketi. Najvažniji su podatci o stvorenoj karti prostora na *ROS* temi `/map`, podatci o globalnoj i lokalnoj navigacijskoj mapi na *ROS* temama `/global_costmap` i `/local_costmap` i podatci o zabranjenim područjima kretanja opisani u 3.5. s teme `/no/go/zones`. Zadavanjem dvodimenzionalnog cilja korištenjem alata *RViz2*, mobilnom robotu se određuju krajnji položaj i orijentacija na *SLAM* mapi.

5.1. Rezultati

Provedeno je nekoliko eksperimenata na način opisan u 5. u otvorenim i zatvorenim prostorijama. U zatvorenim prostorijama robot uspijeva doći do cilja bez poteškoća, osim kada su prepreke postavljene vrlo blizu jedna drugoj, tada ne uspijeva pronaći putanju između prepreka. Uspješno uspijeva prepoznati i klasificirati prepreke putem kamere programskim čvorovima opisanim u poglavljima 3.3.2. i 3.5. Klasifikacija i označavanje prepreka u prostoru povećava redundanciju sustava i pospješuje svojstva navigacije u slučajevima kada se LiDAR-om ne uspijeva točno odrediti područje sa zabranom kretnji. Također, poboljšano je prepoznavanje prepreka u neposrednoj blizini robota koje se

LiDAR-om ne mogu prepoznati i poboljšano je prepoznavanje dinamičnih prepreka u prostoru izravno ispred robota. Ispitivanje na otvorenim prostorima proizvelo je slične rezultate. Mobilni robot je uspijevao doći do odredišta bez poteškoća. Robot je uspješno uspio zaobići i pomične prepreke poput ljudi koji bi mu stali na već određeni put.

5.2. Rasprava i daljnji rad

Provedenim eksperimentima opisanim u 5. i rezultatima opisanim u 5.1. utvrđeno je da postoje dijelovi rada koji bi se mogli unaprijediti u budućnosti. Primjerice, osmisliti dodatak programskom paketu *Nav2* kojim će se mobilnom robotu korištenom u ovomu radu preciznije moći odrediti u kojim je slučajevima potrebno koristiti svojstva gibanja omogućena kotačima opisanim u 2.3.2., a kada se isti kotači mogu koristiti za uobičajeno kretanje mobilnih robota na kotačima. Za svrhe ovog rada odabran je hibridni pristup, gdje se u slučajevima kada je potrebno napraviti okret za 180° koriste svojstva svesmjernih kotača, a u slučajevima kada bi se robot mogao gibati horizontalno, odabran je uobičajeni pristup, zaobilazeći svojstva svesmjernih kotača. Time se postiže da kamera uvijek gleda prema području izravno ispred mobilnog robota, kako bi se mogao segmentirati prostor. Povrh toga, unaprijeđenjem programskog koda za prepoznavanje prepreka 3.3.2. kako bi mogao prepoznati točno koja je točka najbliža kameri, umjesto da se kao najbliža točka uzima ona koja je najbliža u skupu od četiri točke koje omeđuju predmet i središnje točke predmeta, preciznije bi se mogle mjeriti udaljenosti do predmeta.

6. Zaključak

Kako bi se uspješno mogla provoditi lokalizacija i navigacija mobilnog robota u prostoru bilo je potrebno koristiti senzore opisane u poglavlju 2.3.2., osmisliti algoritme za pomoć pri navigaciji iz poglavlja 3.3.2. i 3.5. i primijeniti postojeće algoritme opisane u 4. i 4.2. Uvođenjem optičkog senzora kao što je kamera, sustav pri navigaciji ima dodatnu redundanciju čime se smanjuje vjerojatnost pogreške pri navigaciji kroz prostor. Također, očitanjima s kamere pospješuje se navigacija u nepoznatim prostorima, kada se korištenjem samo LiDAR-a ne može odrediti najdalja točka predmeta. Ako se kamera i LiDAR ispravno kalibriraju, moguće je efikasnije segmentirati okruženje robota i efikasnije doći do odredišne točke.

Literatura

[1] N. Maanyu, G. Raj, V. Krishna, i D. S. B. Choubey, “A study on tesla autopilot”, sv. 71, 2020. [Mrežno]. Adresa: https://www.ijspr.com/citations/v71n1/IJSPR_7101_30573.pdf

[2] E. Guizzo, “Amazon kiva”, https://spectrum.ieee.org/three-engineers-hundreds-of-robots-one-warehouse, pristupljeno: 13.2.2025., 18:00.

[3] Segway, “Nova carter dokumentacija”, https://nvidia-isaac-ros.github.io/nova/getting_started/platforms/nova_carter.html, pristupljeno: 13.2.2025., 18:00.

[4] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, i W. Woodall, “Robot operating system 2: Design, architecture, and uses in the wild”, *Science Robotics*, sv. 7, br. 66, str. eabm6074, 2022. https://doi.org/10.1126/scirobotics.abm6074

[5] O. Robotics, “Arhitektura ros2”, https://docs.ros.org/en/dashing/Concepts/About-Internal-Interfaces.html, pristupljeno: 2.1.2025., 17:50.

[6] R.-T. Innovations, “Dds dokumentacija”, https://community.rti.com/static/documentation/connext-dds/current/doc/manuals/connext_dds_professional/getting_started_guide/cpp11/intro_qos.html#qos-profiles, pristupljeno: 2.1.2025., 17:20.

[7] R. Galati, G. Mantriota, i G. Reina, “Adaptive heading correction for an industrial heavy-duty omnidirectional robot”, *Scientific Reports*, sv. 12, 11 2022. https://doi.org/10.1038/s41598-022-24270-x

[8] J. Shao, T. He, J.-G. Jiang, i Y. Zhang, “Recent patents on omni-directional wheel applied on wheeled mobile robot”, *Recent Patents on Mechanical Engineering*, sv. 9,

str. 215–221, 08 2016. https://doi.org/10.2174/2212797609666160616091009

[9] Livox, “Livox mid 360 lidar”, https://www.livoxtech.com/mid-360, pristupljeno: 13.2.2025., 18:00.

[10] N. Lopac, I. Jurdana, A. Brnelić, i T. Krljan, “Application of laser systems for detection and ranging in the modern road transportation and maritime sector”, *Sensors*, sv. 22, br. 16, 2022. https://doi.org/10.3390/s22165946

[11] C. Soto i R. Murphy, “Using the kinect for search and rescue robotics”, 11 2012. https://doi.org/10.1109/SSRR.2012.6523918

[12] Intel, “Intel d415 dubinska kamera”, https://www.intelrealsense.com/depth-camera-d415/, pristupljeno: 13.2.2025., 18:00.

[13] ——, “Intel realsense dokumentacija”, https://www.intelrealsense.com/beginners-guide-to-depth/, pristupljeno: 28.12.2024., 14:55.

[14] M. Brown, D. Burschka, i G. Hager, “Advances in computational stereo”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, sv. 25, br. 8, str. 993–1008, 2003. https://doi.org/10.1109/TPAMI.2003.1217603

[15] G. Jocher, A. Chaurasia, i J. Qiu, “Ultralytics yolov8”, 2023. [Mrežno]. Adresa: https://github.com/ultralytics/ultralytics

[16] OpenCV, “Opencv dokumentacija”, https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#moments, pristupljeno: 23.12.2024., 14:50.

[17] MathWorks, “Mathworks dokumentacija”, https://www.mathworks.com/help/vision/ug/camera-calibration.html, pristupljeno: 3.12.2024., 12:00.

[18] ——, “Mathworks dokumentacija”, https://www.mathworks.com/help/vision/ref/cameracalibrator-app.html, pristupljeno: 3.12.2024., 12:33.

[19] N. Burrus, “Pretvaranje 2d koordinata u 3d koordinate”, https://nicolas.burrus.name/oldstuff/kinect_calibration/, pristupljeno: 22.12.2024., 23:50.

[20] T. Chong, X. Tang, C. Leng, M. Yogeswaran, O. Ng, i Y. Chong, "Sensor technologies and simultaneous localization and mapping (slam)", *Procedia Computer Science*, sv. 76, str. 174–179, 2015., 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IEEE IRIS2015). https://doi.org/https://doi.org/10.1016/j.procs.2015.12.336

[21] S. Macenski i I. Jambrecic, "Slam toolbox: Slam for the dynamic world", *Journal of Open Source Software*, sv. 6, br. 61, str. 2783, 2021. https://doi.org/10.21105/joss.02783

[22] O. Robotics, "Ros2 čvor za pretvorbu 3d lidarskih očitanja u 2d očitanja", http://wiki.ros.org/pointcloud_to_laserscan, pristupljeno: 26.1.2025., 18:00.

[23] S. Macenski, F. Martin, R. White, i J. Ginés Clavero, "The marathon 2: A navigation system", u *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[24] O. Navigation, "Ros2 čvor za pretvorbu 3d lidarskih očitanja u 2d očitanja", https://docs.nav2.org/concepts/index.html#behavior-trees, pristupljeno: 2.2.2025., 18:00.

Sažetak

Lokalizacija i navigacija mobilnog robota za dekontaminaciju

Dino Smirčić

Cilj ovoga rada bio je osmisliti algoritme za samostalno upravljanje robota u poznatim i nepoznatim okruženjima. Korišteni su LiDAR (*Light Detection and Ranging*) i Intel *Realnsense depth kamera* za stvaranje mape za navigaciju prostorom. Modeliran je nosač za LiDAR i kameru koji se dizajnirao po mjeri za mobilnog robota. Platforma koja se koristila je *Scout Agilex Scout 2.0 mini*, a algoritmi su napisani u programskom alatu *ROS2*. Za prepoznavanje prepreka u okruženju korišten je algoritam *YOLOv8*. Osmišljen je algoritam za prepoznavanje i označavanje prepreka u trodimenzionalnom prostoru kako bi se označila područja u kojima je robotu zabranjeno kretanje. Provedena su ispitivanja navigacije robota u otvorenom i zatvorenom prostoru s pomičnim i nepomičnim preprekama u kojima je robot uspio doći do zadanog cilja.

Ključne riječi: lokalizacija; navigacija; autonomija; segmentacija slike; prepoznavanje predmeta; depth kamera; LiDAR; SLAM

Abstract

Localization and navigation of a mobile robot for decontamination

Dino Smirčić

The goal of this thesis was to design and implement algorithms for autonomous robot navigation. LiDAR (*Light Detection and Ranging*) and *Intel Realsense depth camera* technology were used to accomplish this. A mount was designed and modeled for the purpose of connecting both sensors into a fixed frame. *Scout Agilex 2.0 mini* was used as a mobile base. The algorithms were written in Python as ROS2 packages. A ROS2 3D segmentation package was designed for marking obstacles for collision avoidance in real-world scenarios. Tests were put in place in both open and closed spaces with both static and moving obstacles in which the robot accomplished reaching its goal.

Keywords: localization; navigation; autonomy; image segmentation; object detection; depth camera; LiDAR; SLAM