# Podudaranje skenova u ROS-u

**Šafar, Sandro**

SVEUČILIŠTE U RIJECI
**TEHNIČKI FAKULTET**
Preddiplomski studij računarstva

Završni rad

# Podudaranje skenova u ROS-u

Rijeka, rujan 2022.

Sandro Šafar
0069085928

SVEUČILIŠTE U RIJECI
**TEHNIČKI FAKULTET**
Preddiplomski studij računarstva

Završni rad

# Podudaranje skenova u ROS-u

Mentor: prof.dr.sc. Kristijan Lenac

Rijeka, rujan 2022.

Sandro Šafar
0069085928

**Umjesto ove stranice umetnuti zadatak za završni ili diplomski rad**

## Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, rujan 2022.

_____

Ime Prezime

# Zahvala

Zahvaljujem svojem mentoru prof. dr. sc. Kristijanu Lencu na korisnim usmjeravanjima, savjetima i prijedlozima tijekom pisanja ovoga rada te strpljivost i susretljivost pri mojim brojnim upitima.

Veliku zahvalnost posvećujem i kolegama zbog njihove konstantne moralne i tehničke podrške, posebice kolegici Stelli Dumenčić.

Posebnu zahvalnost dugujem svojim roditeljima, svojoj sestri, svojim prijateljima i svojoj curi, koji su uvijek bili uz mene i pružali mi potporu kad god je bila potrebna.

Veliko HVALA svima!

# Sadržaj

# Poglavlje 1

# Uvod

Robotika je, nedvojbeno, jedna od najzanimljivijih i najbrže rastućih grana tehnologije i inženjerstva. Njezina uska povezanost s računarstvom, elektrotehnikom i strojarstvom kao i njen potencijal da u samo nekoliko desetljeća postane primarni razlog i objašnjenje napretka čovječanstva, objašnjava brzinu kojom postaje sve popularnija.

Osim zbog dobivene popularnosti u medijima, posebice u filmskoj industriji, robotika je u rastu zbog svoje raznolike mogućnosti upotrebe. Postoje brojne vrste robota, primjerice industrijski roboti koji pomažu u proizvodnji, leteći roboti (dronovi) koji nam daju korisne informacije, mobilni roboti koji sami mapiraju okoliš ili izvode operacije traganja i spavašanja, čovjekoliki roboti koji pomažu ljudima direktno te mnogi drugi.

Navedeni skok u popularnosti dovodi do ubrzanog istraživanja svih dijelova robotike. To je istina za prikupljanje novog znanja, ali također i za organizaciju i analiziranje već postojećeg znanja u svrhu lakšeg učenja budućim generacijama. Ovaj završni rad se bavi upravo takvim analiziranjem, s ciljem pojednostavljenja shvaćanja koncepta podudaranja skenova.

Konkretni cilj ovog rada je analizirati implementacije za podudaranje skenova u ROS (eng. Robot Operating System) okruženju za razvoj aplikacija u mobilnoj robotici te testirati jednu od njih i usporediti ju s alternativnom implementacijom.

Sadržaj rada je podijeljen na 8 poglavlja. Nakon ovog uvodnog prvog poglavlja slijedi poglavlje 2 u kojem je objašnjeno što je ROS i kako funkcionira. U poglavlju 3 se detaljno objašnjava što je podudaranje skenova, koje vrste postoje i što ih razlikuje. Zatim se u poglavljima 4 i 5 opisuju dvije metode za podudaranje skenova koje su u poglavlju 7 uspoređene. Nakon toga, u poglavlju 6 se nalaze upute za praktičnu primjenu jedne metode za podudaranje skenova unutar Gazebo simulatora. Završit ćemo zaključkom u poglavlju 8 koje će sumirati rezultate ovog rada.

# Poglavlje 2

# Robot Operating System (ROS)

## 2.1 Što je ROS?

ROS (eng. Robot Operating System) odnosno Robotski Operacijski Sustav (slobodan prijevod) je fleksibilan softverski okvir koji služi za pisanje softvera za robote. To je kolekcija alata i knjižnica čiji je cilj olakšati zadatak stvaranja kompleksnog i robusnog ponašanja robota. Iako u svojem imenu ima riječi "operacijski sustav", ROS bi bilo bolje opisati kao posrednika između operacijskih sustava i procesa koji se izvode. Također, ne može se reći da je ROS operacijski sustav jer se mora instalirati na operacijski sustav da bi radio. Većinom se koristi za operacijski sustav Linux, iako postoje eksperimentalne verzije za operacijske sustave Windows i MacOS.

Prvi kodovi ROSa viđeni su 7. studenog 2007., kada su studenti Sveučilišta Stanford Eric Berger i Keenan Wyrobek, radeći za firmu Willow Garage, objavili prvi izvorni kod na stranicu SourceForge, koja pomaže projektima otvorenog koda da budu što uspješniji. Willow Garage je od 2007. do 2013. bio zadužen za ROS iako ga je razvijalo nebrojivo puno timova i institucija diljem svijeta. Zainteresiranost za ROS skočila je 2012. godine kad je Willow Garage stvorio Open Source Robotics

Foundation (OSRF), odnosno Zakladu za Robotiku Otvorenog Koda. Od tada, svake godine je izašla nova distribucija ROSa kao i anualni festival ROSCon te mnoge knjige o ROSu. OSRF je 2017. godine promijenio ime u Open Robotics te i danas pod tim imenom održava ROS. [1]

Trenutno postoje ROS1 i ROS2. ROS1 je prva verzija ROS-a i njena prva distribucija "ROS Box Turtle" izdana je u ožujku 2010. godine. Bila je dostupna i namijenjena za razne Ubuntu i Linux distribucije, ali mogla se i pokrenuti na MacOS-u uz limitiranu kompatibilnost [2]. ROS2 je izdan u prosincu 2017. godine prvom distribucijom imena "ROS Ardent Apalone". [1] Razlog izdavanja ROS2 je u mnogim nedostacima ROS1 koje se nije moglo popraviti, te je zato stvorena nova verzija na zahtjev zajednice.

### 2.1.1 Zašto ROS?

Razlog stvaranja ROSa leži u tome što je stvaranje softvera za robote vrlo zahtjevno. Problemi koji se ljudima čine vrlo jednostavni, robotima su znatno teži jer oni nemaju razvijenu intuiciju te i najmanja razlika može prouzročiti ponašanje suprotno od očekivanog. Problem je u tome što je vrlo teško definirati sve varijacije određenih problema i sve minimalne razlike koje bi mogle prouzročiti probleme. Za rješavanje tog problema nije sposoban nijedan laboratorij, tim ili institucija.

Kao posljedica toga izgrađen je ROS, kako bi potaknuo zajedničko razvijanje softvera za robote. Primjerice, jedan laboratorij ili institucija bi mogla imati stručnjake za mapiranje unutarnjih prostora i tako pridonijeti sustavu svjetskih razmjera za stvaranje mapa unutarnjih prostora. S druge strane, drugi laboratorij ili institucija bi mogla zato imati stručnjake za navigaciju mapama ili stručnjake za pronalaženje najkraćeg puta između bilo koje dvije točke unutar bilo koje mape.

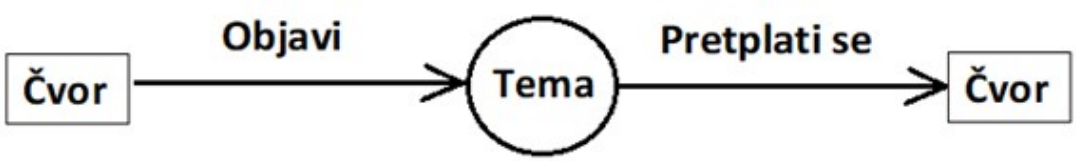
Razvijanje ROS-a je znatno pojednostavilo napredak tehnologije obzirom da je

prije njega svaka institucija koja proizvodi robote morala razvijati svoje programsko sučelje za komunikaciju i upravljanje svojim robotima. Dolaskom i standardiziranjem ROS-a različiti proizvođači mogu koristiti isto sučelje te efikasnije koristiti provedeno vrijeme razvoja.

Glavna ideja je da svi rade zajedno u jednom operacijskom sustavu te nadopunjavaju taj sustav i grade slojeve na rad drugih. Velika prednost ROS-a je to što su svi algoritmi i programi unutar njega otvorenog koda i besplatni te je zato nadogradnja jednostavna.

### 2.1.2 Kako funkcionira?

Robotski Operacijski Sustav nije pravi operacijski sustav, već se više smatra okruženjem. Većina njegove snage nalazi se u mogućnosti da spoji čvorove u jednu cjelinu koristeći pritom "objavi/pretplati se" protokol. To znači da ako čvor ima neku informaciju koju želi podijeliti, to će napraviti tako da "objavi temu". Ako je drugi čvor zainteresiran za tu informaciju, može se "pretplatiti" na tu temu i čitati informaciju. Svaka tema ima svoju vrstu poruke koju čvorovi šalju i primaju kao što ima i svoju frekvenciju slanja navedenih poruka. Najčešće je riječ o kontinuiranom slanju poruka određeni broj puta u sekundi.




Slika 2.1 *Način dijeljenja informacija u ROS-u*

Nadalje, važno je napomenuti da koristeći funkcije možemo vidjeti sve stavke svakog čvora i svake teme te manipulirati tim stavkama. Primjerice, u ROS1 funkcijom

"rostopic" moguće je dobiti informacije o željenoj temi, dok se funkcijom "rosnode" dobivaju informacije o željenom čvoru. Navedene funkcije imaju dodatnu sintaksu koju se mora nadopuniti da bi ROS-u bilo jasno koju temu ili čvor želimo analizirati.

To se radi tako da se koriste ključne riječi do kojih se može doći ako se samo napiše ime funkcije i na kraju se doda nastavak "-h" kojim se traži pomoć od sustava. U toj naredbi, zastavica h označava englesku riječ "help" za pomoć. Primjerice, naredba "rostopic -h" će korisniku objasniti sve vezano uz funkciju rostopic uključujući sve ono što se može prikazati i kojim ključnim riječima.

Navedenom naredbom će se saznati da je moguće nadopuniti funkciju rostopic ključnom riječi "list" da bi se dobio popis svih aktivnih ROS tema. Naredba "rostopic list" bi nam vratila imena svih trenutno aktivnih tema. Ako korisnik želi saznati konkretne stavke i informacije određene teme, napisao bi "rostopic echo X" gdje je X ime teme o kojoj želi znati više. Vidio bi ime teme, vrstu poruke koja se šalje na tu temu kao i broj čvorova koji objavljuju na temu te broj čvorova koji su pretplaćeni na temu. Doda li se tome i zastavica -v (što je kratica za englesku riječ "verbose" koja označava opisivanje uz više riječi od potrebnog) na kraj, bit će vidljivi i detalji o svim navedenim čvorovima, ne samo njihov broj.

Postoje i mnoge druge naredbe koje daju informacije o čvorovima i temama što pobliže objašnjava kako ROS funkcionira i koje se informacije šalju što je pogotovo korisno u otkrivanju problema sa sustavom pri njegovom neočekivanom ponašanju.

# Poglavlje 3

# Podudaranje skenova

## 3.1 Što je sken?

Sken je pojam koji opisuje proces digitalizacije slike. Digitalizacija slike omogućava da slika bude pohranjena, modificirana ili da ju računalo može razumijeti. Ovo se najčešće radi s optičkim skenerom.

Danas je moguće kameru svojeg pametnog mobitela koristiti kao skener. Primjerice, kamerom se mogu skenirati QR kodovi [1] koji korisnika vode do ciljanih poveznica.

Skenovi mogu imati više dimenzija. Najčešće se radi samo o 2D i 3D skenovima, no u slučaju ultrazvuka beba, postoje i 4D skenovi koji daju efekt videa umjesto slike. 2D ultrazvuci, kao i 2D skenovi, su najčešći u praksi i stvaraju jednostavnu 2D digitalnu sliku bez ikakve dubine. 3D skenovi daju 3D digitalnu sliku s dubinom tako da kod ultrazvuka roditelji mogu vidjeti lice svojeg djeteta, a ne samo njegov obris.

[1] QR (eng. Quick Response) kod je kod koji se sastoji od crnih i bijelih kvadrata, najčešće korišten za pohranu poveznica koje se mogu čitati kamerom pametnog mobitela.

Slika 3.1 *Primjer QR koda (ovaj vodi na glavnu stranicu Wikipedije) [3]*


Slika 3.2 *Primjer 2D, 3D i 4D skenova [4]*

## 3.2 Sken u robotici

Sken u području robotike označava posebnu vrstu skena koja se radi laserom. Može se zamisliti kao niz skeniranih točaka koji predstavlja krivulju skeniranog okoliša. Laserskim skeniranjem robot može raditi detekciju objekata na način da izmjeri udaljenost između lasera i površine objekta kojeg skenira. Ovo omogućava konstrukciju 3D modela objekta kombiniranjem više modela njegove površine iz mnogih različitih kuteva.

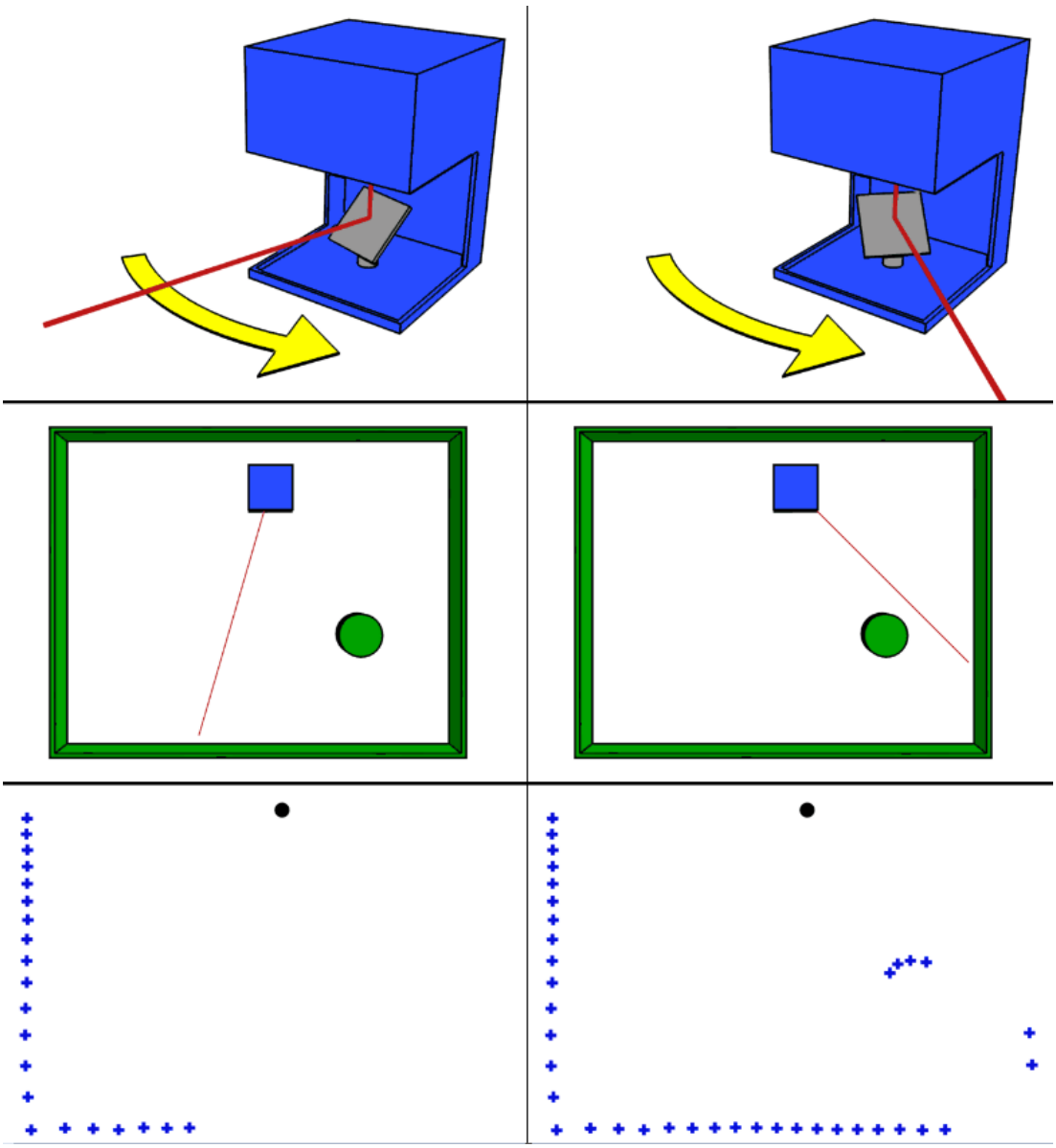
Osim detekcije objekata, lasersko skeniranje se koristi za detekciju prepreka, skeniranje zgrada, fosila, interijera te okoliša općenito. Ovo znatno pomaže u mapiranju te se u robotici naziva LIDAR što je engleska skraćenica za "LIght Detection And Ranging" odnosno Detekcija I Oscilacija Svjetla (slobodan prijevod) ili "Laser Imaging, Detection And Ranging" odnosno "Snimanje, Detekcija I Oscilacija Lasera" (slobodan prijevod). [5] LIDAR određuje udaljenost objekta koristeći formulu (3.1):

$$d = \frac{c * t}{2} \tag{3.1}$$

gdje je:

- $d$ udaljenost između detektora i detektiranog objekta ili površine
- $c$ brzina svjetlosti
- $t$ vrijeme potrebno da laser dođe do objekta te se vrati do detektora

Laser ne može se proći kroz čvrste objekte te je zato potrebno mnogo perspektiva za potpunu mapu okoliša. Slika 3.3 prikazuje kako izgleda laserski sken. Prvi red prikazuje smjer rotacije LIDAR senzora te njegovu trenutnu poziciju. Drugi red prikazuje gdje u okolišu se crveni laser nalazi te pokazuje njegov okoliš. Treći red prikazuje što robot vidi. Plavi plus znakovi označavaju dosad mapirani okoliš.



Slika 3.3 *Primjer LIDAR skena prije i poslije prolaska objekta [5]*

## 3.3 Što je podudaranje skenova?

Podudaranje skenova (eng. Scan matching) je tehnika koja opisuje podudaranje trenutnog skena napravljenog na novoj poziciji robota s prijašnjim skenom napravljenim na prethodnoj poziciji robota te nam pomaže u mapiranju okoliša, manipulaciji robotom i detekcijom objekata.

## 3.4 Vrste podudaranja skenova

Postoji mnogo algoritama za ostvarivanje podudaranja skenova. Algoritme koji postižu podudaranje skenova dijelimo prema načinu na koji funkcioniraju:

- Algoritmi na temelju korespondencije točaka skena (primjer - Iterativna Najbliža Točka odnosno ICP)
- Algoritmi na temelju korespondencije značajki skena (gdje značajke moraju biti statične)
- Algoritmi koji se ne temelje na korespondenciji (primjer - Transformacija Normalne Distribucije odnosno NDT)

te je bitno reći da se ovi algoritmi mogu koristiti u dva načina skeniranja: Sken-sa-skenom (eng. Scan-to-scan) i sken-sa-mapom (eng. Scan-to-map). Algoritam Transformacija Normalne Distribucije bit će objašnjen u poglavlju 4 te će algoritam Iterativna Najbliža Točka biti objašnjen u poglavlju 5. Oba načina za primjenu algoritama za podudaranje skenova te algoritam Na Temelju Značajki (eng. Feature-based) objašnjeni su u nastavku.

### 3.4.1 Sken-sa-skenom

Sken-sa-skenom je najpoznatija i osnovna verzija podudaranja skenova. Također je najintuitivnija za shvaćanje. U ovoj vrsti podudaranja skenova jednostavno se uspoređuje trenutni sken s prethodnim skenom, neovisno o drugim faktorima.

Glavna mana ovog pristupa je tzv. akumulacija greške. Radi se o tome da se u podudaranju skenova zbog raznih nepredvidivih okolnosti može dogoditi mala pogreška. To su okolnosti poput vremenskih prilika, neravne površine na kojoj je robot, nesavršenosti korištene opreme, itd. Zbog toga što se sken uvijek uspoređuje isključivo s drugim skenom, bez ikakve konstantne vrijednosti na koju se može vratiti, može se dogoditi da se te greške počnu akumulirati te posljedično i povećavati.

Primjerice, ako imamo robota koji svake sekunde uzme jedan sken (i usporedi ga s prethodnim) te zbog greške svakim skenom pretpostavi da je 2 milimetra desno u odnosu na svoju stvarnu poziciju, to se ne čini kao problem. Međutim, zbog akumulacije greške, vremenom će se drastično povećavati razlika između pozicije na kojoj robot misli da se trenutno nalazi te njegove stvarne pozicije. U navedenom primjeru to znači da će se za 1000 sekundi, odnosno 16 minuta i 40 sekundi, on nalaziti 2 metra desno u odnosu na poziciju na kojoj misli da je trenutno.

To, naravno, dovodi do velikih problema kod mapiranja okoliša i lokalizacije robota u danom okolišu. Zbog toga je uz ovu vrstu podudaranja skenova potreban dodatan način za kontrolu apsolutne pozicije, kao primjerice GPS.

### 3.4.2 Sken-sa-mapom

Ova vrsta podudaranja skenova radi se na način da se novi sken uspoređuje s cjelokupnom mapom. Prednost toga je što ne može doći do akumulacije greške jer je mapa uvijek fiksna, odnosno statična. Koristi se u mnogim industrijama, a primjer

primjene nalazi se u autonomnoj vožnji [6]. Postoje čak i podvrste, odnosno različiti načini implementacije sken-sa-mapom podudaranja skenova.

U radu Alexandrosa Filotheoua 2021. godine radi se o podvrsti za koju nije potrebno odrediti odgovarajuće podudaranje između zraka dobivenih iz robotskog fizičkog senzora i drugog izvedenog simulacijom njegovog rada unutar karte. [7]

Drugi primjer je rad za koji su zaslužni Ryu K., Dantanarayana L, Furukawa T. i Dissanayake G. 2016. godine u kojem se predlaže prikaz mape u obliku rešetke. To je značajno jer u svakoj ćeliji navedene rešetke postoji više normalnih distribucija. Također se novi sken prikazuje kao normalna distribucija, pa ovdje također prvi put pričamo o normalna-distribucija-sa-normalnom-distribucijom tehnici podudaranja skenova. Njihovi eksperimentalni rezultati prikazuju poboljšanja u brzini uz manje potrebne komputacijske snage. [8]

### 3.4.3 Na temelju značajki

Kao primjer sken-sa-skenom podudaranja postoji i 2D varijanta podudaranja skenova na temelju značajki točaka i linija koja je prvi put predložena u radu "Feature-Based Laser Scan Matching and Its Application for Indoor Mapping" koji je napisan 2016. godine. [9]

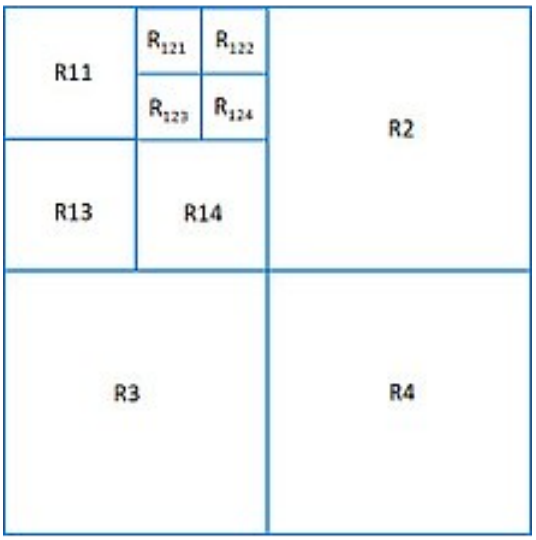
Ovo se radi na način da se prvo značajne točke prepoznaju koristeći SIFT (Scale-invariant feature transform) algoritam. SIFT je algoritam kojeg je izumio David Lowe 1999. godine, a služi za prepoznavanje objekata, mapiranje, navigaciju, modeliranje, itd. Prepoznavanje objekata se radi na način da se trenutna slika uspoređuje s poznatom referentnom bazom pritom podudarajući odgovarajuće značajke temeljene na udaljenosti.

Slika 3.4 *Značajne točke nakon SIFT algoritma*

Zatim se značajne linije prepoznaju koristeći modificirani Split-and-Merge algoritam. Split-and-Merge je tehnika procesiranja slike u kojoj se slika dijeli na pravilne kvadrante, svaki od kojih se testira na definirani kriterij homogenosti. Segmenti se spajaju sa susjedima ako prolaze kriterij homogenosti. Proces razdvajanja te kasnijeg spajanja se ponavlja sve dok svi segmenti nisu homogeni.

Nakon toga se značajke opisuje histogramom te se parovi s najboljim rezultatom uzimaju kao potencijalno odgovarajući parovi.

Također je navedeno da algoritam precizno i pouzdano izbacuje uljeze te se može koristiti i za istovremenu lokalizaciju i mapiranje (eng. SLAM).




Slika 3.5 *Primjer dijeljenja slike tijekom Split-and-Merge algoritma*

# Poglavlje 4

# Normal Distributions Transform (NDT)

## 4.1 Što je NDT?

Transformacija Normalne Raspodjele (slobodan prijevod) je algoritam koji se može koristiti za lokalizaciju robota, podudaranje skenova i mapiranje te je prepoznata kao jedna od glavnih tehnika za istovremenu lokalizaciju i mapiranje (eng. Simultaneous Localization And Mapping – SLAM) robota.

Prvi put je predložena u studenom 2003. godine u radu Petera Bibera i Wolfganga Straßera. U njemu je objašnjen navedeni algoritam, njegove moguće upotrebe, njegova važnost te razlika u odnosu na druge algoritme i radove. [10] Danas su prisutne već mnoge varijacije NDT-a koje se implementiraju u preko tisuću GitHub repozitorija.

### 4.1.1 Za što se koristi?

Dvije glavne uporabe algoritma NDT su lokalizacija robota u slučaju dane mape unutar koje se nalazi te mapiranje okoliša u slučaju dane lokalizacije (npr. GP-Som). Također može izvoditi relativno praćenje pozicije te istovremenu lokalizaciju i mapiranje.

U prijašnje navedenom radu je također dokazano da je algoritam sposoban vjerodostojno mapirati interijer u pravom vremenu, pritom ne koristeći odometriju[1].[10]

### 4.1.2 Kako funkcionira?

Ključni element ovog algoritma je reprezentacija referentnog skena. Umjesto da se podudaranje trenutnog skena izvodi direktno s točkama referentnog skena, modelira se vrijednost nalaženja površinske točke na određenoj poziciji linearnom kombinacijom normalnih distribucija.

To se radi na način da se prostor podijeli u jednake dijelove (npr. kvadrat od 2 metra kvadratna). Rezultat toga je dobivena gustoća vjerojatnosti koja se može spojiti s drugim skenom koristeći Newtonov algoritam. Zbog tog razloga se ne moraju odrediti međusobno odgovarajuće točke u oba skena.

Konkretno je riječ o koracima. Prije izvođenja koraka vrlo je važno podijeliti sken u jednake kvadrate konstantne veličine. Zatim za svaki kvadrat koji sadrži barem tri točke napraviti sljedeće korake:

1) Skupiti sve 2D točke u ovom kvadratu prema izrazu (4.1):

$$Xi = 1..n \tag{4.1}$$

[1]Odometrija je korištenje podataka od senzora pokreta za procjenu promjene pozicije kroz vrijeme

2) Izračunati prosjek q prema izrazu (4.2):

$$q = \frac{1}{n} \sum iXi \tag{4.2}$$

3) Izračunati matricu kovarijance (simbol za matricu kovarijance je kapitalno grčko slovo $\sigma$ odnosno sigma) prema izrazu (4.3):

$$\sum = \frac{1}{n} \sum i(Xi - q)(Xi - q)^t \tag{4.3}$$

Nakon toga je vjerojatnost mjerenje uzorka na 2D točki X u ovom kvadratu modelirana normalnom distribucijom N (q, $\sum$) prema izrazu (4.4):

$$p(x) \sim exp(-\frac{(x-q)^t \sum^{(} -1)(x-q)}{2}) \tag{4.4}$$

Kad je to gotovo, dobivena je gustoća vjerojatnosti te je sada podudaranje skenova lagano. Opet se radi u koracima radi jednostavnosti.

1) Inicirati parametre (kao nula ili koristeći odometriju).

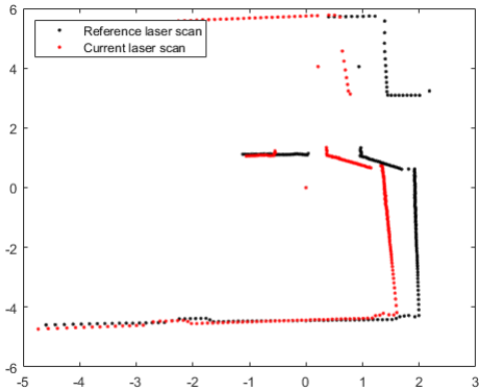
2) Mapirati jednu po jednu točku drugog skena u odnosu na gustoću vjerojatnosti prema parametrima.

3) Odrediti odgovarajuću normalnu distribuciju za svaku mapiranu točku.

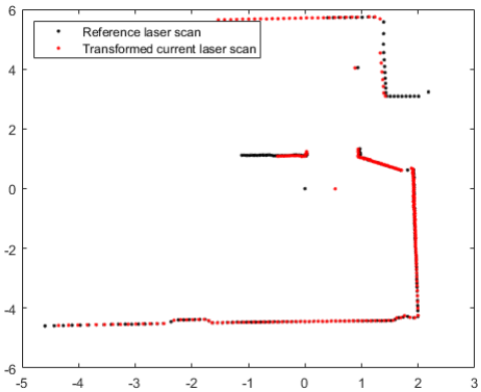
4) Odrediti novu vrijednost parametara izračunom odgovarajuće normalne distribucije i sumiranjem rezultata.

5) Za optimizaciju jednim korakom Newtonovog algoritma podesiti novu vrijednost parametara.

6) Vratiti se na korak 2 dok kriterij za konvergenciju nije ispunjen.[10]




Slika 4.1 *Primjer 2D skeniranja [11]*

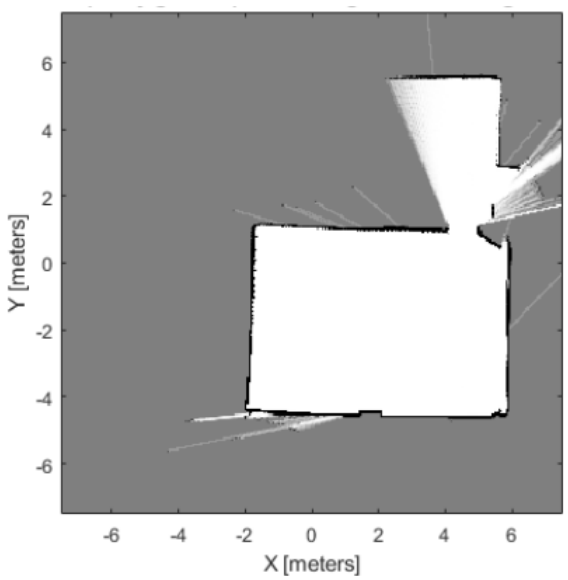



Slika 4.2 *Primjer 2D skeniranja nakon poravnavanja [11]*

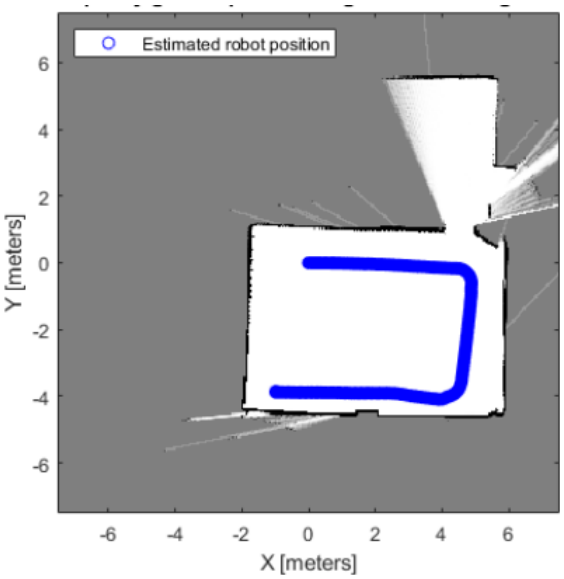
## 4.2 Specifikacije

### 4.2.1 Dimenzionalnost

Postoje verzije NDTa za 2D i za 3D podudaranje skeniranja. Naravno, 2D implementacija je brža zbog znatno manje računanja, pogotovo kod rotacije, dok je 3D implementacija češće viđena u praksi zbog njene intuitivnosti. Detaljne razlike između 2D i 3D NDT-a vidljive su u radu Martina Magnussona objavljenom u prosincu 2009 gdje, između ostalog, detaljno objašnjava formule i izračune potrebne za implementaciju NDT metode.[12]




Slika 4.3 *Prikaz mape kreirane od skenova [11]*




Slika 4.4 *Prikaz kretnje robota na kreiranoj mapi [11]*

# Poglavlje 5

# Iterative Closest Point (ICP)

## 5.1 Što je ICP?

Iterativna Najbliža Točka (dalje ICP) je algoritam koji se koristi u svrhu minimiziranja razlike između dva pointclouda odnosno oblaka točaka. Prvi put je osmišljen i predstavljen u radovima Yanga Chena i Gérarda Medionia [13] te P. Besla i Neila D. McKaya [14]. Postoje mnoge varijante ICP algoritma, ali najpopularnije su varijante točka-do-točke i točka-do-ravnine.

Razlog zašto postoji mnogo varijanti nalazi se u tome što se svakom varijantom pokušaju optimizirati svi dijelovi algoritma. Primjerice, uvijek postoji drugačija kombinacija rješenja za razne male probleme koje ICP algoritam treba riješiti, kao što su: Koje točke koristiti, kako definirati pogrešku, kako prepoznati i maknuti anomalije u rezultatima itd.

Postoje radovi u kojima se svaka varijanta ICP-a uspoređuje. Jedan takav rad je rad Szymona Rusinkiewicza i Marca Levoya iz 2001. godine [15] u kojem se i optimizira ICP koristeći kombinaciju različitih predložih varijanti.

### 5.1.1 Za što se koristi?

ICP se implementira u mapiranju i lokalizaciji robota, postizanju optimalnog planiranja puta te za rekonstrukciju 2D ili 3D površina.

Razvojem ICP-a i predloženim varijantama, vremenom se brzina izvođenja algoritma povećala. Ovo je bio važan napredak u robotici jer je značio da su ljudi mogli pustiti robota samog u određenu okolinu koja je možda preopasna za ljude i označavala bi rizik za sigurnost zaposlenika. Prije npr. lokalizacije robota, nije se moglo znati gdje unutar mape se robot nalazi te bi često dolazilo do oštećivanja robota ako bi podaci o poziciji dolazili prekasno.

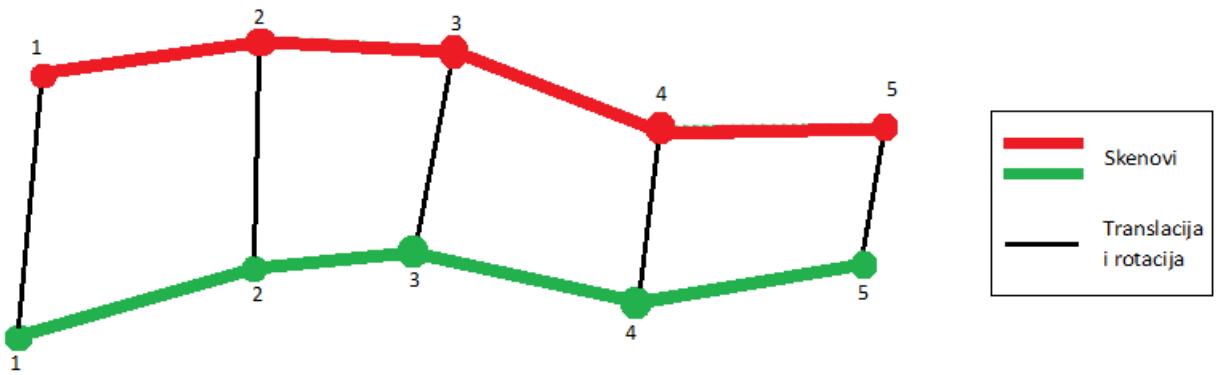
Također je bilo važno optimizirati ovaj algoritam da može u nekoliko sekundi usporediti skenove i ispravno vratiti korisniku mapu kako bi korisnik znao u kojem smjeru dalje krenuti.

### 5.1.2 Kako funkcionira?

Glavna ideja je da jedan oblak točaka, tzv. referentni oblak, ostaje fiksiran dok se drugi, novi oblak, transformira koristeći rotaciju i translaciju da bi se što bolje povezao s referentnim. Bolja povezanost označava minimiziranje udaljenosti korespondnih točaka. Ovo se najčešće radi zbrajanjem kvadrata razlike između koordinata spojenih parova. [16]

Konkretni koraci su:

1. Spajanje svake točke u novom oblaku s najbližom točkom referentnog oblaka
2. Procjena kombinacije rotacije i translacije koja je potrebna da bi se što bolje spojile točke spojene u prijašnjem koraku. Ovo se radi koristeći tehniku minimizacije kvadrata udaljenosti točaka.




Slika 5.1 *Primjer ICP algoritma [15]*

3. Transformacija točaka novog oblaka koristeći transformaciju procjenjenu u prijašnjem koraku
4. Iteracija odnosno ponavljanje

Komputacijski najteži dio zapravo je izračun najbližih točaka te u tome leži razlog zašto zna biti isplativije koristiti druge metode umjesto ICP-a, što naravno ovisi o vrsti problema s kojim se suočavamo.

# Poglavlje 6

# Testiranje implementacija podudaranja skenova u ROSu

U ovom poglavlju objašnjen je praktični dio, odnosno testiranje podudaranja skenova. Korak po korak se navodi kako na vlastitom računalu možete pokrenuti i analizirati implementacije podudaranja skenova.

## 6.1 Potrebne tehnologije

Za testiranje će nam biti potreban operacijski sustav Linux Ubuntu, ROS Noetic i robotski simulator Gazebo.

### 6.1.1 Ubuntu

Za ovo testiranje korišten je Ubuntu 20.04.5 LTS(Focal Fossa). Taj operacijski sustav može se instalirati preko poveznice `https://releases.ubuntu.com/20.04/`. Ja sam odabrao Desktop Image te sam taj file pokrenuo koristeći virtualnu mašinu na

programu Oracle VM.

Pokretanjem Ubuntua sam išao na "Install Ubuntu" te izabrao hrvatski kao jezik sustava i jezik za tipkovnicu. Odabrao sam minimalnu instalaciju, ali instalirao obje Ostale mogućnosti. U sljedećem izborniku sam odabrao "Obriši disk i instaliraj Ubuntu" obzirom da se radi o virtualnoj mašini koja nema ništa drugo. Dalje sam završio instalaciju odabirom vremenske zone te postavljanja imena i lozinke.

### 6.1.2 ROS

Instalaciju ROS distribucije Noetic pratimo sa poveznice `http://wiki.ros.org/noetic/Installation/Ubuntu`:

1) Konfigurirati Ubuntu repozitorije da dopuštaju Restricted, Universe i Multiverse pakete prateći poveznicu u fusnoti [1]

2) Konfigurirati ključeve naredbama:

```
sudo apt install curl
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt
```

3) Povući promjene naredbom:

```
sudo apt update
```

4) Instalirati ROS Noetic naredbom:

```
sudo apt install ros-noetic-desktop-full
```

[1] https://help.ubuntu.com/community/Repositories/Ubuntu

### 6.1.3 Gazebo

Gazebo je 3D robotski simulator čija je svrha vizualizacija i simulacija robotskih komponenti, algoritama upravljanja te raznih ideja za budući razvoj istih. Navedeni simulator nudi mnoge usluge kao što su: napredna 3D grafika, pristup senzorima, 2D i 3D kamerama, raznim modelima robota, simulaciju rada u oblaku te mnoge druge.

Također na svojoj stranici imaju ekstenzivne tutoriale, projekte i blogove kojima se možete priključiti da se u potpunosti upoznate s okruženjem, primjenama okruženja i novim vijestima koje su vezane uz Gazebo.

Gazebo je već instaliran jer je dio paketa ros-noetic-desktop-full koji smo prije instalirali, ali za zasebnu instalaciju samo je potrebno upisati naredbu:

```
sudo apt install gazebo11
```

## 6.2 Postupak testiranja

### 6.2.1 Priprema

Ove upute oblikovane su poveznicom[2] kako bi pripremili testiranje (na vrhu je potrebno odabrati Distribuciju ROSa):

1) Instalirati pakete ovisnosti naredbom:

```
sudo apt-get install ros-noetic-joy ros-noetic-teleop-twist-joy \
ros-noetic-teleop-twist-keyboard ros-noetic-laser-proc \
ros-noetic-rgbd-launch ros-noetic-rosserial-arduino \
ros-noetic-rosserial-python ros-noetic-rosserial-client \
ros-noetic-rosserial-msgs ros-noetic-amcl ros-noetic-map-server \
```

[2]https://emanual.robotis.com/docs/en/platform/turtlebot3/quick-start/

```
  ros-noetic-move-base ros-noetic-urdf ros-noetic-xacro \
  ros-noetic-compressed-image-transport ros-noetic-rqt* ros-noetic-rviz \
  ros-noetic-gmapping ros-noetic-navigation ros-noetic-interactive-markers
```

4) Instalirati TurtleBot3 pakete:

```
sudo apt install ros-noetic-dynamixel-sdk
sudo apt install ros-noetic-turtlebot3-msgs
sudo apt install ros-noetic-turtlebot3
```

5) Konfigurirati programsko okružje:

```
echo 'export TURTLEBOT3_MODEL=burger' >> ~/.bashrc
source ~/.bashrc
```

Nakon ovoga je sustav spreman za simulaciju.

### 6.2.2 Pokretanje

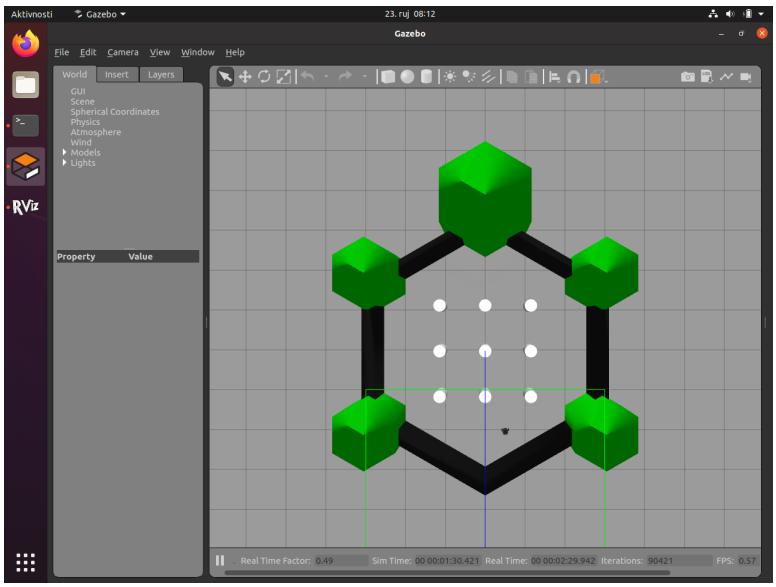
Terminal 1 (pokreće svijet s robotom u Gazebu):

```
source /opt/ros/noetic/setup.bash
roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

Nakon ovoga, na ekranu je vidljiv Gazebo simulator sa slike (6.1).

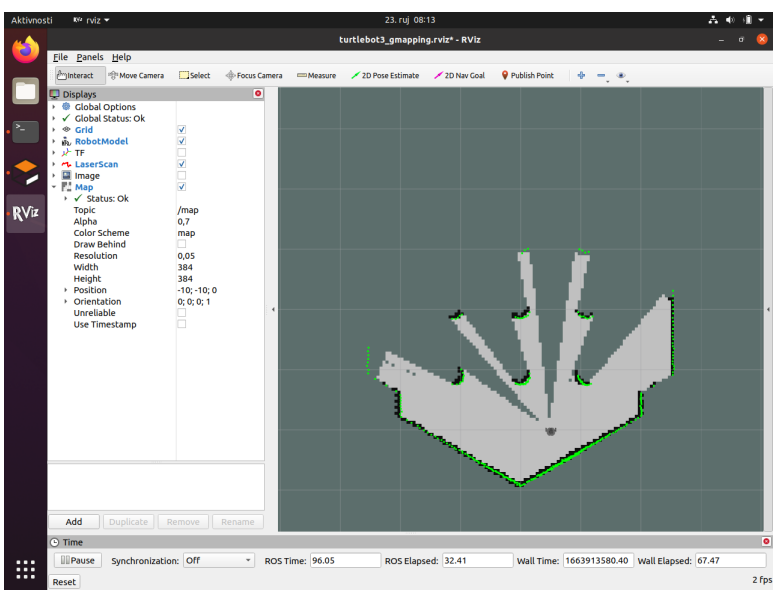
Terminal 2 (pokreće potrebni SLAM čvor):

```
source /opt/ros/noetic/setup.bash
roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping
```

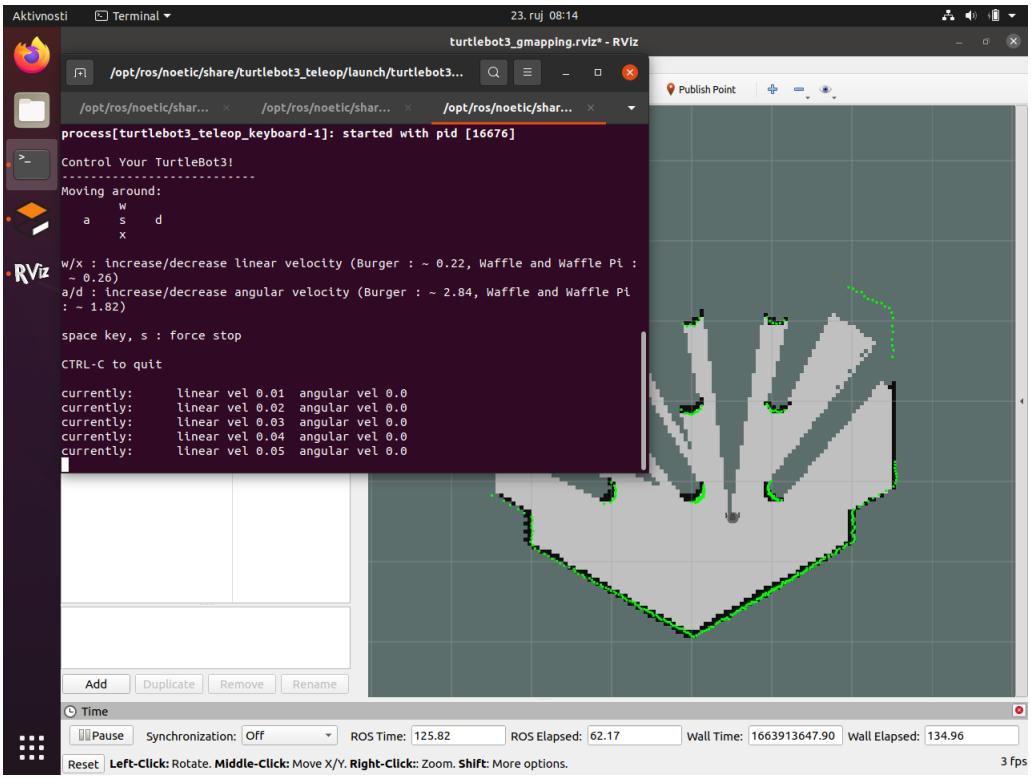



Slika 6.1 *Prikaz Gazeba*

Zatim, vidljiva je slika (6.2) na kojoj je prikazan Rviz, program koji omogućuje pogled robotove perspektive.




Slika 6.2 *Prikaz Rviz-a*




Slika 6.3 *Prikaz kretnje i mapiranja*

Terminal 3 (omogućava kretnju robota tipkovnicom):

```
source /opt/ros/noetic/setup.bash
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

Posljedično se otvara mogućnost pomicanja robota koristeći tipkovnicu. Slika (6.3) prikazuje kako robot aktivno mapira okoliš tijekom kretnji.

Već znamo da se podudaranje skenova koristi u mapiranju, te je zato u testiranju ideja prikazati kako brzina robota ovisi o kvaliteti mapiranja.

Postepeno će se brzina robota povećavati te će se doći do zaključka ovisi li to o kvaliteti dobivene mape. Pretpostavka je da bi se kvaliteta mape (odnosno površina koju otkrijemo skeniranjem) trebala smanjiti zbog veće brzine koja je uzrok ili manjeg

| Broj mjerenja | Brzina robota | Vrijeme robota | Kvaliteta mape |
|---|---|---|---|
| 1. | 0.1m/s | 35.22s | ∼98% |
| 2. | 0.3m/s | 12.06s | ∼95% |
| 3. | 0.5m/s | 7.27s | ∼87% |
| 4. | 0.7m/s | 5.43s | ∼80% |
| 5. | 0.9m/s | 3.95s | ∼45% |

Tablica 6.1 *Testiranje robota pri različitim brzinama*

broja skenova ili lošijih kuteva skenova u slučaju pravilno razmaknutih prepreka. Recimo da je robotu s lijeve strane ograda dvorišta koja je pola metra prisutna i zatim pola metra nije (i tako se ograda pravilno ponavlja). Zatim recimo da se robot kreće pola metra po sekundi i uzima sken svaku sekundu. Kao posljedica, robot uvijek skenira kada je pokraj ograde te zato nikad ne vidi potencijalnu opasnost direktno iza ograde.

U tablici testiranja (6.1), brzina robota mjeri se u metrima po sekundi (m/s), vrijeme robota označava prosječno vrijeme da robot pređe sa jedne strane mape na drugu dok ide ravno i izražava se u sekundama te se kvaliteta mape mjeri u postotku otkrivenosti mape gdje 100% označava u potpunosti otkrivenu mapu.

U svrhu preciznosti pri testiranju, u terminalu 3 je zamijenjena naredba

```
"roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch"
```

naredbom:

```
"rosrun rqt_robot_steering rqt_robot_steering"
```

jer daje veću kontrolu nad brzinom robota.

Po pretpostavci, u prva 4 mjerenja je robot postepeno imao lošije rezultate kvalitete mape te je čak u petom mjerenju imao znatno lošiji rezultat jer je stigao samo jednom uzeti sken umjesto dvaput kao i u ostalim mjerenjima.

# Poglavlje 7

# Usporedba NDT-a i ICP-a

Prva stvar koju je bitno znati je da se ni za jedan od ova dva algoritma ne može reći da je objektivno bolji. Razlog zašto se oba koriste leži u tome što svaki od njih ima svoju svrhu i primjenu. Detaljnim objašnjenjem algoritma ICP u poglavlju 5 i detaljnim objašnjenjem algoritma NDT u poglavlju 4 se može doći do zaključka da oba algoritma imaju svoje prednosti i mane prema kojima se u stvarnosti primjenjuju.

Primjerice, iz objašnjenja algoritama se može zaključiti da je NDT brži od ICP-a jer ICP sadrži kompleksne izračune za određivanje međusobno najbližih točaka za razliku od NDT-a, koji je baziran na vjerojatnosti pojavljivanja točke u ćelijama jednake veličine. Zbog navedene razlike u brzini, NDT se češće koristi u sustavima gdje je brzina vrlo bitna.

Primjer ovoga se može naći u automobilskoj industriji pri autonomnoj vožnji gdje je od krucijalne važnosti brzina algoritma, pogotovo pri velikim brzinama automobila.

S druge strane, ICP se koristi u sustavima gdje brzina nije od velike važnosti te nam je točnost prioritet. Primjeri ovoga uključuju generalno mapiranje okoliša i prostorija, ali kao konkretni primjer postoji istraživanje špilja koristeći robote. Ovo se radi kad je ulazak u određenu špilju vrlo rizičan ili nemoguć za čovjeka. Kada bi

bilo moguće poslati robotska vozila da autonomno mapiraju špilje i rudnike, moglo bi se sačuvati zdravlje i pružiti sigurnost životima tisuće rudara.

U navedenom primjeru razlika u sekundama ili minutama nije nimalo bitna. Bitna je samo ekstremna točnost i preciznost te se iz tog razloga radije koristi ICP za mapiranje u tom slučaju.

Naravno, ove razlike u navedenim algoritmima se konstantno mijenjaju zbog sve više predloženih varijanti tih algoritama, ali ovo služi kao gruba usporedba kako bi se čitatelj upoznao s osnovnim verzijama algoritama te uspješno započeo svoj daljnji put u istraživanje.

Detaljnija usporedba ovih algoritama se može naći u radu M. Magnussona, A. Nüchtera, C. Lörkena, A. J. Lilienthala i J. Hertzberga iz 2009. godine [17] u kojem se praktično uspoređuju algoritmi ICP, NDT te NDT s trilinearnom interpolacijom što je zapravo varijanta NDT-a predložena u tom radu.

Nadalje, u navedenom radu postoji referenca na rad M. Magnussona, A. J. Lilienthala i T. Ducketta iz 2007. godine [18] u kojem predlažu 3D-NDT, što je varijacija NDT algoritma optimizirana za mapiranje špilja. U tom radu dokazuju kako je 3D-NDT brži od ICP algoritma što je direktni dokaz da se ovi algoritmi mijenjaju.

Zato je kod velikih projekata od iznimne važnosti vrlo dobro istražiti i biti oprezan pri biranju algoritma koji će optimalno izvršiti željeni zadatak.

# Poglavlje 8

# Zaključak

Ovim radom čitatelj je uveden u područje robotike te spreman za daljnje samostalno istraživanje. U prvom dijelu rada objašnjeno je što je ROS, zašto je važan, koje vrste ROSa postoje te kako ROS funkcionira. Zatim je objašnjen koncept podudaranja skenova te načini i algoritmi kojima se podudaranje skenova može ostvariti. Nakon toga, u drugom dijelu rada posebno su testirani i razjašnjeni algoritmi Transformacija Normalne Raspodjele (eng. NDT - Normal Distributions Transform) te Iterativna Najbliža Točka (eng. ICP - Iterative Closest Point) nakon čega su i uspoređeni kako bi čitatelju bila jasna razlika među njima.

Rad je napisan koristeći akademske članke, podatke dostupne na internetu, stručne časopise, knjige i dokumentacije. Svi izvori navedeni su u literaturi.

# Bibliografija

[1] Wikipedia, "Robot Operating System", s Interneta, https://en.wikipedia.org/wiki/Robot_Operating_System, 5. travanj 2022.

[2] Open Robotics, "ROS Box Turtle", s Interneta, http://wiki.ros.org/boxturtle, 12. lipanj 2011.

[3] Wikipedia, "QR code", s Interneta, https://en.wikipedia.org/wiki/QR_code, 31. kolovoz 2022.

[4] Baby Scanning Boutique, "2D 3D 4D Ultrasound scanning Anglesey", s Interneta, https://3d4dbabyultrasoundscan.co.uk/cities/2d-3d-4d-ultrasound-scanning-an 18. kolovoz 2020.

[5] Wikipedia, "Lidar", s Interneta, https://en.wikipedia.org/wiki/Lidar, 20. rujan 2022.

[6] Hao, F.; Lei, Y.; Rui, Y.; Tao, W.: "An efficient scan-to-map matching approach for autonomous driving", 2016.

[7] Filotheou, A.: "Correspondenceless scan-to-map-scan matching of homoriented 2D scans for mobile robot localisation", Robotics and Autonomous Systems, broj 149, 19 stranica, 2022., https://arxiv.org/abs/2106.14003

[8] Ryu, K.; Dantanarayana, L.; Furukawa, T.; Dissanayake, G.: "Grid-based scan-to-map matching for accurate 2D map building, Advanced Robotics", 2016.

[9] Li, J.; Zhong, R.; Hu, Q.; Ai, M.: "Feature-Based Laser Scan Matching and Its Application for Indoor Mapping", Sensors, broj 16, verzija 8, članak 1265, 2016.

[10] Biber, P.; Straßer, W: "The Normal Distributions Transform: A New Approach to Laser Scan Matching", IEEE International Conference on Intelligent Robots and Systems (IROS 2003.), stranice 2743 - 2748, 2003.

[11] MathWorks, "Estimate Robot Pose with Scan Matching", s Interneta, `http://www.mathworks.com/help/nav/ug/estimate-robot-pose-with-scan-matching.html`, 28. rujan 2021.

[12] Magnusson, M.: "The Three-Dimensional Normal-Distributions Transform — an Efficient Representation for Registration, Surface Analysis, and Loop Detection", doktorska disertacija, 2009.

[13] Chen, Y.; Medioni, G: "Object modelling by registration of multiple range images", Image and Vision Computing, broj 10, verzija 3, stranice 145-155, 1992., `https://www.sciencedirect.com/science/article/pii/026288569290066C`

[14] Besl, P. J.; McKay, N. D.:"A method for registration of 3-D shapes", IEEE Transactions on Pattern Analysis and Machine Intelligence, broj 14, verzija 2, stranice 239-256, 1992.

[15] Rusinkiewicz, S.; Levoy, M.: "Efficient variants of the ICP algorithm", konferencija Proceedings Third International Conference on 3-D Digital Imaging and Modeling, 2001.

[16] Wikipedia, "Iterative closest point", s Interneta, `https://en.wikipedia.org/wiki/Iterative_closest_point`, 26. lipanj 2022.

[17] Magnusson, M.; Nuchter, A.; Lorken, C.; Lilienthal, A.; Hertzberg, J.: "Evaluation of 3D Registration Reliability and Speed - A Comparison of ICP and NDT", IEEE International Conference on Robotics and Automation (ICRA), stranice 3907 - 3912 , 2009.

[18] Magnusson, M.; Lilienthal, A.; Duckett, T.: "Scan Registration for Autonomous Mining Vehicles Using 3D-NDT", Journal of Field Robotics, broj 24, stranice 803-827, 2007.

# Popis slika

# Sažetak

U ovom radu objašnjeno je što je ROS, zašto je važan, koje vrste ROSa postoje te kako ROS funkcionira. Zatim je objašnjen koncept podudaranja skenova te načini i algoritmi kojima se podudaranje skenova može ostvariti. Nakon toga, u drugom dijelu rada posebno su testirani i razjašnjeni algoritmi Transformacija Normalne Raspodjele te Iterativna Najbliža Točka nakon čega su i uspoređeni kako bi čitatelju bila jasna razlika među njima.

***Ključne riječi*** **— ROS, podudaranje skenova, NDT, ICP**

# Abstract

This paper explains what ROS is, why it is important, what types of ROS exist and how ROS works. Then the concept of scan matching is explained as well as the methods and algorithms by which the matching of scans can be achieved. After this, in the second part of the paper, the Normal Distribution Transform (NDT) and Iterative Closest Point (ICP) algorithms were tested and clarified, after which they were compared so that the reader could understand the difference between them.

***Keywords*** **— ROS, scan matching, NDT, ICP**