Modeliranje prostora za analizu problema razmještaja osjetila

Banov, Marina

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij računarstva

Diplomski rad

MODELIRANJE PROSTORA ZA ANALIZU PROBLEMA RAZMJEŠTAJA OSJETILA

Rijeka, rujan 2022.

Marina Banov

0069083031

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij računarstva

Diplomski rad

MODELIRANJE PROSTORA ZA ANALIZU PROBLEMA RAZMJEŠTAJA OSJETILA

Mentor: prof. dr. sc. Kristijan Lenac

Rijeka, rujan 2022.

Marina Banov

0069083031

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 15. ožujka 2022.

Zavod: **Zavod za računarstvo**
Predmet: **Mobilna robotika**
Grana: **2.09.04 umjetna inteligencija**

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Marina Banov (0069083031)**
Studij: Diplomski sveučilišni studij računarstva
Modul: Programsko inženjerstvo

Zadatak: **Modeliranje prostora za analizu problema razmještaja osjetila / Environment modeling for sensor placement**

Opis zadatka:

Analizirati problem razmještaj osjetila. Proširiti model Turtlebot3 robota s Intel Realsense kamerom i izraditi modele prostora za korištenje u Gazebo simulatoru za validaciju spajanja oblaka točaka dobivenih dubinskom kamerom na robotu. Izraditi modele dvodimenzionalnog i trodimenzionalnog prostora iz spojenog oblaka točaka.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 21. ožujka 2022.

Mentor:

Prof. dr. sc. Kristijan Lenac

Predsjednik povjerenstva za diplomski ispit:

Prof. dr. sc. Kristijan Lenac

IZJAVA O SAMOSTALNOJ IZVEDBI RADA

Ovime izjavljujem da sam samostalno izradila diplomski rad pod vodstvom mentora prof. dr. sc. Kristijana Lenca.

_____

Marina Banov

ZAHVALA

Želim zahvaliti svojoj obitelji i dragim prijateljima na
ljubavi, podršci i ohrabrenju za vrijeme studija.

Zahvaljujem svom mentoru prof. dr. sc. Kristijanu Lencu
na znanju prenesenom na brojnim kolegijima i na stručnom vodstvu.

Posebno zahvaljujem dr. sc. Diegu Sušnju na svim savjetima i pomoći
prilikom izrade ovog rada. Hvala na inspiraciji i slušanju.

Na kraju veliko hvala svim kolegama, asistentima, profesorima i svima
koji su bili dio mog razvoja i uljepšali mi studij.

SADRŽAJ

1. UVOD

Mjerna osjetila su elektronički sklopovi koji pretvaraju prikupljene vrijednosti neke fizikalne veličine u električni signal pogodan za daljnju obradu [1]. Kada se takvi elektronički sklopovi nadograde baterijom, sustavom za obradu i sustavom za komunikaciju govorimo o osjetilima. Ovisno o veličini koju mjere (npr. temperatura, vlažnost, svjetlost, zvuk, kemijski sastav...), osjetila se mogu koristiti u rješavanju problema iz velikog broja područja. Osjetila su često ključan dio ugradbenih računalnih sustava, neophodna su za autonomno gibanje mobilnih robota prilikom estimacije položaja, a nalaze se i u pametnim telefonima.

Osjetila se u računalnim sustavima mogu koristiti samostalno, ali se kod složenijih zadataka pojavljuje potreba za suradnjom više istovrsnih ili raznovrsnih osjetila. Povezivanjem osjetila nastaje struktura zvana mreža osjetila kojoj je glavni cilj prikupiti veću količinu podataka na većem području. Najpoznatiji primjer primjene mreža osjetila su sigurnosni sustavi, odnosno sustavi za nadzor. Ako su osjetila povezana i komuniciraju korištenjem bežičnih tehnologija radi se o bežičnim mrežama osjetila koje su posebno važne za razvoj Interneta stvari (eng. Internet of Things, IoT). Bežične mreže osjetila mogu se koristiti za detekciju prirodnih katastrofa, šumskih požara ili razine zagađenja zraka, u medicini za praćenje pokazatelja zdravstvenog stanja [2] ili kao temelj za komunikaciju među autonomnim vozilima [3].

1.1. Problem umjetničke galerije

Prilikom projektiranja mreža osjetila potrebno je imati na umu neke od mogućih zahtjeva kao što su razmještaj i povezanost čvorova, pokrivenost prostora ili energetska učinkovitost sustava. U svakom će pojedinačnom slučaju biti potrebno optimizirati određene zahtjeve po cijenu onih koji u tom trenutku nisu prioritet. Za ovaj rad posebno je značajan zahtjev pokrivenosti prostora i problem razmještaja osjetila, koji se temelji na problemu umjetničke galerije, postavljenom 1973. godine [4, 5]. Pokrivenost je omjer prostora koji se nalazi unutar područja vidljivosti osjetila naspram cijelog prostora. Kod rješavanja tog problema, cilj je pronaći dovoljan broj osjetila te njihov optimalan razmještaj tako da poznati konačni zatvoreni prostor bude u potpunosti pokriven.

U prvom je koraku osmišljavanja vlastitog pristupa potrebno definirati modele osjetila i modele prostora.

U originalnoj je verziji problema umjetničke galerije, umjesto elektroničkih osjetila, cilj bio rasporediti čuvare po prostoru kako bi mogli u svakom trenutku nadzirati sve dijelove umjetničke galerije ili muzeja. Čuvari su u formaliziranom obliku bili definirani kao binarna svesmjerna statična osjetila koja se mogu nalaziti bilo gdje u prostoriji. Ponekad se takav model može postrožiti postavljanjem ograničenja na položaj osjetila (npr. dozvoljeno je postaviti osjetila samo duž bridova prostorije ili samo na vrhove) ili korištenjem usmjerenih osjetila s ograničenim kutom i dosegom vidljivosti. U nekim verzijama tog problema čuvari nisu statični već im je dozvoljeno kretanje po nekim segmentima prostora. Osjetila mogu biti binarna što znači da mogu deterministički detektirati sve objekte unutar dosega vidljivosti, ali mogu imati i drugačiji, probabilistički model osjetilne sposobnosti kao što su stepenasti, eksponencijalni ili stepenasto-eksponencijalni [6]. U sklopu ovog rada, uzelo se u obzir realna ograničenja nekih od osjetila dostupnih na tržištu. Zbog energetske se učinkovitosti posebna pozornost obratila svesmjernim osjetilima s oznakom Bluetooth niske energije (eng. Bluetooth Low Energy, BLE).

Rješavanje problema razmještaja osjetila tražit će drugačiji pristup ovisno o načinu na koji je definiran prostor koji pokušavamo pokriti. Prostori se često pojednostavljuju do razine tlocrta, odnosno dvodimenzionalnih granica. Rješenja se mogu razlikovati ovisno o obliku prostora, primjerice za konkavne i konveksne poligone, za ortogonalne poligone i poligone s prazninama (praznine u tom slučaju odgovaraju preprekama kao što je namještaj ili arhitektonski elementi u zatvorenom prostoru). Predstavljanje prostora kao poligona u ravnini u suštini znači gubitak značajnog broja informacija o stvarnim uvjetima. Kako bi se prostor prikazao što je vjernije moguće i time dobilo što točnije rješenje, koriste se trodimenzionalni modeli koji uzimaju u obzir i način na koji visina prepreka utječe na polje vidljivosti iz neke točke. Za očekivati je, međutim, da će u tim slučajevima pronalazak rješenja biti računski složeniji proces.

1.2. Ciljevi rada

Ovaj rad razmatra kako stvarne prostore zapisati u obliku kakav je predstavljen u radu [7], kako bi se taj model razmještaja osjetila mogao implementirati i na primjerima domova i institucija u kojima svakodnevno obitavamo, ne samo u hipotetskim slučajevima.

Predloženi model prostora određuje skup prepreka i slobodnog prostora. U poopćenom modelu pogodnom za analizu na dvodimenzionalnoj razini, elementi prostora opisani su kao bazni poligoni koji odgovaraju tlocrtu elementa. U slučajevima u kojima je važna i treća dimenzija, opis elemenata prostora proširuje se zadanim visinama na kojima taj element počinje i

završava i tako geometrijski lik postaje tijelo. Svi elementi prostora promatraju se u zajedničkom globalnom koordinatnom sustavu.

Ciljevi ovog rada bili su:

1. snimiti željene prostore uz pomoć mobilnih robota i dubinskih kamera,
2. izdvojiti i adekvatno zapisati sve elemente unutar tih prostora te
3. provjeriti učinkovitost modela razmještaja osjetila na dobivenim modelima prostora.

Ciljevi su realizirani kroz tri slijedna koraka.

Prvi je korak bilo prikupljanje podataka u nekom od formata koji mogu vjerno opisati trodimenzionalne prostore s naglaskom na oblak točaka (eng. point cloud). Ovaj se rad zbog tehničkih ograničenja uočenih u početnoj fazi zasniva na simuliranim prostorima. Proces snimanja prostora, bilo da se izvodi u simuliranom ili stvarnom okruženju, potrebno je pažljivo isplanirati i provesti kako bi konačan rezultat bio spojen i kompletan oblak točaka što će omogućiti i olakšati nastavak rada. U sklopu ovog rada pripremljeno je programsko rješenje za registraciju oblaka točaka [8].

Drugi je korak modeliranje interijera iz dobivenih snimaka što je značajan zadatak, ne samo u analizi problema razmještaja osjetila, već i u arhitekturi, digitalnom očuvanju kulturne baštine i u video igricama zasnovanima na virtualnoj stvarnosti. Zbog široke primjene i važnosti problema, neka rješenja za ovaj problem već postoje. Jedan je od prvih radova koji su pokušali automatizirano izdvojiti tlocrte iz oblaka točaka objavljen 2010. godine [9]. Dvije godine kasnije, autori rada [10] predstavili su automatizirani sustav dobivanja 3D modela interijera iz oblaka točaka koji koristi plohe kao primitive za prikaz glavnih arhitektonskih struktura. Rezultat primjene ovog pristupa je jednostavan 3D model interijera koji se sastoji isključivo od zidova, podova, stropova i stubišta. Rad [11] se također fokusirao na rekonstrukciju samo glavnih strukturnih elemenata uz dodatno semantičko grupiranje u sobe. Ta rješenja, međutim, u ovom slučaju ne bi bila potpuna jer je potrebno uspješno prepoznati i prepreke koje se nalaze u prostoru, što može biti izazovno s obzirom na to koliko su arhitektonske strukture dominantne u odnosu na namještaj. U ovom je radu izrađen novi algoritam [12] kako bi nadoknadio navedeni nedostatak, iako je pristup problemu bio sličan radovima iz literature.

Treći je korak uslijedio kada su se iz oblaka točaka izdvojili svi elementi u željenom obliku. Proveden je eksperiment kojim se pokušala utvrditi uspješnost modela razmještaja osjetila na kompleksnijim prostorima. Korišteno je šest optimizacijskih algoritama u svrhu maksimizacije pokrivenosti prostora te je promatrano kolika se pokrivenost postiže s do 50

osjetila. Provedena je statistička analiza nad dvjema zavisnim varijablama, postignutom relativnom pokrivenosti prostora i vremenom izvršavanja.

1.3. Struktura rada

Drugo poglavlje opisuje korištene alate i tehnologije te proces pripreme mobilnog robota i virtualnih prostora u simulacijskom okruženju. U trećem poglavlju, nakon kratkog teorijskog pregleda o percepciji dubine, predstavljen je problem registracije oblaka točaka koji se pojavljuje u procesu snimanja prostora i novo rješenje za taj problem. Četvrto poglavlje sadrži algoritam za izradu modela prostora iz oblaka točaka te provjeru uspješnosti takvog algoritma. U petom se poglavlju dobiveni modeli koriste za analizu problema razmještaja osjetila i usporedbu optimizacijskih algoritma putem statističke analize rezultata ranije spomenutog eksperimenta.

2. SIMULACIJSKO OKRUŽENJE

Prvi je korak u procesu modeliranja prostora za opisani problem bio postaviti radno okruženje koje će omogućiti pripremu i prikupljanje dovoljne količine relevantnih primjera na kojima će algoritam kasnije biti testiran. Prikupljanje podataka odnosno trodimenzionalnih snimaka prostora postignuto je uz pomoć mobilnog robota i dubinske kamere.

U fazi spajanja dubinske kamere i robota došlo je do tehničkih ograničenja koja su diktirala daljnje korake u razvoju. Iako je u teoriji moguće jednostavno spojiti fizičku dubinsku kameru na robota koristeći univerzalnu serijsku sabirnicu (eng. Universal Serial Bus, USB), u praksi je širina pojasa bila nepremostiva prepreka. Problemi koji nastaju prilikom povezivanja fizičke kamere i robota putem USB kabela nisu neuobičajeni jer dubinske kamere šalju veliku količinu podataka na računalo, što zahtjeva upotrebu visokokvalitetnog i visokopropusnog kabela. Zbog nemogućnosti korištenja fizičke kamere i robota, pripremljeno je simulacijsko okruženje.

Simulacijsko okruženje omogućilo je rad unatoč ograničenjima fizičke opreme te je ubrzalo proces razvoja i testiranja. Radno okruženje temeljilo se na robotskom operacijskom sustavu (eng. Robot Operating System, ROS). ROS nije operacijski sustav u tradicionalnom značenju pojma jer ne obavlja sve zadaće operacijskog sustava (npr. upravljanje procesima), već bi točnije bilo reći da se radi o programskoj podršci otvorenog koda za rad s robotima. ROS pruža apstrakciju sklopovlja i druge usluge za strukturiranu komunikaciju heterogenog skupa računala [13]. U ovom je radu korištena stabilna ROS2 Foxy distribucija, Python programski jezik i druge programske knjižnice i alati otvorenog koda kao što su Gazebo simulator i RViz alat za vizualizaciju.

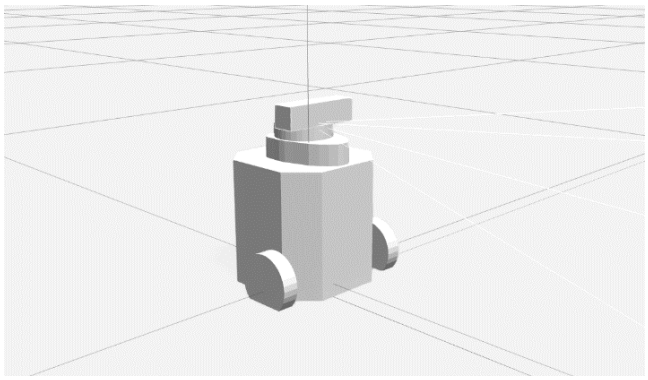
Posebno je važan bio Gazebo simulator zbog toga što nudi velik broj gotovih modela robota, prostora i predmeta, ali i mogućnost dizajniranja vlastitih robota i virtualnih prostora unutar intuitivnog grafičkog sučelja. Gazebo ima i druge prednosti kao što su realističan prikaz scena, nekoliko naprednih fizikalnih modela i jako dobra podrška za rad u ROS-u, što znači da je kod pisan za simulator prenosiv i na fizičku opremu [14].

U simulacijskom je okruženju bilo potrebno pripremiti virtualnog robota kojim će biti moguće upravljati i koji će moći uspješno prikupljati i obrađivati podatke s dubinske kamere. Također je bilo potrebno pripremiti reprezentativne virtualne prostore u kojima će se robot kretati.

2.1. TurtleBot3 robot s Intel RealSense kamerom

Za izvršavanje željenog zadatka izabran je mobilni robot TurtleBot3, i to model Burger. TurtleBot3 se često koristi u obrazovanju, istraživanjima i hobističkoj robotici jer je to malen, programabilan, lako nadogradiv i pristupačan (iako to nije presudan faktor kada se radi o virtualnim modelima u simulacijskom okruženju) model [15]. Od percepcijskih osjetila ovaj model ima integrirano samo lasersko daljinsko osjetilo (eng. Laser Distance Sensor, LDS). To nije bilo dovoljno za potrebe ovog rada, pa je mobilni robot nadograđen dubinskom kamerom Intel RealSense D435i. Načelo funkcioniranja dubinskih kamera bit će detaljnije pojašnjeno u sljedećem poglavlju, a za sada je jedino bitno spomenuti da je cilj pomoću dubinske kamere snimiti prostore i izgraditi oblak točaka.

Prikupljanje podataka postignuto je upravljanjem robotom, prolaskom kroz cijeli prostor unutar Gazebo simulacijskog okruženja i čitanjem podataka dobivenih s dubinske kamere. Modeli TurtleBot3 robota koji su spremni za korištenje nakon instalacije osnovnih paketa dolaze s ograničenim brojem osnovnih osjetila te je za svaku prilagođenu upotrebu potrebno nadograditi te modele. U ovom je slučaju korištena Intel RealSense dubinska kamera, što je zahtijevalo uređivanje zadanog modela robota. Takav je unaprijeđeni robot dio paketnog rješenja i prikazan je na Slici 2.1.

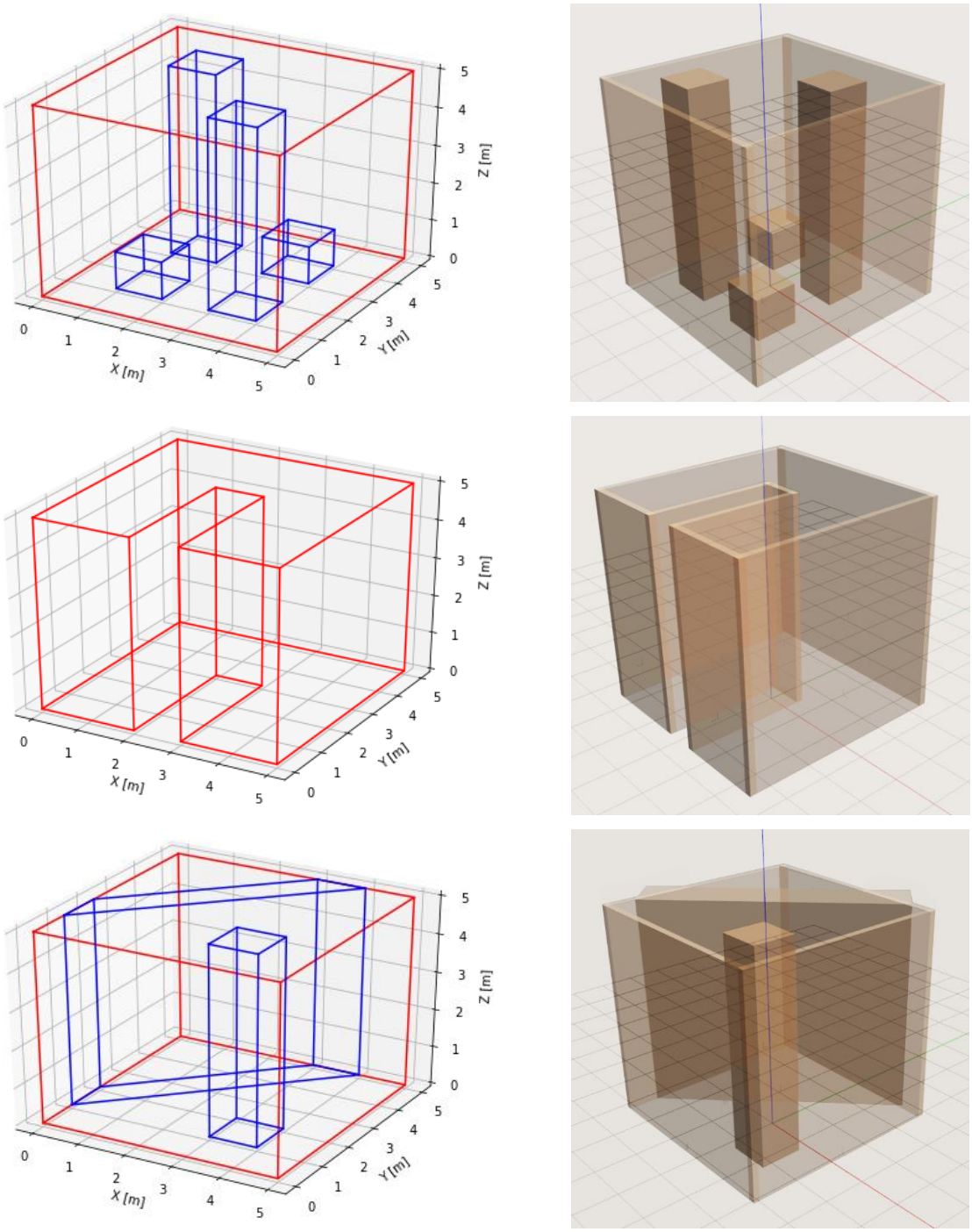


Slika 2.1. TurtleBot3 s Intel RealSense kamerom u Gazebo okruženju

Ispravno postavljeno radno okruženje koje će omogućiti upravljanje takvim virtualnim robotom sadrži osnovne knjižnice i alate za rad (eng. Software Development Kit, SDK) s Intel RealSense kamerama [16], sučelje (eng. wrapper) za komunikaciju s RealSense kamerama u ROS-u [17] te RealSense Gazebo dodatak (eng. plugin) [18].

2.2. Modeli prostora

Definirana su tri virtualna trodimenzionalna prostora po uzoru na rad [7] u kojima će se TurtleBot3 robot kretati i graditi oblak točaka. Prostori su relativno jednostavni, ali predstavljaju neke važne slučajeve kao što su konkavni i neortogonalni prostori. Slika 2.2. prikazuje navedene prostore kako su definirani u literaturi i kako su ostvareni u Gazebo simulatoru.



Slika 2.2. Modeli prostora u Gazebo okruženju (env1, env2 i env3)

Kao primjer složenijeg prostora korišten je jedan od postojećih modela u Gazebo simulatoru, TurtleBot House, koji se sastoji od šest prostorija i hodnika, a sadrži i namještaj koji će biti potrebno modelirati kao prepreke. Ovaj je model prikazan na Slici 2.3. i posebno je zanimljiv zbog većeg broja elemenata prostora i većih dimenzija prostora, što je odlika stvarnih interijera za koje se pokušava odrediti optimalni razmještaj osjetila.

Slika 2.3. Primjer složenog prostora u Gazebo okruženju (TurtleBot House)

Nakon instalacije svih potrebnih knjižnica i alata, uspješnog nadograđivanja TurtleBot3 robota s dubinskom kamerom Intel RealSense D435i te dizajniranja modela prostora, bilo je potrebno prikupiti podatke s dubinske kamere i obraditi ih u spojeni oblak točaka.

3. OBRADA OBLAKA TOČAKA

3.1. Percepcija dubine i oblak točaka

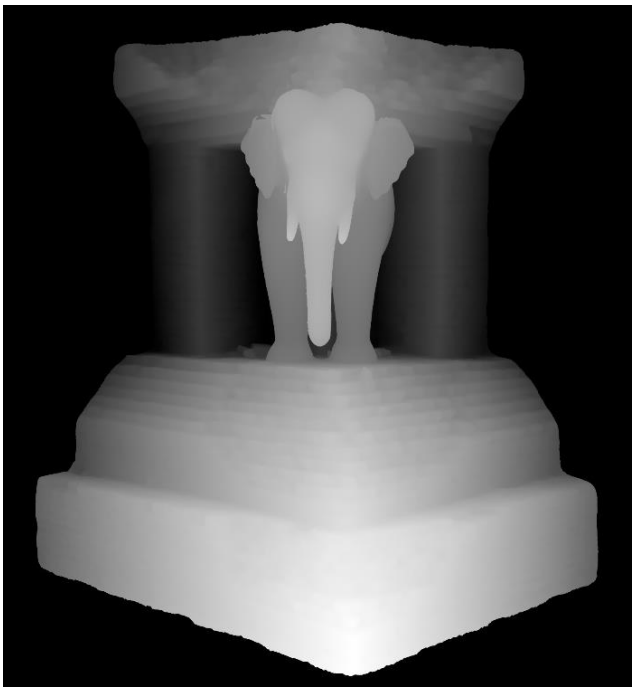
Mobilni roboti kao što je TurtleBot3 u pravilu su opremljeni s više percepcijskih osjetila koja mjere stanja okoline robota i pružaju korisne informacije za lokalizaciju i/ili izgradnju mape prostora. Neki od najpoznatijih vrsta osjetila koje se koriste u tu svrhu su sonari, lidari i dubinske kamere. Intel RealSense D435i je primjer dubinske kamere.

Samo sklopovlje takvih uređaja temelji se na teoriji osjeta i percepcije i oponaša ljudski vid za prepoznavanje dubine. Postoji niz različitih znakova prema kojima prepoznajemo dubinu objekata u sceni, a razlikujemo monokularne i binokularne znakove dubine [19]. Monokularni znakovi dubine koje možemo spoznati samo s jednim okom su preklapanje, sjene, atmosferska i linearna perspektiva, relativne i apsolutne (naučene) veličine objekata i gradijent tekstura. Binokularni znakovi dubine koje percipiramo s oba oka temelje se na disparitetu slika iz lijevog i desnog oka i konvergenciji očiju. Drugi naziv za percipiranje binokularnih znakova dubine je stereoskopski vid.

Za razliku od običnih kamera, dubinske (ili stereoskopske) kamere sastoje se od dvije leće koje imitiraju dva ljudska oka i na taj se način oslanjaju na binokularne znakove dubine. Informacija o dubini pojedinih objekata ili piksela se izračunava prema udaljenosti između optičkih centara lijeve i desne leće, žarišnoj udaljenosti i disparitetu piksela. Što je veća razlika u pripadajućim pikselima na lijevoj i desnoj slici, to je objekt bliži promatraču. Primjer tog efekta je vidljiv na Slici 3.1. (uočiti razliku u položaju lijevog kaktusa na dvije slike).

Slika 3.1. Lijeva i desna slika stereoskopske kamere [19]

Dubinske kamere računaju disparitet svakog piksela na slikama koje dobivaju iz lijeve i desne leće, mapiraju ga u ograničeni raspon vrijednosti i pohranjuju u obliku mape dispariteta. Mape dispariteta ili dubinske mape su crno-bijele dvodimenzionalne slike na kojima svjetliji dijelovi opisuju piksele koji se nalaze bliže, a tamniji dijelovi udaljenije piksele. Iz njih se dobiva poseban format zapisa trodimenzionalnih modela zvan oblak točaka. Primjer jedne dubinske mape je vidljiv na Slici 3.2.



Slika 3.2. Dubinska mapa [20]

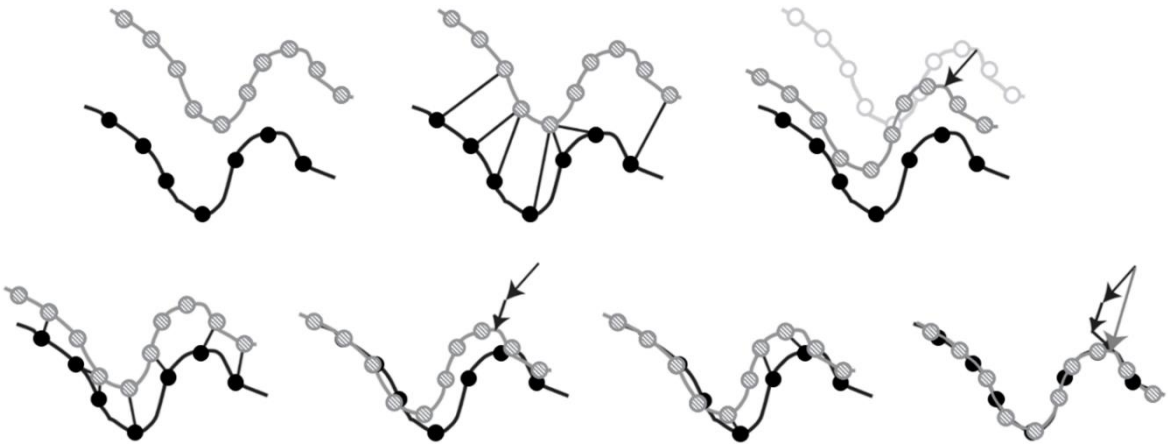
Kao i kod ostalih percepcijskih osjetila, očekivani su nedostaci dubinskih kamera računska složenost obrade informacija i problemi interpretacije percepcijskih informacija. U ovom slučaju, s obzirom na to da se TurtleBot3 robot kreće u prostoru dok ga snima, problem nastaje zbog promjene položaja u vremenu. Koordinate u oblaku točaka koji korisnik dobiva u svakom koraku relativne su u odnosu na lokalni koordinatni sustav kamere, a ne u odnosu na globalni koordinatni sustav. To znači da je potrebno transformirati točke u svakom kadru kako ne bi došlo do preklapanja u konačnom oblaku točaka.

U nastavku će biti predstavljene neke od uobičajenih metoda za spajanje oblaka točaka i novo programsko rješenje za upravljanje robotom i istovremenu izgradnju kompletnog oblaka točaka.

3.2. Spajanje oblaka točaka

Predstavljeni problem pronalaska transformacija s ciljem smislenog spajanja oblaka točaka poznat je kao problem podudaranja preslika (eng. scan matching) ili registracija oblaka točaka (eng. point cloud registration). Taj problem ima široke primjene u autonomnoj vožnji, 3D rekonstrukciji, detekciji objekata, istovremenoj lokalizaciji i mapiranju (eng. Simultaneous Localization And Mapping, SLAM), kreiranju panoramskih slika te virtualnoj i proširenoj stvarnosti. Algoritmi koji se koriste za rješavanje podudarnosti snimaka dijele se u dvije skupine ovisno o pristupu. Jedna skupina algoritama temelji se na pretpostavci da je moguće prvo odrediti značajne točke na pojedinim preslikama, a onda pronaći odgovarajuće parove značajnih točaka i tako spojiti dva kadra. Druga skupina algoritama istovremeno pronalazi značajne točke i potrebne transformacije.

Jedan od najpoznatijih algoritama u području registracije oblaka točaka zove se iterativna najbliža točka (eng. Iterative Closest Point, ICP), prvi puta predstavljen 1992. godine [21]. Kao što ime algoritma sugerira, radi se o iterativnom postupku koji u svakom koraku pokušava minimizirati udaljenost između točaka dviju skupina podataka. Slika 3.3. ilustrira nekoliko koraka u pokušaju podudaranja dva skupa točaka od kojih je jedan referentan i statičan, a drugi je pomičan, odnosno za njega pronalazimo transformacije koje bi maksimizirale podudaranja. U svakom se koraku za neke točke pomičnog skupa izračunava najmanja udaljenost od nekih točaka referentnog skupa. Potom se pronalazi transformacija koja bi minimizirala udaljenost težišta odabranih točaka iz oba skupa podataka [22]. ICP algoritam u početnim iteracijama konvergira brže, a kasnije nešto sporije, ali općenito postiže jako dobre performanse i zbog toga se često koristi u robotici.



Slika 3.3. ICP algoritam

S obzirom na to da je u ovom slučaju želja izgraditi potpuni oblak točaka dok korisnik upravlja robotom, korištene tehnologije nude jednu olakotnu okolnost: u ROS-u je u svakom trenutku poznata odometrija robota, odnosno pomak robota u odnosu na početnu točku. To znači da nije potrebno iterativno pronalaziti moguća podudaranja, već su transformacije iz lokalnog u globalni koordinatni sustav lako dostupne i izračun se svodi na samo jedan korak: vektorsko množenje liste očitanih točaka s transformacijskim matricama.

Potrebno je naglasiti da određena rješenja za problem transformacije oblaka točaka prema položaju i orijentaciji robota već postoje [23], te se mogu smatrati pouzdanim izvorima za ROS1 i/ili C++ čvorove. U trenutku pisanja ovog rada, knjižnice namijenjene ROS2 Python paketima nisu bile spremne za produkciju [24], te je predstavljeno rješenje [8] nastalo iz potrebe za savladavanjem tog nedostatka.

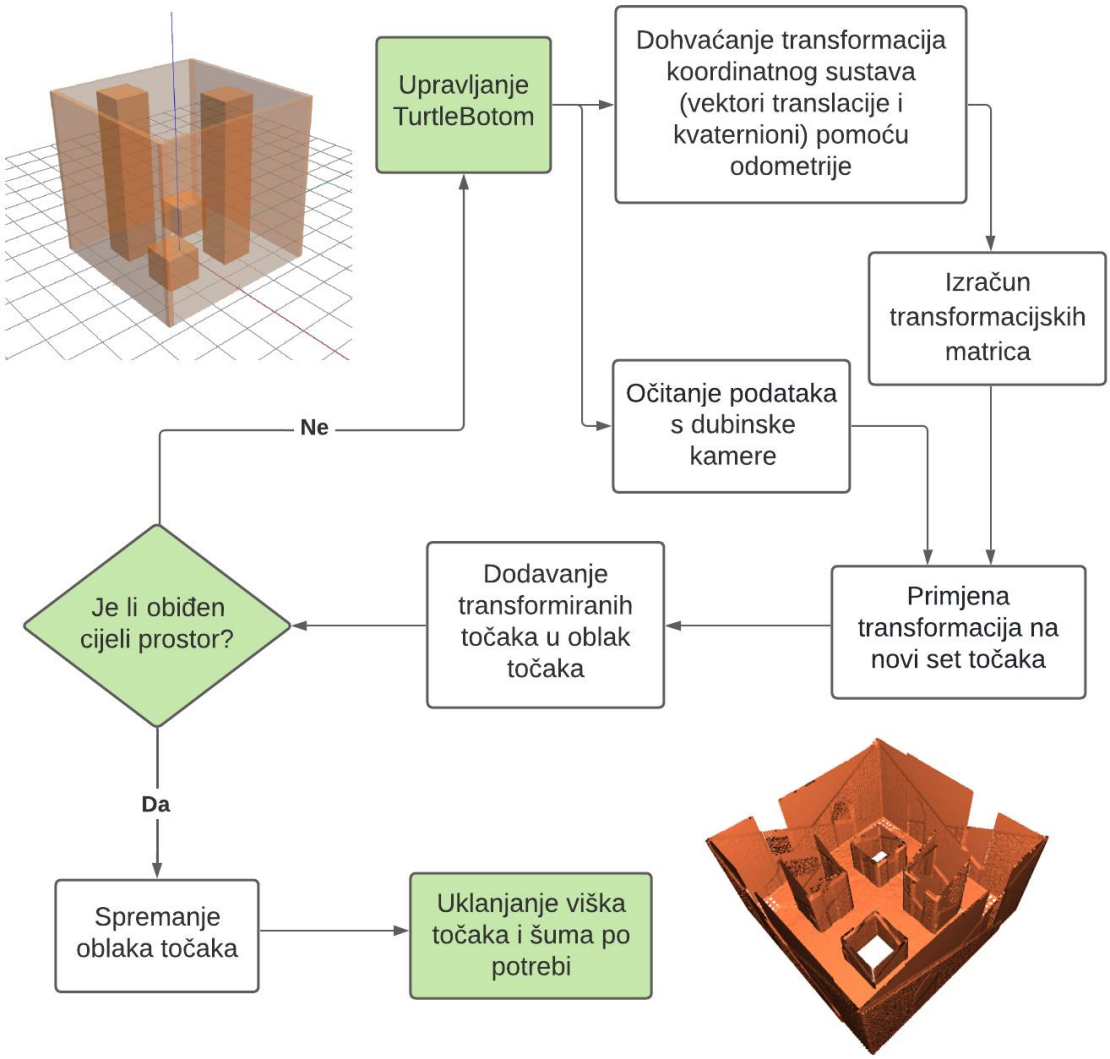
U nastavku slijedi pregled razvoja rješenja i ključnih koraka algoritma za registraciju oblaka točaka.

3.3. Pregled razvoja rješenja

Razvijen je ROS2 paket pisan u Python programskom jeziku koji omogućuje kreiranje spojenog oblaka točaka za vrijeme prolaska TurtleBot3 robota po prostorima definiranima u potpoglavlju 2.2. ROS paketi su kolekcije izvršnih i ostalih pomoćnih datoteka. Pokrenute instance izvršnih datoteka nazivaju se čvorovi. Čvorovi međusobno komuniciraju putem tema. Paketi mogu sadržavati posebnu vrstu datoteka pod nazivom lansirne datoteke koje pokreću više čvorova i služe za postavljanje parametara.

Kod pokretanja lansirne datoteke korisnik prvo mora definirati u kojoj od zadanih okolina će se robot inicijalizirati postavljanjem argumenta `world`. Lansirna datoteka će potom pokrenuti čvorove iz ostalih paketa: Gazebo simulator, RViz alat za vizualizaciju, TurtleBot3 robota i OctoMap server o kojemu će nešto kasnije biti riječi. Korisnik mora samostalno pokrenuti upravljački čvor `teleop_keyboard` i čvor `get_pcd` koji je zadužen za transformaciju kadrova i njihovo spajanje u jedinstveni oblak točaka. Kako bi se osiguralo učinkovitije korištenje resursa, čvor `get_pcd` dodaje samo diskretan skup kadrova u završni oblak točaka. Na korisniku je da upravlja robotom, kreće se po prostoriji i pošalje jednostavan signal onda kada se robot nalazi na položaju s kojeg želi obraditi podatke s dubinske kamere i dodati ih u oblak točaka.

Slika 3.4. ilustrira algoritam za spajanje oblaka točaka po kojem funkcionira čvor `get_pcd`. Na dijagramu toka zelenom bojom označeni su neautomatizirani koraci kojima upravlja korisnik, odnosno koji zahtijevaju unos korisnika ili njegovu procjenu.



Slika 3.4. Algoritam za spajanje oblaka točaka

3.4. Očitanje podataka

Podaci s osjetila u ROS-u najčešće se prenose putem specijaliziranih tokova podataka, tema. Čvorovi koji generiraju podatke objavljuju ih na za to predviđene teme, a čvorovi koji žele očitati podatke pretplaćuju se na te teme.

Čvor `get_pcd` pretplaćen je na temu `/depth/color/points` na kojoj Intel RealSense dubinska kamera konstantno objavljuje oblak točaka relativan u odnosu na lokalni koordinatni

sustav kamere. Također je putem instance klase `TransformListener` pretplaćen na temu na kojoj dohvaća promjenu položaja robota preko odometrije, odnosno podatke koji predstavljaju transformaciju iz lokalnog koordinatnog sustava u globalni koordinatni sustav. Pretplata na te teme postavlja se u konstruktoru čvora, kao što je prikazano na Kodnom isječku 3.1.

```
from rclpy.node import Node
from tf2_ros.buffer import Buffer
from tf2_ros.transform_listener import TransformListener
from sensor_msgs.msg import PointCloud2

class GetPcdNode(Node):
    def __init__(self):
        super().__init__("get_pcd")
        self.points = set()
        self.buffer = Buffer()
        self.tf_listener = TransformListener(self.buffer, self)
        self.create_subscription(
            PointCloud2, "/depth/color/points", self.process_frame, 5
        )
```

Kodni isječak 3.1. Konstruktor glavnog ROS čvora [8]

Kada Intel RealSense kamera objavi nove podatke na temu `/depth/color/points`, pozvat će se funkcija za obradu podataka. Korisnik mora prilikom prolaska po virtualnom prostoru poslati čvoru `get_pcd` signal kada želi pohraniti snimku iz nekog položaja u potpuni oblak točaka.

3.5. Transformacija oblaka točaka

Transformacija točaka iz lokalnog koordinatnog sustava dubinske kamere u globalni koordinatni sustav vrši se u funkciji povratnog poziva (eng. callback function) `process_frame` klase `GetPcdNode`. Skraćena je verzija te metode priložena u Kodnom isječku 3.2., koji sadrži dohvaćanje transformacija iz odometrije, čitanje podataka s dubinske kamere i dodavanje transformiranih točaka u oblak točaka `points`. Dan je i teorijski pregled matričnog računa, a njegova implementacija temelji se na programskoj knjižnici Transformations [25].

```
from rclpy.duration import Duration
import numpy as np
from sensor_msgs_py.point_cloud2 import read_points
from tf2_ros import TransformException

SOURCE = "odom"
TARGET = "camera_depth_optical_frame"
timeout = Duration(seconds=0)

def process_frame(self, data):  # unutar klase GetPcdNode
    try:
        time = data.header.stamp
        self.buffer.can_transform(TARGET, SOURCE, time, timeout)
        trans = self.buffer.lookup_transform(TARGET, SOURCE, time)
    except TransformException as _:
        self.get_logger().warning(f"Could not transform")
        return
    Rq = trans.transform.rotation
    Tv = trans.transform.translation
    pcd_data = np.array(list(read_points(
        data, field_names=['x', 'y', 'z'], skip_nans=True
    )))
    P = np.ones((pcd_data.shape[0], 4))  # dodaj 4. stupac
    P[:, :-1] = pcd_data
    # Matrični račun ...
    self.points.update(list(map(lambda t: (t[0], t[1], t[2]), P)))
```

Kodni isječak 3.2. Transformacija oblaka točaka [8]

Prilikom očitanja podataka odometrije, dobiveni su vektor translacije `Tv` i kvaternion rotacije `Rq`. Ti će podaci biti ključni za izvod matrice translacije T i matrice rotacije R koje se množe s oblakom točaka P' relativnog u odnosu na lokalni koordinatni sustav radi dobivanja oblaka točaka P relativnog u odnosu na globalni koordinatni sustav:

$$P = R \cdot T \cdot P'. \tag{3.1}$$

Vektor translacije `Tv` sadrži x, y i z koordinate položaja kamere u globalnom koordinatnom sustavu. Matrica translacije T dobiva se prema izrazu:

$$T = \begin{bmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -y \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.2}$$

Kvaternion rotacije `Rq` poseban je oblik zapisa orijentacije kamere. Kvaternioni su matematički koncepti vrlo korisni za opisivanje rotacija u tri dimenzije, posebno u onim granama računarstva gdje je važna mogućnost brzog izračuna velikog broja jednostavnih aritmetičkih operacija, kao što su računalna grafika ili robotika. Kvaternioni se sastoje od četiri komponente: tri imaginarne, x, y i z, i jedne realne komponente w. Iako operacije s kvaternionima mogu ukloniti neke pogreške koje se pojavljuju kada se rotacija u trodimenzionalnom prostoru računa uz pomoć Eulerovih kuteva oko x, y i z osi, one zahtijevaju i dublje razumijevanje složenije kvaternionske algebre [26]. Radi lakšeg računanja, kvaternion se pretvara u Eulerove kuteve prema izrazu:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} arctan\dfrac{2\ wx + yz}{1 - 2\ x^2 + y^2} \\ arcsin(2\ wy - xz\ ) \\ arctan\dfrac{2\ wz + xy}{1 - 2\ y^2 + z^2} \end{bmatrix}. \tag{3.3}$$

Nakon toga se definiraju matrice rotacije za x, y i z os, i konačno matrica rotacije R:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\phi & -sin\,\phi & 0 \\ 0 & sin\phi & cos\,\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.4}$$

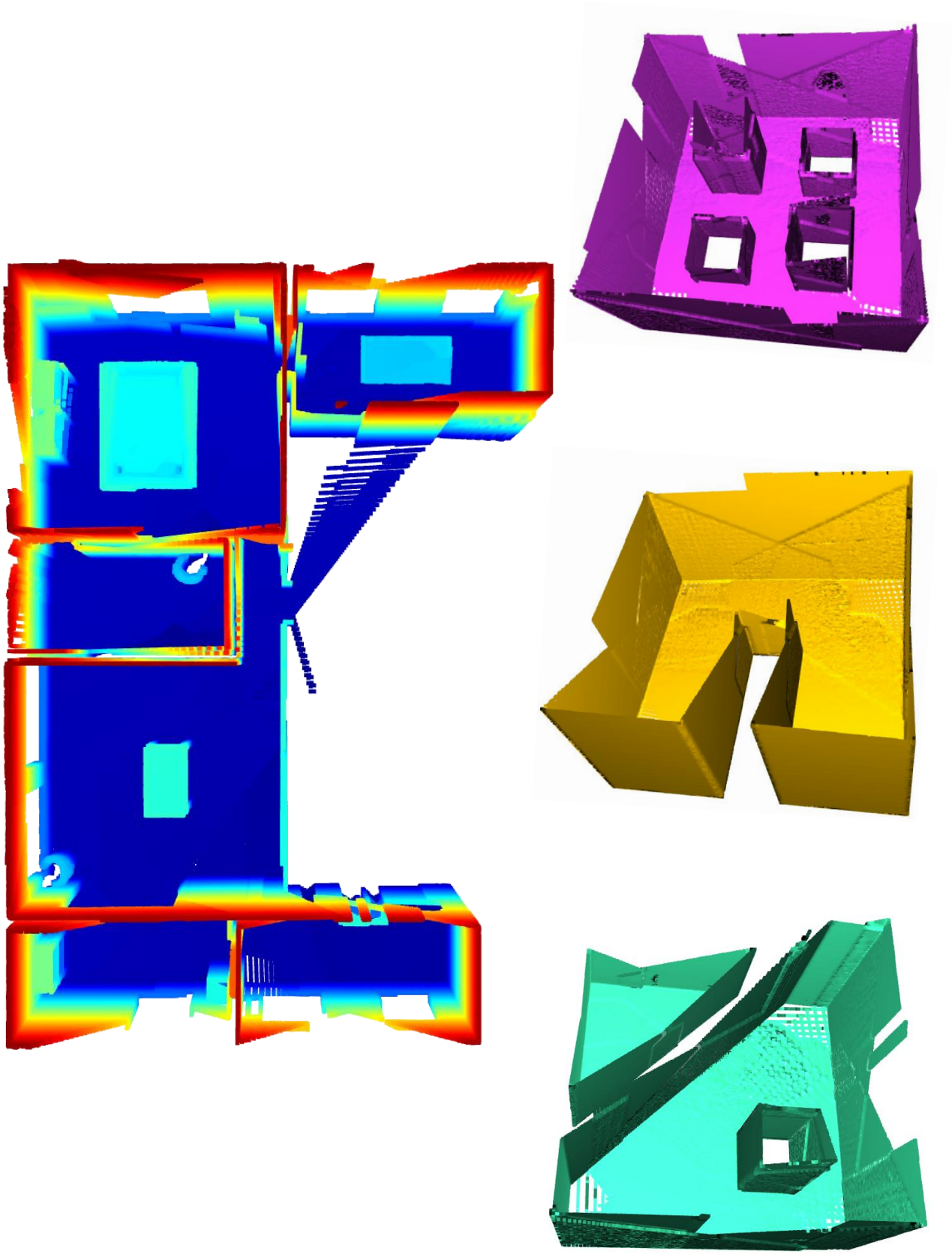
$$R_y = \begin{bmatrix} cos\,\theta & 0 & sin\ \theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\,\theta & 0 & cos\,\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.5}$$

$$R_z = \begin{bmatrix} cos\psi & -sin\,\psi & 0 & 0 \\ sin\,\psi & cos\,\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.6}$$

$$R\ =\ R_x \cdot R_y \cdot R_z. \tag{3.7}$$

Tako transformirane točke se u svakom kadru dodavaju u spojeni oblak točaka.

Rezultati predstavljene metode transformacije oblaka točaka vidljivi su na Slici 3.5.



Slika 3.5. Konačni oblaci točaka

3.6. OctoMap

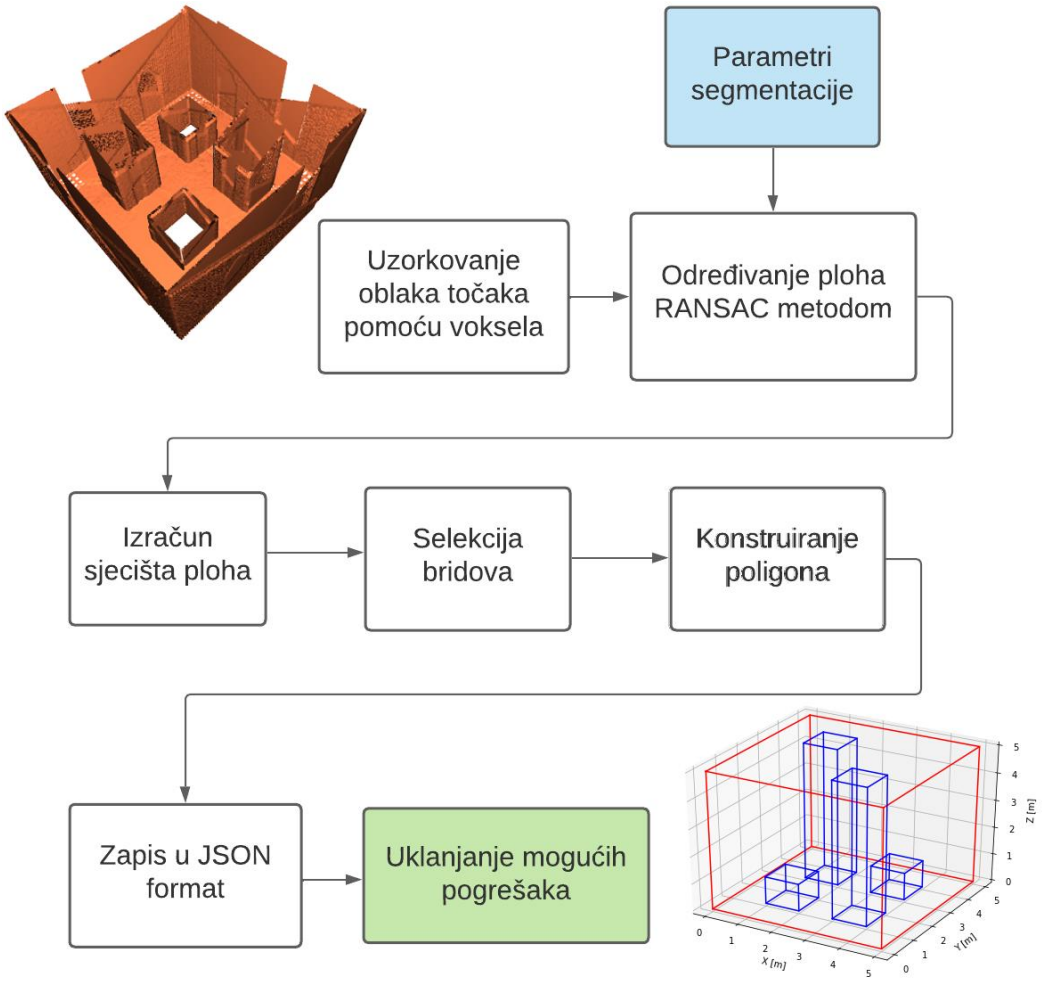
Ranije je rečeno da lansirna datoteka inicijalizira i pokreće čvorove, a među njima je naveden i OctoMap server. Zahvaljujući programu imena OctoMap, za vrijeme prolaska robota po virtualnim prostorima osim oblaka točaka kreira se još jedan format zapisa trodimenzionalnih podataka, oktostablo (eng. octree) [27]. OctoMap koristi probabilističku procjenu popunjenosti kako bi iz podataka koji mogu sadržavati i šum mapirao mrežu popunjenosti (eng. occupancy grid mapping) i eksplicitno definirao zauzeta, prazna i nepoznata područja. Istovremeno koristi učinkovite metode za kompresiju kako bi konačno oktostablo zauzimalo što manje memorije.

Ovaj bi format mogao biti koristan u budućim istraživanjima trodimenzionalnih modela prostora ili u nadogradnjama algoritma za izradu modela prostora koji će biti predstavljen u nastavku, ako količina memorije ili vrijeme izvršavanja postane ograničavajući faktor.

4. IZRADA MODELA PROSTORA

Iz oblaka točaka nastalog spajanjem kadrova koje je prikupila Intel RealSense dubinska kamera za vrijeme prolaska TurtleBot3 robota po simuliranom okruženju potrebno je izraditi apstraktni model prostora. Željeni model približno opisuje promatrani prostor kroz elemente slobodnog prostora i prepreka, te je prvi puta predstavljen 2020. godine [28]. Svaki element definiran je baznim poligonom, odnosno svojim tlocrtom, a u trodimenzionalnom modelu prostora taj se geometrijski lik omeđen baznim poligonom istisne u geometrijsko tijelo koje počinje i završava na zadanim visinama globalnog koordinatnog sustava.

Postupak izdvajanja i adekvatnog zapisivanja elemenata prostora prikazan je dijagramom toka na Slici 4.1. Korak *Uklanjanje mogućih pogrešaka* označen je zelenom bojom jer zahtijeva unos korisnika ili njegovu procjenu, odnosno nije automatiziran. Parametri segmentacije također se ne definiraju automatski već uz procjenu korisnika te su zato istaknuti plavom bojom.



Slika 4.1. Algoritam za izradu modela prostora

Implementacija algoritma [12] uvelike se temelji na knjižnici oblaka točaka (Point Cloud Library, PCL) programskoj podršci otvorenog koda za osnovne 3D funkcionalnosti, često korištenoj u istraživanjima na području robotike. PCL nudi rješenja za probleme poput registracije, filtriranja i segmentacije oblaka točaka, prepoznavanja objekata (eng. feature estimation) i rekonstrukcije površina [29]. Algoritam je implementiran u Python programskom jeziku, a s obzirom na to da je PCL razvijan na C++ programskom jeziku, korišteno je i odgovarajuće sučelje (eng. binding) za PCL [30].

Program je razvijan s ciljem da što vjernije modelira ortogonalne prostore s ortogonalnim rasporedom prepreka, kao što su env1, env2 i TurtleBot House sa Slika 2.2. i 2.3. Primjer env3 odstupa od tog ograničenja zbog dijagonalno postavljene prepreke, ali kako bi se dobilo što bolje rješenje, razvijeno je i Jupyter Notebook [31] grafičko sučelje u kojem korisnik ima veću kontrolu nad parametrima segmentacije.

U nastavku slijedi detaljniji opis svakog od koraka algoritma za izradu modela prostora iz spojenog oblaka točaka, kratki prikaz Jupyter Notebook sučelja i analiza učinkovitosti rješenja.

4.1. Uzorkovanje oblaka točaka

Uzorkovanje oblaka točaka bio je nužan korak prema poboljšanju performansi rješenja. U ovom je slučaju uzorkovanje, odnosno smanjenje količine podataka uz zadržavanje najbitnijih značajki, ostvareno korištenjem mreže voksela i PCL-ove klase `VoxelGrid` kako je prikazano u Kodnom isječku 4.1. U pozadini se preko ulaznog oblaka točaka kreirala trodimenzionalna mreža podijeljena na osnovne jedinice prostora, voksele, i sve su točke pojedinog voksela bile aproksimirane njihovim centroidom. Aproksimacija središtem voksela bila bi brža, ali ovakav pristup točnije predstavlja oblak točaka. Odabrani su vokseli veličine 5 cm × 5 cm × 5 cm.

```
def downsample(cloud, leaf_size=0.05):
    vg = cloud.make_voxel_grid_filter()
    vg.set_leaf_size(leaf_size, leaf_size, leaf_size)
    return vg.filter()
```

Kodni isječak 4.1. Metoda za uzorkovanje oblaka točaka [12]

Smanjenjem broja točaka efektivno se smanjila veličina ulazne datoteke u memoriji i kompleksnost izračuna u svim narednim koracima, a time i vrijeme izvršenja. Tablica 4.1. sadrži podatke o broju točaka u oblaku točaka i vremenu izvršenja programa bez i s uzorkovanjem, na tri primjera modela prostora. U stupcu *S uzorkovanjem* u zagradama su izraženi i postotci u odnosu na stupac *Bez uzorkovanja*.

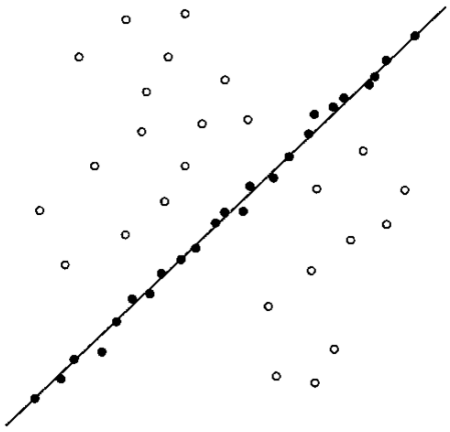
Tablica 4.1. Utjecaj uzorkovanja na broj točaka i vrijeme izvršavanja

	Bez uzorkovanja		**S uzorkovanjem**	
	Broj točaka	**Vrijeme [s]**	**Broj točaka**	**Vrijeme [s]**
env1	4 928 686	139,1986	59 479 (1,2%)	2,2449 (1,6%)
env2	2 561 572	53,9095	42 606 (1,7%)	1,0778 (1,9%)
env3	3 550 078	124,7061	46 459 (1,3%)	1,4421 (1,2%)

4.2. Određivanje ploha

Model prostora koji je predstavljen u radu [7] razlikuje slobodan prostor i prepreke te opisuje pojedine elemente kao geometrijske likove. Sama definicija željenog konačnog formata implicirala je prepoznavanje granica objekata i podjelu ulaznog oblaka točaka na podskupove, odnosno korištenje nekog od algoritama za segmentaciju. Segmentacija je proces koji omogućuje prepoznavanje i grupiranje sličnih podataka na način pogodan za buduću analizu.

U kontekstu računalnog vida i obrade slika i oblaka točaka najpoznatiji algoritam za segmentaciju je algoritam nasumičnog konsenzusnog uzorka RANSAC (eng. RANdom SAmple Consensus, RANSAC), nastao 1981. godine [32]. RANSAC je iterativna metoda za procjenu parametara matematičkog modela koji dobro opisuje zadani skup točaka uz istovremenu detekciju točaka koje se ne uklapaju u taj model i koje ne bi trebalo uključiti u izračun. U svakom koraku nasumično se odabire podskup izvornih podataka iz kojeg se izračunava hipotetski model. Ako se ostale točke u skupu podataka po nekoj funkciji gubitka mogu opisati tim modelom, one ulaze u konsenzusni skup. Iz iteracije u iteraciju, zadržava se onaj model koji je postigao najveći konsenzusni skup. Rezultat ovog procesa prikazan je na Slici 4.2. Vjerojatnost uspjeha algoritma povećava se s brojem iteracija, ali ovisi i o udjelu iznimaka u podacima i o još nekim parametrima. Otkada je metoda objavljena, razvijeno je i nekoliko alternativa kojima je cilj nadoknaditi nedostatke RANSAC-a. PCL nudi implementaciju svih tih metoda za segmentaciju oblaka točaka ali u većini je slučajeva preporučeno koristiti RANSAC.



Slika 4.2. Primjer RANSAC segmentacije

PCL-ova klasa `SACSegmentationFromNormals` pogodna je za ekstrakciju linija, ravnina, krugova, sfera, valjaka i stožaca. Elemente u korištenim primjerima najbolje bi opisao kvadar, ali budući da nije dostupna podrška za takav model korišten je model ravnina s definiranom normalom jer je spajanjem više ravnina moguće opisati kvadar. Kodni isječak 4.2. prikazuje metodu koja koristi PCL podršku za RANSAC segmentaciju plohe. Potrebno je proslijediti oblak točaka i x, y i z komponentu normale. Moguće je proslijediti i argumente koji će utjecati na izračun normala točaka, maksimalni broj iteracija RANSAC algoritma, dozvoljeno odstupanje točaka od izračunatog modela i dozvoljeno kutno odstupanje plohe od normale.

```
def ransac_plane(cloud, x, y, z, **kwargs):
    kwargs = {
        'ksearch': 10,
        'max_iterations': 1000,
        'distance_threshold': 0.1,
        'eps_angle': 0.035,  # otprilike 2°
        **kwargs,
    }

    seg = cloud.make_segmenter_normals(ksearch=kwargs['ksearch'])
    seg.set_method_type(pcl.SAC_RANSAC)
    seg.set_model_type(pcl.SACMODEL_PERPENDICULAR_PLANE)
    seg.set_optimize_coefficients(True)
    seg.set_max_iterations(kwargs['max_iterations'])
    seg.set_distance_threshold(kwargs['distance_threshold'])
    seg.set_axis(x, y, z)  # normala
    seg.set_eps_angle(kwargs['eps_angle'])
    return seg.segment()
```

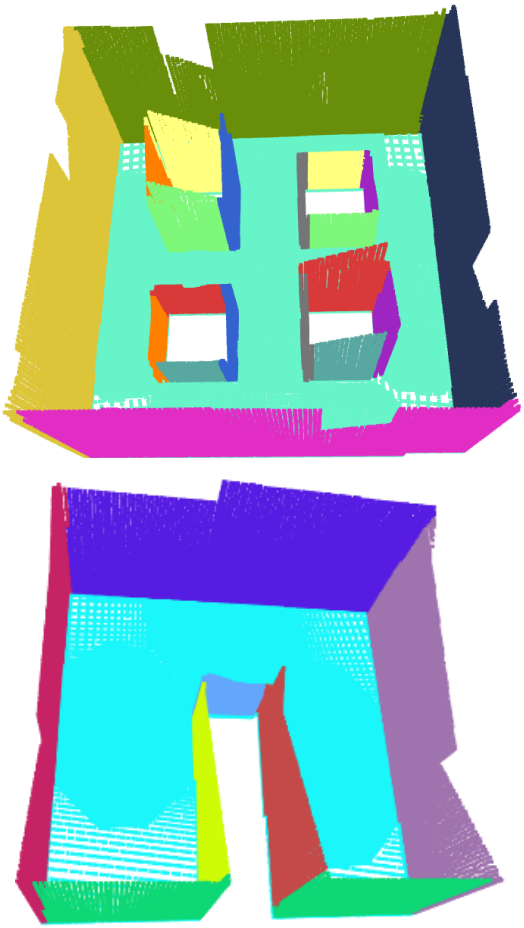
Kodni isječak 4.2. Metoda za RANSAC segmentaciju ravnina [12]

Funkcija `ransac_plane` se u glavnom programu poziva više puta: segmentiranje započinje s jednom ravninom koja je okomita na z os (ravnina poda), a potom se naizmjence segmentiraju ravnine okomite na x i na y os. Metoda vraća niz koeficijenata analitičkog oblika jednadžbe segmentirane ravnine koji je dan u sljedećem izrazu i listu indeksa svih točaka u oblaku točaka koje pripadaju toj ravnini:

$$Ax + By + Cz + D = 0. \tag{4.1}$$

Dobiveni podskup točaka mora se izdvojiti i ukloniti iz početnog oblaka točaka kako ne bi ometao buduće korake segmentacije i kako bi se mogao odrediti završetak koraka segmentacije. Segmentacija više nije moguća kada su sve točke početnog oblaka točaka uspješno klasificirane. Rezultati svake segmentacije dodaju se u listu `planes` (koeficijenti jednadžbi ravnina) i u listu `segmented_cloud` (podskupovi točaka).

Slika 4.3. pokazuje rezultate segmentacije oblaka točaka na primjerima env1 i env2 iz kojih je vidljivo uspješno prepoznavanje i izdvajanje ravnina za glavne elemente prostora.

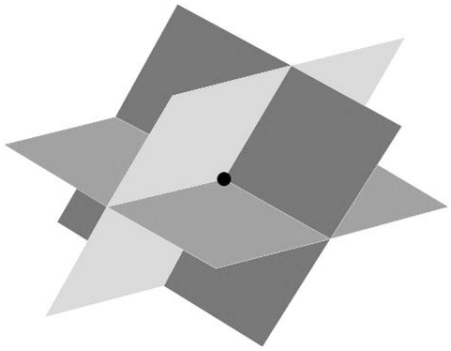


Slika 4.3. Primjeri segmentiranih oblaka točaka

4.3. Izračun sjecišta ploha

Nakon određivanja ploha unutar oblaka točaka slijedio je korak izračuna sjecišta ploha, odnosno točaka koje će u konačnom rezultatu predstavljati vrhove poligona koji opisuju prostor i prepreke. Bitno je naglasiti da je u prethodnom koraku iz oblaka točaka izdvojena samo jedna ravnina okomita na z os, ravnina tla, a sve ostale segmentirane ravnine okomite su na nju i s njom dijele pravac. Tlocrt svakog elementa u prostorima koji su uzeti kao primjer bit će ortogonalni poligon. Svaki vrh tih geometrijskih likova predstavlja sjecište triju ravnina – jedne okomite na x os, jedne okomite na y os i ravnine tla.

Bilo koje tri ravnine mogu se sjeći u jednoj točki, u beskonačno mnogo točaka kada je njihov presjek pravac ili ravnina ili niti u jednoj točki kada su ravnine paralelne. Od ta tri slučaja, onaj koji će za problem pronalaska vrhova poligona biti najznačajniji jest prvi, ilustriran na Slici 4.4.



Slika 4.4. Sjecište triju ravnina u jednoj točki [33]

Problem pronalaska sjecišta triju ravnina rješava se analitički kao sustav triju jednadžbi s tri nepoznanice [34]. Uz poznate koeficijente triju jednadžbi ravnina (jednadžbe ravnine tla Π_0, jednadžbe ravnine okomite na x os Π_i i jednadžbe ravnine okomite na y os Π_j), rješavanjem sljedećeg sustava dobivaju se x, y i z koordinate sjecišta:

$$\begin{aligned} A_0x + B_0y + C_0z + D_0 &= 0 \\ A_ix + B_iy + C_iz + D_i &= 0 \\ A_jx + B_jy + C_jz + D_j &= 0. \end{aligned} \tag{4.2}$$

Ovakav sustav se računalom najjednostavnije rješava u matričnom obliku jednom kada se definira matrica sustava A, stupac slobodnih koeficijenata b i vektor nepoznanica x:

$$\begin{bmatrix} A_0 & B_0 & C_0 \\ A_i & B_i & C_i \\ A_j & B_j & C_j \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -D_0 \\ -D_i \\ -D_j \end{bmatrix}. \tag{4.3}$$

Implementacija tog izračuna uz pomoć NumPy programske knjižnice [35] prikazana je u Kodnom isječku 4.3.

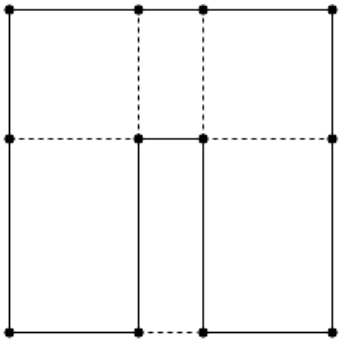
```
import numpy as np

def intersection(planes):
    corners = []
    for i in range(1, len(planes), 2):
        for j in range(2, len(planes), 2):
            A = [planes[0][:3], planes[i][:3], planes[j][:3]]
            b = [-planes[0][3], -planes[i][3], -planes[j][3]]
            corners.append(list(np.linalg.solve(A, b)))
    return corners
```

Kodni isječak 4.3. Metoda za izračun sjecišta ploha [12]

Skup rješenja predstavlja pojednostavljenje problema u svim budućim koracima jer su podaci kojima se raspolaže prebačeni iz trodimenzionalne u dvodimenzionalnu domenu. Iako je za svaku točku izračunata i z koordinata, treba imati na umu da su sve točke dobivene kao sjecište jedne te iste ravnine okomite na z os s nekom ravninom okomitom na x os i nekom ravninom okomitom na y os. Zbog toga bi z koordinata svih točaka trebala biti jednaka. Eventualne razlike među točkama nastale su zbog mogućih kutnih odstupanja u koraku segmentacije ravnina te su zanemarive.

4.4. Selekcija bridova i konstruiranje poligona

Neuređeni skup točaka dobivenih u prethodnom koraku nije dovoljan za jednoznačno definiranje poligona. Slika 4.5. pokazuje na koje se sve načine mogu povezati vrhovi dobiveni kao sjecište okomitih ravnina u oblaku točaka za primjer env2 i važnost pravilnog odabira bridova. Pune linije povezuju sve vrhove i točno opisuju model prostora. Iscrtanim linijama označeni su bridovi koji također povezuju dobivene vrhove ali koje ne bi trebalo uzeti u obzir.



Slika 4.5. Odabir bridova poligona uz zadani neuređeni skup vrhova

Za rješavanje ovog problema prvo je potrebno utvrditi uzastopne parove točaka koji se nalaze na istom pravcu. Potom se za svaki takav par točaka provjerava ako u segmentiranom oblaku točaka postoje točke koje nisu dio ravnine poda i koje bi se mogle nalaziti na tom bridu. Ako je utvrđeno da one postoje, onda se taj brid proglašava dijelom poligona te se pohranjuje najniža i najviša z koordinata pronađenih točaka, što će biti potrebno za definiciju modela prostora u trodimenzionalnom koordinatnom sustavu. U protivnom se radi o slučaju kakav opisuju iscrtane linije na Slici 4.5. i takav brid odbacujemo.

S ovim je korakom dobivena lista svih bridova u tlocrtu prostora. Sljedeći je korak objediniti te bridove u poligone koji će predstavljati zasebno granice prostora i prepreke.

Konstruiranje poligona iz zadane liste bridova je iterativan proces. Iz liste bridova uzima se jedan brid. Prva točka tog brida bit će početna točka novog poligona. Potom se u listi bridova traži sljedeći brid koji sadrži drugu točku prvog brida i on se dodaje u poligon. Taj se korak ponavlja sve dok se poligon ne zatvori, odnosno dok se ne odabere drugi brid koji sadrži početnu točku. Ako je poligon zatvoren, a u listi bridova je ostalo još elemenata, započinje proces konstruiranja novog poligona s nekim od bridova u listi. Za svaki se poligon pohranjuje i najmanja i najveća z koordinata svih bridova jer one označavaju najnižu i najvišu točku objekta kojeg taj poligon opisuje.

4.5. Zapis u JSON format

Zadnji automatizirani korak algoritma je pohranjivanje dobivenih poligona u JavaScript zapisu objekata (eng. JavaScript Object Notation, JSON) kako je zadano u Kodnom isječku 4.4. Eventualne pogreške korisnik će morati ručno ispraviti. Ovaj korak je prilično jednostavan, ali jedini trenutak u kojem treba biti oprezan je prilikom definiranja nekog poligona kao granice

prostora ili kao granice prepreke. U tu svrhu korištena je programska knjižnica Shapely [36] i metoda `Polygon::covered_by(Polygon p)` koja će pomoći pri klasifikaciji poligona. Ako je neki poligon sadržan unutar ili pokriven većim poligonom, taj element prostora možemo smatrati preprekom.

```
"example": {
    "name": "example",
    "definition": {
        "positivemeshes": [{
            "polygon": [[0, 0], [5, 0], [5, 5], [0, 5], [0, 0]],
            "bottom": 0,
            "top": 5
        }],
        "negativemeshes": [{
            "polygon": [[3, 1], [4, 1], [4, 2], [3, 2], [3, 1]],
            "bottom": 0,
            "top": 3
        }]
    }
}
```

Kodni isječak 4.4. Primjer zapisa elemenata u modelu prostora [7]

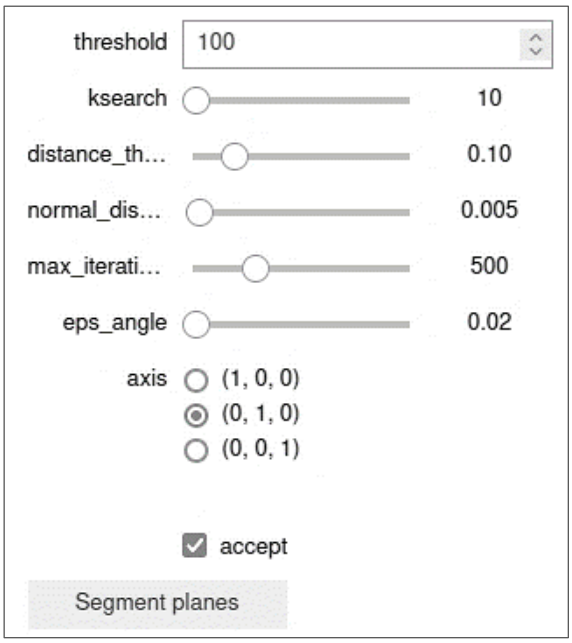
4.6. Jupyter Notebook interaktivno sučelje

Opisani algoritam može uspješno dobiti model prostora iz oblaka točaka na jednostavnim primjerima ortogonalnih prostora kao što su env1 i env2. Što se tiče TurtleBot House primjera koji je nešto složeniji, najbolji pristup bio bi podijeliti snimljeni oblak točaka na nekoliko manjih segmenata, npr. prostorija, na njima provesti predloženi algoritam i na kraju spojiti pojedinačna rješenja. Međutim, za prostore koji ne zadovoljavaju ograničenje ortogonalnosti nije zajamčena uspješna izrada modela. Primjer takvog prostora je env3, s jednom dijagonalno postavljenom preprekom.

Za takve je prostore moguće uz manje izmjene prilagoditi predloženi algoritam. Koraci poput uzorkovanja, izračuna sjecišta ploha, selekcije bridova, konstruiranja poligona i zapisa u JSON format u tom smislu su indiferentni na ograničenje ortogonalnosti. Segmentacija ploha može predstavljati problem. Ranije je opisano kako se u tom procesu izdvaja samo jedna ravnina okomita na z os i potom naizmjence ravnine okomite na x i na y os, sve dok u oblaku točaka ne

ponestane točaka za segmentaciju. Dijagonalno postavljene prepreke i granice prostora neće biti moguće segmentirati na taj način. Međutim, u potpoglavlju 4.2. spomenuto je i nekoliko parametara RANSAC segmentacije kojima korisnik može upravljati.

Kako bi se omogućila veća kontrola nad parametrima segmentacije te izrada modela za prostore koji ne zadovoljavaju ograničenje ortogonalnosti, razvijeno je interaktivno grafičko sučelje u Jupyter Notebook okruženju sa Slike 4.6.

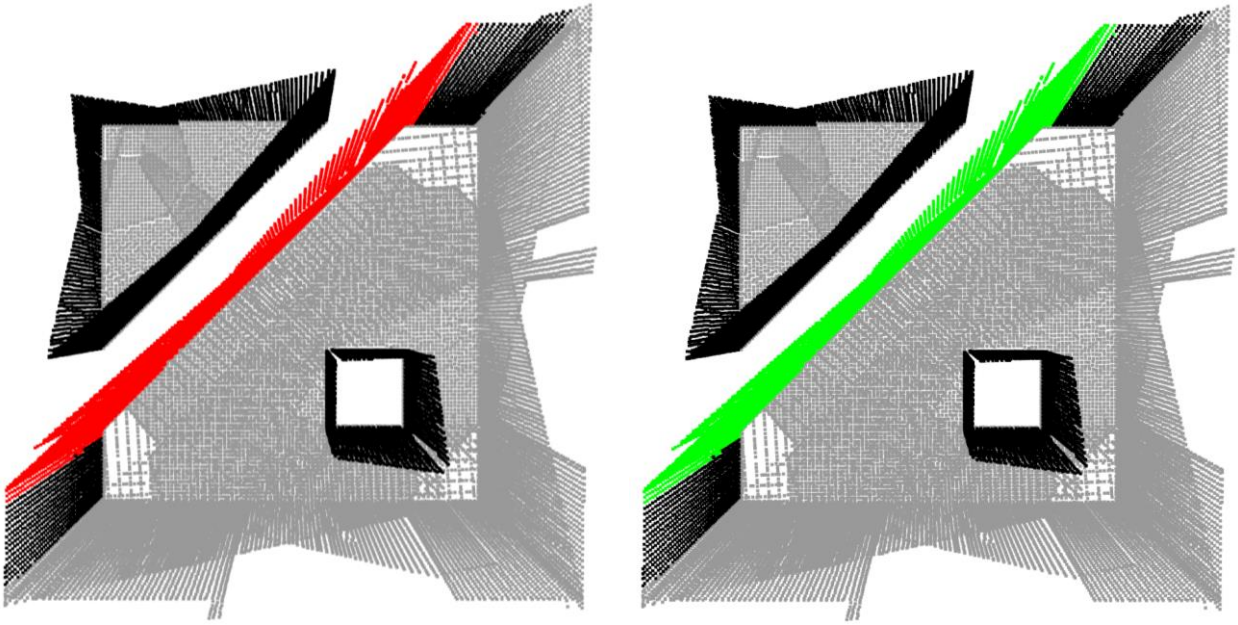


Slika 4.6. Jupyter Notebook interaktivno sučelje za upravljanje parametrima segmentacije

Osim upravljanja parametrima, od kojih je za spomenuti problem najznačajniji dozvoljeni kut odstupanja od normale `eps_angle`, korištenje Jupyter Notebook okruženja omogućuje i vizualizaciju procesa segmentacije. Vizualizacija segmentacije u oblaku točaka vidljiva je na Slici 4.7.

Svaki puta kada korisnik odabere željene parametre otvara se prozor u kojem je prikazan podskup točaka obuhvaćenih tako postavljenom segmentacijom. Sivom bojom su označene one točke za koje je već određeno kojoj ravnini pripadaju, crnom bojom podskup još uvijek nesegmentiranih točaka, a crvenom bojom su označene one točke koje bi bile uključene u trenutni korak segmentacije.

Ako je korisnik zadovoljan tako definirano ravninom, označit će polje za potvrdu `accept`, te će se točke obojiti zeleno što je znak da se može započeti s određivanjem sljedeće ravnine.



Slika 4.7. Vizualizacija RANSAC segmentacije

4.7. Validacija dobivenih modela prostora

Uspješnost predstavljenog algoritma za izradu modela prostora iz oblaka točaka treba utvrditi pouzdanim statističkim metodama koje će usporediti dobivene rezultate s polazišnim modelom prostora (eng. ground truth). Polazišni model prostora je onaj koji je učitan u Gazebo simulacijsko okruženje prije snimanja virtualnog prostora TurtleBot3 robotom. Literatura upućuje na dva koeficijenta pri odabiru adekvatne metrike za provjeru sličnosti polazišnog modela i dobivenog rješenja: Jaccardov indeks (eng. Intersection over Union, IoU) i Sørensen–Dice indeks (eng. Dice Similarity Coefficient, DSC).

Prva metrika, IoU, često se koristi i u drugim granama računarstva, kao primjerice u validaciji modela strojnog učenja. Prvi puta ju je definirao Grove Karl Gilbert 1884. godine u svojim istraživanjima na području meteorologije [37], a dobila je ime po botaničaru Paulu Jaccardu koji ju je također izrazio 1912. godine [38]. U kontekstu mjerenja uspješnosti modela segmentacije, izražava se kao omjer presjeka dobivenog rezultata s polazišnim podacima i njihove unije:

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|}. \tag{4.4}$$

Diceov koeficijent sličnosti neovisno su formulirali botaničari Lee Raymond Dice 1945. godine [39] i Thorvald Sørensen 1948. godine [40]. Izražava se kao omjer dvostrukog presjeka dobivenog rezultata s polazišnim podacima i njihovim zbrojem:

$$DSC(A,B) = \frac{2|A \cap B|}{|A| + |B|}. \tag{4.5}$$

Ove su dvije metrike proporcionalne jedna drugoj i u nekim drugim formulacijama (npr. kod korištenja matrica zabuna za problem statističke klasifikacije) donekle su slične. Razlikuju se po tome što IoU metrika nastoji kazniti pojedinačne slučajeve loše klasifikacije više nego Diceov koeficijent čak i kada se obje slažu da je taj konkretan primjer loše klasificiran. Može se reći da je DSC mjera prosječnog slučaja, dok je IoU mjera najgoreg slučaja.

Za primjere korištene u ovom radu izračunati su IoU i DSC koeficijenti uz pomoć programske knjižnice Shapely [36] te su prikazani u Tablici 4.2.

Tablica 4.2. Validacija dobivenih modela prostora

	IoU	**DSC**
env1	0,98733	0,99362
env2	0,98238	0,99111
env3	0,98121	0,99052
TurtleBot House	0,92262	0,95975

Grafička usporedba polazišnih modela i dobivenih rezultata nalazi se u Prilogu A.

5. PROBLEM RAZMJEŠTAJA OSJETILA

U prethodnom poglavlju prikazana je metoda dobivanja modela prostora pogodnih za rješavanje problema razmještaja osjetila iz ulaznih oblaka točaka.

Od izrađenih modela prostora, TurtleBot House je posebno interesantan. Taj se virtualni prostor proteže po površini od 86,68 m^2 i sadrži šest prostorija, hodnik i sedam prepreka (od kojih je algoritam za izradu modela prostora uspješno detektirao šest). Zbog svoje složenosti omogućava testiranje modela razmještaja osjetila [7] na primjeru s kakvim bismo se mogli susresti u svakodnevnom životu. Također, nudi mogućnost za prikupljanje puno veće količine korisnih informacija o optimizacijskim metodama od hipotetskih neorganskih prostora kao što su env1, env2 i env3, ali uz očekivano puno veće korištenje računalnih resursa i veće vrijeme izvršavanja.

S ciljem ispitivanja problema pokrivenosti prostora na primjeru dobivenog modela prostora TurtleBot House proveden je eksperiment. Prostor je podijeljen na konačan broj voksela veličine 10 cm × 10 cm × 10 cm. Generirano je 300 različitih ispitnih slučajeva koristeći do 50 osjetila koja su razmještena u prostoru i šest optimizacijskih algoritama. U eksperimentu se provjeravalo kako odabir optimizacijskog algoritma utječe na postignutu relativnu pokrivenost prostora i na prosječno vrijeme izvršavanja. Osim odabira algoritma, druga nezavisna varijabla u istraživanju bio je broj osjetila, uz početnu pretpostavku da pokrivenost i vrijeme izvršavanja rastu proporcionalno s porastom broja osjetila.

U izračunu pokrivenosti prostora korištena su BLE osjetila koja periodički odašilju radio-frekvencijski signal u svim smjerovima. Ta vrsta osjetila definira se pomoću dva parametra: jačine signala na udaljenosti od jednog metra (eng. Received Signal Strength Indicator, RSSI) i razine intenziteta šuma, odnosno vrijednosti ispod koje se signal ne može raspoznati u odnosu na ostala elektromagnetska zračenja u prostoru. U ovom slučaju korištena su osjetila sa značajkama $rssi_{1m} = -75\ dB$ i $noise = -85\ dB$, s maksimalnom vidljivošću od otprilike 3,16 m.

Svaki ispitni slučaj ponovljen je 30 puta radi uklanjanja mogućih varijabilnosti nastalih zbog stohastičke prirode optimizacijskih algoritama. Zbog velikog broja ispitnih slučajeva i ponavljanja bilo je potrebno osigurati dostatne računalne resurse što je omogućilo superračunalo Bura Sveučilišta u Rijeci [41].

5.1. Optimizacijski algoritmi

Za pronalazak optimalnog razmještaja osjetila koji maksimizira pokrivenost prostora korišteno je šest metaheurističkih stohastičkih algoritama. Metaheuristički algoritmi su poopćene tehnike rješavanja problema koje funkcioniraju neovisno o konkretnim detaljima zadatka i zato se mogu primijeniti na širok raspon problema, a mnogi od njih formulirani su tijekom druge polovice 20. stoljeća. Jedna od poznatih skupina metaheurističkih metoda su metode koje se temelje na stanju populacije, koje su inspirirane inteligencijom roja, evolucijskom teorijom, genetikom i drugim pojavama iz prirode. U ovom eksperimentu korišteni su sljedeći algoritmi (detaljnije predstavljeni u literaturi [42]):

- Algoritam uspona uz brijeg (eng. Hill Climbing Algorithm, HCA) je algoritam lokalne pretrage koji iterativno čini malene preinake prvotno nasumično odabranom rješenju dok god uspijeva postići bolji rezultat iz jedne iteracije u drugu. Najveća mana HCA je mogućnost odabira lokalnog maksimuma kod konkavnih prostora.
- Simulirano kaljenje (eng. Simulated Annealing, SA) u svakoj iteraciji provjerava susjede trenutnog rješenja i prihvaća rješenja s boljim rezultatom, ali za razliku od HCA ima mogućnost doći do globalnog maksimuma jer s određenom vjerojatnošću (koja se s vremenom smanjuje) može prihvatiti i neko rješenje koje je gore od trenutnog, a koje će u kasnijim iteracijama možda dovesti do optimalnog.
- Nelder-Mead metoda (eng. Nelder-Mead Method, NMM) koristi simpleks, poligon s $n + 1$ vrhova u n-dimenzionalnom prostoru, prilikom pronalaska optimalnog rješenja, gdje n odgovara broju nezavisnih varijabli. Simpleks se povećava, smanjuje i mijenja oblik prema definiranom skupu pravila i u svakoj iteraciji konvergira ka rješenju.
- Algoritam umjetnog roja pčela (eng. Artificial Bee Colony, ABC) koristi znanje populacije i istovremeno prati veći broj rješenja. Temelji se na navikama pomoću kojih roj pčela dolazi do izvora hrane.
- Algoritam optimizacije roja čestica (eng. Particle Swarm Optimization, PSO) prati populaciju mogućih čestica (rješenja) i pomiče te čestice u prostoru pretrage prema matematičkoj formuli koja uzima u obzir položaj i brzinu čestice. Kretanje svake čestice ovisi o obližnjem lokalnom maksimumu, ali i o drugim poznatim lokalnim maksimumima u prostoru pretrage.

- Algoritam vatrometa (eng. FireWorks Algorithm, FWA) je najnoviji algoritam inteligencije roja, razvijen 2010. godine [43]. Implementacija se odvija analogno lančanim „eksplozijama“ iskra u vatrometu: u svakoj iteraciji pretražuju se svi susjedi određenih čestica dok se ne pronađe odgovarajuća optimalna točka.

5.2. Statistička analiza pokrivenosti

U Prilogu B, u Tablici B.1., dane su srednje vrijednosti relativne pokrivenosti dobivene optimizacijom koristeći svih šest algoritama za ispitni prostor TurtleBot House uz varijabilni broj osjetila. U Tablici 5.1. dana je deskriptivna statistika za svaki od ispitanih algoritama.

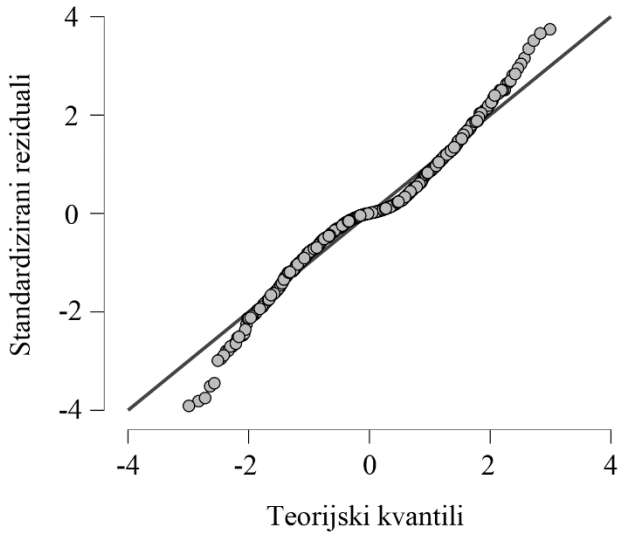
Tablica 5.1. Deskriptivna statistika za relativnu pokrivenost

	ABC	**PSO**	**HCA**	**FWA**	**SA**	**NMM**
Srednja vrijednost	0,5433	0,5172	0,4501	0,4239	0,4157	0,4021
Standardna devijacija	0,2244	0,2119	0,1796	0,1733	0,1799	0,1788
Najmanja vrijednost	0,0386	0,0387	0,0388	0,0379	0,0308	0,0269
25%	0,3791	0,3620	0,3199	0,2907	0,2755	0,2599
50%	0,5992	0,5706	0,4884	0,4561	0,4487	0,4341
75%	0,7365	0,7013	0,6031	0,5743	0,5720	0,5611
Najveća vrijednost	0,8189	0,7766	0,6791	0,6561	0,6592	0,6489

Kada je cilj statističke analize testirati postoji li statistički značajna razlika između aritmetičkih sredina više populacija, koristi se analiza varijance (eng. ANalysis Of VAriance, ANOVA). U analizi varijance značajnost utjecaja analiziranih faktora na promatranu slučajnu varijablu provodi se empirijskim F-testom. Uvjet je za ovo testiranje da uzorci potječu iz populacija s jednakim varijancama (uvjet homoskedastičnosti) i da su reziduali normalno distribuirani. Nultom se hipotezom pretpostavlja da su prosjeci svih populacija jednaki, a alternativnom hipotezom da nisu svi jednaki [44].

U ovom je slučaju korištena analiza varijance s dva promjenjiva faktora (eng. two-way ANOVA) jer se ispitivalo djelovanje dvije nezavisne varijable (odabir optimizacijskog algoritma i broj osjetila) na vrijednost jedne zavisne varijable (postignute relativne pokrivenosti prostora), a ponovljena mjerenja međusobno su nezavisna. U model dvofaktorske ANOVA-e može se uključiti i efekt interakcije dvaju faktora.

Normalna distribucija reziduala provjerava se QQ dijagramom. Slika 5.1. može biti dokaz da su reziduali otprilike normalno distribuirani jer točke koje predstavljaju reziduale otprilike prate normalne teorijske kvantile (pravac $y = x$). Dvofaktorska ANOVA smatra se dovoljno robusnom metodom i može proizvesti pouzdane rezultate ako postoje manja odstupanja po ovom pitanju. Homoskedastičnost podataka provjerava se Leveneovim testom koji je u ovom slučaju opovrgnuo nultu hipotezu ($F_{299,8700} = 13{,}568;\ p < {,}001$), ali taj je uvjet moguće zanemariti ako je broj mjerenja u svim skupinama jednak i dovoljno velik [45].



Slika 5.1. QQ dijagram za pokrivenost prostora

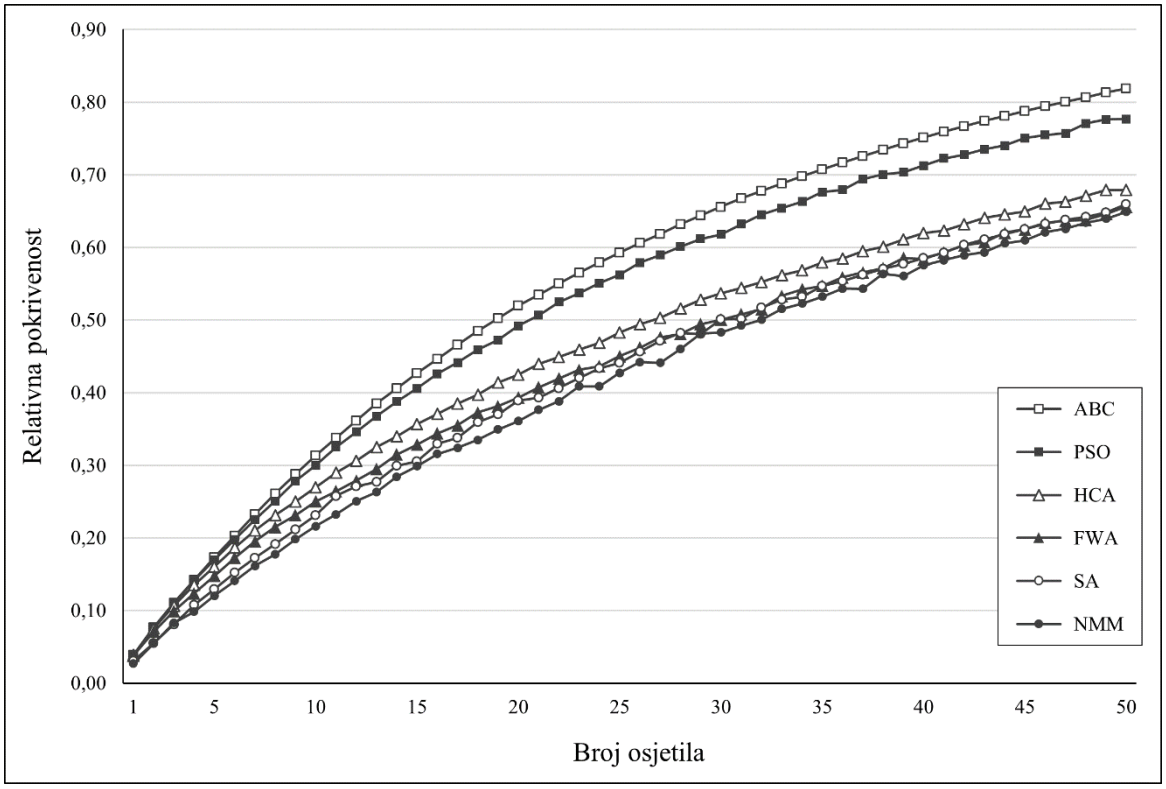
Rezultati ANOVA-e iz Tablice 5.2. utvrdili su da postoji statistički značajna razlika među srednjim vrijednostima za pokrivenost prostora u ovisnosti o odabiru optimizacijskog algoritma ($F_{5,8700} = 27\ 901{,}422;\ p < {,}001$), o broju osjetila ($F_{49,8700} = 36\ 194{,}408;\ p < {,}001$) i o interakciji ta dva faktora ($F_{245,8700} = 82{,}680;\ p < {,}001$). Mjera veličine učinka η^2 najveća je za faktor broja osjetila, a najmanja za interakciju dvaju promatranih faktora.

Tablica 5.2. Rezultati ANOVA testa za pokrivenost

	Sum of Squares	**df**	**Mean Square**	**F**	**p**	**η²**
algorithm	25,366	5	5,073	27 901,422	< ,001***	0,072
n_sensors	322,478	49	6,581	36 194,408	< ,001***	0,913
algorithm * n_sensors	3,683	245	0,015	82,680	< ,001***	0,010
Residuals	1,582	8700	0,182			

* p < ,05, ** p < ,01, *** p < ,001

Grafički prikaz rezultata za pokrivenost prostora vidljiv je na Slici 5.2.



Slika 5.2. Relativna pokrivenost

Kako bi se utvrdilo među kojim algoritmima postoji značajna razlika napravljena je *post hoc* analiza te su utvrđene statistički značajne razlike među svim parovima algoritama.

Post hoc analiza utjecaja broja osjetila na pokrivenost utvrdila je da postoji signifikantna razlika među prosječnim vrijednostima pokrivenosti prostora koje se postižu sa svakim povećanjem broja osjetila, ali se taj efekt smanjuje kada se koristi više od 40 osjetila.

Skraćena verzija *post hoc* analize utjecaja broja osjetila na pokrivenost prostora za svaki algoritam dana je u Prilogu B, u Tablici B.2. koja sadrži p vrijednosti za usporedbu parova uzastopnih brojeva osjetila. Utvrđeno je da su statistički značajne razlike za svaki algoritam u početku uočljive za parove uzastopnih brojeva osjetila, a nakon 10 do 15 osjetila taj se efekt smanjuje i potrebno je dodavati sve više osjetila da bi se postigle statistički značajne razlike. To je u skladu s grafičkim prikazom rezultata jer za svaki algoritam pokrivenost u početku raste brže, a kasnije sporije. Cjelovita *post hoc* analiza utvrdila je da 50 osjetila nije dovoljno za postizanje zasićenja.

Na temelju *post hoc* analize interakcije faktora algoritma i broja osjetila iz Priloga B, Tablice B.3. (skraćena verzija) može se zaključiti:

- Algoritam ABC i algoritam HCA postižu slične rezultate s do 5 osjetila, a kasnije dolazi do statistički signifikantne razlike u korist algoritma ABC. Isto vrijedi i za algoritme FWA i HCA (u korist algoritma HCA) te za algoritme HCA i PSO (u korist algoritma PSO).
- Algoritam ABC i algoritam PSO postižu slične rezultate s do 10 osjetila, a kasnije dolazi do statistički signifikantne razlike u korist algoritma ABC.
- Postoji statistički signifikantna razlika između algoritma FWA i algoritma SA u korist algoritma FWA, ali ne kada se koristi više od 20 osjetila.
- Postoji statistički signifikantna razlika između algoritma FWA i algoritma NMM u korist algoritma FWA, ali ne kada se koristi više od 40 osjetila.
- Postoji statistički signifikantna razlika između algoritma NMM i algoritma SA u korist algoritma SA, ali ne kada se koristi do 5 osjetila ili više od 40 osjetila.

U provedene *post hoc* analize uključena je Bonferroni korekcija.

5.3. Statistička analiza vremena izvršavanja

U Prilogu B, u Tablici B.4. dane su srednje vrijednosti vremena izvršavanja dobivenih optimizacijom uz korištenje svih šest algoritama za ispitni prostor TurtleBot House uz varijabilni broj osjetila. Vrijednosti su izražene u sekundama i zaokružene na najbliži cijeli broj. U Tablici 5.3. dana je deskriptivna statistika za vrijeme izvršavanja za svaki od ispitanih algoritama.

Tablica 5.3. Deskriptivna statistika za vrijeme (dd:hh:mm:ss format)

	ABC	**PSO**	**HCA**	**FWA**	**SA**	**NMM**
Srednja vrijednost	05:18:21:53	05:17:27:24	05:11:23:08	05:09:24:50	05:09:51:15	05:14:27:48
Standardna devijacija	05:02:38:02	05:01:38:10	04:20:24:23	04:19:15:14	04:19:27:58	04:23:21:36
Najmanja vrijednost	00:00:10:28	00:00:09:47	00:00:10:21	00:00:09:00	00:00:09:07	00:00:09:34
25%	01:02:45:42	01:02:48:56	01:01:30:09	01:00:43:42	01:00:36:41	01:02:02:04
50%	04:10:19:56	04:09:42:54	04:05:01:22	04:03:06:57	04:03:17:15	04:07:09:03
75%	09:19:46:51	09:16:34:09	09:07:04:54	09:03:13:11	09:05:01:30	09:10:52:26
Najveća vrijednost	16:17:50:53	16:10:44:09	15:20:31:57	15:14:40:08	15:19:26:21	16:05:41:27

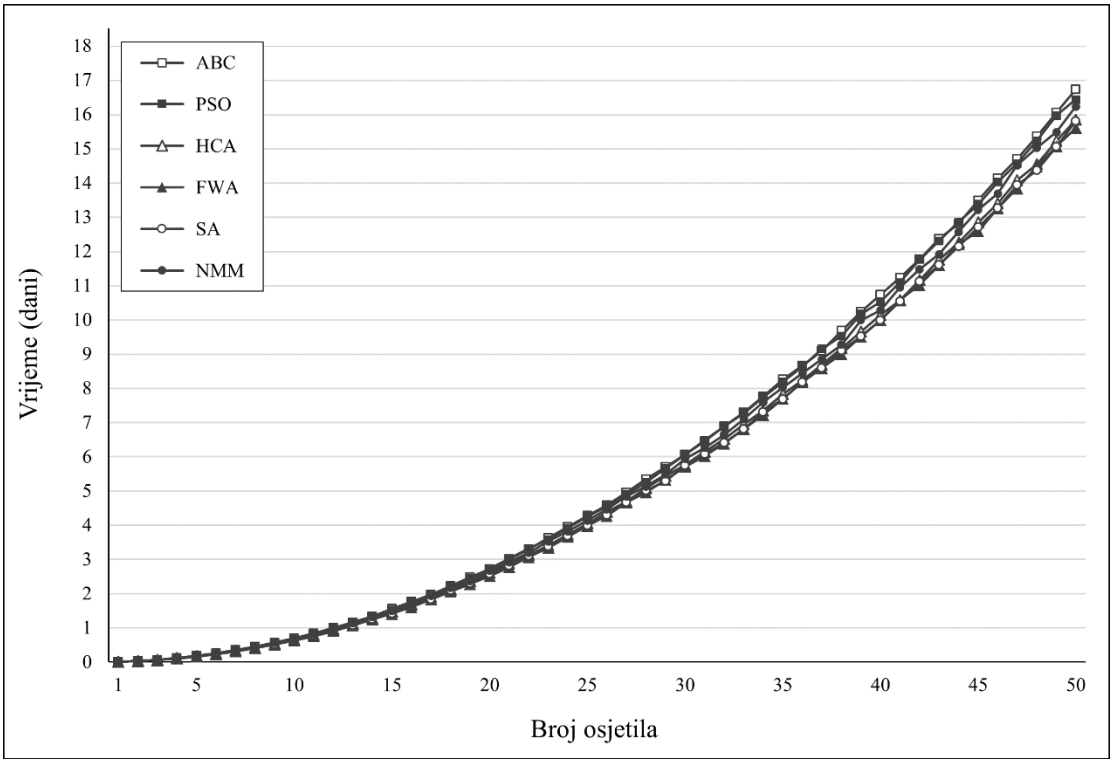
Rezultati ANOVA-e iz Tablice 5.4. utvrdili su da ne postoji statistički značajna razlika među srednjim vrijednostima za vrijeme izvršavanja u ovisnosti o odabiru optimizacijskog algoritma ($F_{5,294} = 0{,}053;\ p = 0{,}998$), ali da postoji statistički značajna razlika među srednjim vrijednostima u ovisnosti o broju osjetila ($F_{49,250} = 3\ 262{,}025;\ p <$,001).

Tablica 5.4. Rezultati ANOVA testa za vrijeme

	Sum of Squares	**df**	**Mean Square**	**F**	**p**
Algorithm	4,84E+10	5	9,68E+9	0,053	0,998
n_sensors	5,35E+13	49	1,09E+12	3 262,025	< ,001***

* p < ,05, ** p < ,01, *** p < ,001

Grafički prikaz rezultata za vrijeme izvršavanja vidljiv je na Slici 5.3.



Slika 5.3. Vrijeme izvršavanja

Post hoc analiza s Bonferroni korekcijom utvrdila je da korištenje do 10 osjetila ne utječe značajno na vrijeme izvršavanja, a nakon 40 osjetila vrijeme obrade statistički značajno raste za svako sljedeće dodano osjetilo. To je u skladu s grafičkim prikazom rezultata jer za svaki algoritam vrijeme izvršavanja u početku raste sporije, a kasnije brže. Skraćena verzija *post hoc* analize dana je u Prilogu B, u Tablici B.5.

6. ZAKLJUČAK

U sklopu ovog rada razvijen je algoritam za izradu modela prostora iz oblaka točaka. Problem modeliranja zatvorenih prostora iz trodimenzionalnih snimaka značajan je i pronalazi svoju primjenu u raznim područjima kao primjerice u arhitekturi, digitalnom očuvanju kulturne baštine, video igricama zasnovanima na virtualnoj stvarnosti te u projektiranju mreža osjetila. U literaturi [9, 10, 11] su predstavljena rješenja koja uspješno prepoznaju i modeliraju glavne arhitektonske strukture u interijerima, a doprinos ovog rada je mogućnost istovremenog modeliranja namještaja. Algoritam za izradu modela prostora temelji se na RANSAC segmentaciji ploha. Inicijalno je razvijan za modeliranje prostora s ortogonalnim tlocrtom i razmještajem prepreka, ali uz nižu razinu automatiziranosti i uz pomoć grafičkog korisničkog sučelja Jupyter Notebook uspješno modelira i prostore koji odstupaju od ograničenja ortogonalnosti.

U radu je korišten virtualni model mobilnog robota koji prolazi po virtualnim prostorima i koristeći dubinsku kameru gradi potpuni oblak točaka. U tu je svrhu bilo potrebno implementirati proces spajanja oblaka točaka u ROS2 okruženju za Python programski jezik. Na snimkama dobivenima u simuliranom okruženju učinkovitost algoritma za izradu modela prostora iz oblaka točaka kreće se oko 98% u najgorem i oko 99% u prosječnom slučaju za jednostavne prostore, te oko 92,26% u najgorem i oko 95,98% u prosječnom slučaju za složenije prostore. U slučaju da se podaci prikupljaju u stvarnom okruženju, bit će potrebno provjeriti uspješnost algoritma za tako dobivene rezultate pri čemu se očekuju nešto lošiji rezultati zbog veće količine šuma.

Algoritam za izradu modela prostora iz oblaka točaka razvijen je radi podrobnije analize modela razmještaja osjetila za pokrivanje zatvorenog prostora [7] na složenijim prostorima. Problem razmještaja osjetila svodi se na pronalaženje optimalnog broja i razmještaja osjetila potrebnih za postizanje potpune pokrivenosti prostora, a temelji se na istraženom problemu umjetničke galerije. Značajno je promatrati uspješnost modela razmještaja osjetila na dobivenom modelu prostora TurtleBot House jer taj prostor po svom obliku, veličini i sadržaju ima odlike prostora s kojima se susrećemo u svakodnevnom životu. Uvid u uspješnost modela na takvom primjeru može biti koristan pri analizi razmještaja osjetila u stvarnim prostorima za život i rad.

Proveden je eksperiment sa šest metaheurističkih optimizacijskih algoritama i do 50 osjetila s ciljem pronalaska razmještaja osjetila koji bi maksimizirao pokrivenost u TurtleBot House prostoru. Proračuni su provedeni na superračunalu Bura Sveučilišta u Rijeci [41].

ANOVA analizom utvrđeno je da algoritam ABC postiže najbolje rezultate (pokrivenost prostora do 81,89%), ali treba naglasiti da je to ujedno i najsporiji od ispitanih algoritama. Dobre rezultate postiže i PSO algoritam, a ostali algoritmi postižu značajno lošije rezultate kada se koristi veći broj osjetila. Korištenjem 50 osjetila i dalje nije dostignuto zasićenje, te bi se s većim brojem osjetila mogli postići statistički značajno bolji rezultati. Korisnicima bi, ovisno o njihovim potrebama, dugotrajno izvršavanje algoritama (otprilike 15 dana i 14 sati za najbrži optimizacijski algoritam, FWA) moglo biti presudno. U slučajevima kada je vrijeme izračuna važnije od postizanja maksimalne pokrivenosti, zadovoljavajući se rezultati mogu ostvariti s 45 osjetila. U istraživanjima kojima bi cilj bio dostići zasićenje pokrivenosti usprkos velikom vremenu izvršavanja, bilo bi bolje koristiti veću stopu povećanja broja osjetila (ispitati 55, 60, 65, 70… osjetila). ANOVA analizom utvrđeno je da ne postoje statistički značajne razlike u vremenu izvršavanja ovisno o odabiru algoritma, ali potrebno je imati na umu da se za veće brojeve osjetila algoritmi međusobno razlikuju do gotovo 100 tisuća sekundi što iznosi otprilike 27 sati. Ovi rezultati daju predodžbu o tome koliku bi uspješnost model razmještaja osjetila mogao ostvariti ako bi bio korišten za projektiranje sustava za nadzor stanova ili ureda.

LITERATURA

[1] Leksikografski zavod Miroslav Krleža, “Hrvatska enciklopedija: mjerno osjetilo,” [Mrežno]. Dostupno na: http://www.enciklopedija.hr/Natuknica.aspx?ID=45689. [Pokušaj pristupa: 6. ožujka 2022.].

[2] K. Grgić, D. Žagar i V. Križanović, “Medical applications of wireless sensor networks - current status and future directions,” *Medicinski glasnik,* svez. 9, br. 1, pp. 23-31, 2011.

[3] E.-K. Lee, M. Gerla, G. Pau, U. Lee i J.-H. Lim, “Internet of Vehicles: From intelligent grid to autonomous cars and vehicular fogs,” *International Journal of Distributed Sensor Networks*, svez. 12, br. 9, 2016.

[4] V. Chvátal, “A Combinatorial Theorem in Plane Geometry,” *Journal of Combinatorial Theory, Series B,* svez. 18, br. 1, pp. 39-41, 1975.

[5] J. O'Rourke, Art Gallery Theorems and Algorithms, New York: Oxford University Press, Inc., 1987.

[6] M. Hefeeda i H. Ahmadi, “Energy Effcient Protocol for Deterministic and Probabilistic Coverage in Sensor Networks,” *IEEE Transactions on Parallel and Distributed Systems,* svez. 21, br. 5, pp. 579-593, 2009.

[7] D. Sušanj, *Model razmještaja osjetila za pokrivanje zatvorenoga prostora,* Rijeka: Sveučilište u Rijeci, Tehnički fakultet, 2021.

[8] M. Banov, “Point Cloud Registration,” Laboratorij za umjetnu percepciju i autonomne sustave, Sveučilište u Rijeci, Tehnički fakultet, [Mrežno]. Dostupno na: https://gitlab.com/apaslab/point_cloud_registration. [Pokušaj pristupa: 13. rujna 2022.].

[9] A. Budroni i J. Boehm, “Automated 3D Reconstruction of Interiors from Point Clouds,” *International Journal of Architectural Computing*, svez. 8, br. 1, pp. 55-73, 2010.

[10] V. Sanchez i A. Zakhor, “Planar 3D Modeling of Building Interiors from Point Cloud Data,” u *2012 19th IEEE International Conference on Image Processing*, 2012.

[11] S. Murali, P. Speciale, M. R. Oswald i M. Pollefeys, “Indoor Scan2BIM: Building

Information Models of House Interiors," u *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2017.

[12] M. Banov, "Point Cloud Modelling," Laboratorij za umjetnu percepciju i autonomne sustave, Sveučilište u Rijeci, Tehnički fakultet, [Mrežno]. Dostupno na: https://gitlab.com/apaslab/point_cloud_modelling. [Pokušaj pristupa: 13. rujna 2022.].

[13] M. Quigley, K. Conley, B. P. Gerkey i suradnici, "ROS: an open-source Robot Operating System," u *ICRA Workshop on Open Source Software,* Kobe, 2009.

[14] N. Koenig i A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," u *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* Sendai, 2004.

[15] R. Amsters i P. Slaets, "Turtlebot 3 as a Robotics Education Platform," u *Robotics in Education,* Cham, 2020.

[16] Intel, "Intel RealSense SDK," [Mrežno]. Dostupno na: https://github.com/IntelRealSense/librealsense. [Pokušaj pristupa: 14. ožujka 2022.].

[17] Intel, "ROS Wrapper for Intel RealSense Devices," [Mrežno]. Dostupno na: https://github.com/IntelRealSense/realsense-ros. [Pokušaj pristupa: 14. ožujka 2022.].

[18] PAL Robotics, "Intel RealSense Gazebo ROS plugin," [Mrežno]. Dostupno na: https://github.com/pal-robotics/realsense_gazebo_plugin. [Pokušaj pristupa: 14. ožujka 2022.].

[19] E. B. Goldstein, "Perceiving Depth and Size," u *Sensation and Perception,* Belmont, Thomson Wadsworth, 2010, pp. 228-257.

[20] P. Bourke, "Creating depth maps using PovRay," [Mrežno]. Dostupno na: http://paulbourke.net/reconstruction/depthmap2/. [Pokušaj pristupa: 2. rujna 2022.].

[21] P. Besl i H. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* svez. 14, br. 2, pp. 239-256, 1992.

[22] M. Pound, "Joining Point Cloud Scans (ICP) - Computerphile," YouTube, [Mrežno]. Dostupno na: https://www.youtube.com/watch?v=4uWSo8v3iQA. [Pokušaj pristupa: 24.

ožujka 2022.].

[23] J. Kammerl i W. Woodall, “PCL ROS,” [Mrežno]. Dostupno na: https://github.com/ros-perception/perception_pcl. [Pokušaj pristupa: 14. veljače 2022.].

[24] A. H. Cordero i C. Lalancette, “ROS2 - Geometry2 (TF2 Sensor Messages),” [Mrežno]. Dostupno na: https://github.com/ros2/geometry2. [Pokušaj pristupa: 14. veljače 2022.].

[25] C. Gohlke, “Transformations,” [Mrežno]. Dostupno na: https://github.com/cgohlke/transformations. [Pokušaj pristupa: 3. rujna 2022.].

[26] M. Jafari, “Quaternions Algebra and Its Applications: An Overview,” *International Journal of Theoretical and Applied Mathematics*, svez. 2, br. 2, pp. 79-85, 2016.

[27] A. Hornung, K. M. Wurm, M. Bennewitz i suradnici, “OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees,” *Autonomous Robots,* svez. 34, br. 3, pp. 189-206, 2013.

[28] D. Sušanj, D. Pinčić i K. Lenac, “Effective Area Coverage of 2D and 3D Environments With Directional and Isotropic Sensors,” *IEEE Access,* svez. 8, pp. 185595-185608, 2020.

[29] R. B. Rusu i S. Cousinsy, “3D is here: Point Cloud Library (PCL),” u *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, 2011.

[30] A. Straw, “Python PCL,” [Mrežno]. Dostupno na: https://github.com/strawlab/python-pcl. [Pokušaj pristupa: 2. kolovoza 2022.].

[31] T. Kluyver, B. Ragan-Kelley i suradnici, “Jupyter Notebooks - a publishing format for reproducible computational workflows,” u *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, Göttingen, 2016.

[32] M. A. Fischler i R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM,* svez. 24, br. 6, pp. 381-395, 1981.

[33] J. Abramson i suradnici, “Systems of Linear Equations: Three variables,” u *Precalculus*, Houston, OpenStax Rice University, 2017, pp. 774-784.

[34] G. Strang, Linear algebra and its applications, Belmont: Thomson, Brooks/Cole, 2006.

[35] C. R. Harris i suradnici, "Array programming with NumPy," *Nature,* svez. 585, br. 7825, pp. 357-362, 2020.

[36] S. Gillies, "Shapely," [Mrežno]. Dostupno na: https://github.com/shapely/shapely. [Pokušaj pristupa: 12. kolovoza 2022.].

[37] A. H. Murphy, "The Finley Affair: A Signal Event in the History of Forecast Verification," *Weather and Forecasting,* svez. 11, br. 1, pp. 3-20, 1996.

[38] P. Jaccard, "The Distribution of the Flora in the Alpine Zone," *New Phytologist,* svez. 11, br. 2, pp. 37-50, 1912.

[39] L. R. Dice, "Measures of the Amount of Ecologic Association Between Species," *Ecology,* svez. 26, br. 3, pp. 297-302, 1945.

[40] T. J. Sørensen, A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species Content and Its Application to Analyses of the Vegetation on Danish Commons, Copenhagen: I kommission hos E. Munksgaard, 1948.

[41] Centar za napredno računanje i modeliranje, "Superračunalo Bura," [Mrežno]. Dostupno na: https://cnrm.uniri.hr/hr/bura/. [Pokušaj pristupa: 29. kolovoza 2022.].

[42] E.-G. Talbi, Metaheuristics: From Design to Implementation, Hoboken: John Wiley & Sons, 2009.

[43] Y. Tan i Y. Zhu, "Fireworks Algorithm for Optimization," u *Advances in Swarm Intelligence*, Berlin, 2010.

[44] J. Arnerić i K. Protrka, "Modeli analize varijance (ANOVA)," *Matematičko-fizički list,* svez. LXX, br. 1, pp. 25-32, 2019./2020.

[45] D. W. Zimmerman, "A note on preliminary tests of equality of variances," *The British journal of mathematical and statistical psychology,* svez. 57, br. 1, pp. 173-181, 2004.

[46] H. Karl i A. Willig, "Introduction," u *Protocols and Architectures for Wireless Sensor Networks*, Chichester, Wiley-Interscience, 2005, pp. 1-13.

POPIS KRATICA

ABC, eng. Artificial Bee Colony – umjetni roja pčela

ANOVA, eng. ANalysis Of VAriance – analiza varijance

BLE, eng. Bluetooth Low Energy – Bluetooth niska energija

DSC, eng. Dice Similarity Coefficient – Diceov koeficijent sličnosti

FWA eng. FireWorks Algorithm – algoritam vatrometa

HCA, eng. Hill Climbing Algorithm – algoritam uspona uz brijeg

ICP, eng. Iterative Closest Point – iterativna najbliža točka

IoT, eng. Internet of Things – Internet stvari

IoU, eng. Intersection over Union – presjek nad unijom

JSON, eng. JavaScript Object Notation – JavaScript zapis objekata

LDS, eng. Laser Distance Sensor – lasersko daljinsko osjetilo

NMM, eng. Nelder-Mead Method – Nelder-Mead metoda

PCL, eng. Point Cloud Library – knjižnica oblaka točaka

PSO eng. Particle Swarm Optimization – optimizacija roja čestica

RANSAC, eng. RANdom SAmple Consensus –nasumični konsenzusni uzorak

ROS, eng. Robot Operating System – robotski operacijski sustav

RSSI, eng. Received Signal Strength Indicator – indikator jačine primljenog signala

SA, eng. Simulated Annealing – simulirano kaljenje

SDK, eng. Software Development Kit – osnovne knjižnice i alati za rad

SLAM, eng. Simultaneous Localization And Mapping – istovremena lokalizacija i mapiranje

USB, eng. Universal Serial Bus – univerzalna serijska sabirnica

POPIS SLIKA

POPIS KODNIH ISJEČAKA

POPIS TABLICA

SAŽETAK

Ovaj rad razmatra kako kreirati model prostora pogodan za korištenje u istraživanjima problema razmještaja osjetila. Razvijeno je programsko rješenje koje u trodimenzionalnom oblaku točaka prepoznaje granice zatvorenog prostora, namještaj i druge prepreke te zapisuje njihov tlocrt kao skup poligona. U radu su kratko objašnjeni teorijski pojmovi koji su bili polazište za istraživanje – problem umjetničke galerije, stereoskopske kamere, registracija oblaka točaka, RANSAC segmentacija i drugi. Opisan je proces snimanja prostora pomoću TurtleBot3 mobilnog robota i dubinske kamere u simulacijskom okruženju, a potom i detalji izrade modela prostora. Učinkovitost algoritma za izradu apstraktnog modela prostora iz oblaka točaka provjerena je usporedbom dobivenih rezultata s polazišnim modelima prostora, pri čemu je izračunata sličnost od 98 – 99% za jednostavne prostore te 92 – 96% za složene prostore.

Dobiveni primjer složenijeg prostora korišten je za analizu problema razmještaja osjetila kojim se pokušava pronaći optimalan broj i razmještaj osjetila potrebnih za postizanje potpune pokrivenosti zatvorenog prostora. Taj je primjer značajan jer može dati uvid u uspješnost modela razmještaja osjetila na stvarnim prostorima s kojima se susrećemo u svakodnevnom životu. Proveden je eksperiment koristeći šest metaheurističkih optimizacijskih algoritama i do 50 osjetila. Na temelju ANOVA statističke analize podataka doneseni su zaključci o utjecaju odabira optimizacijskog algoritma i broja osjetila nad postignutom relativnom pokrivenosti prostora i vremenom izvršavanja. Uočeno je da najbolje rezultate postiže algoritam umjetnog roja pčela, a dobre rezultate daje i algoritam optimizacije roja čestica.

Ključne riječi: oblak točaka, model prostora, razmještaj osjetila, TurtleBot3, RANSAC segmentacija, metaheuristika.

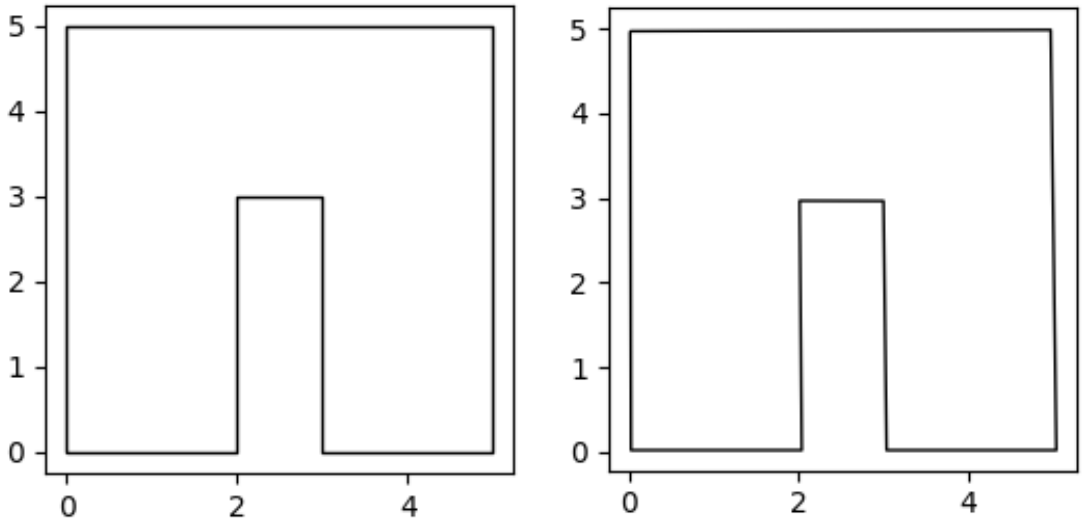
ABSTRACT

This paper considers how to create an environment model suitable for use in research on the problem of spatial sensor distribution. A software solution was developed that recognizes the boundaries of interiors, furniture, and other obstacles in a three-dimensional point cloud and records their floor plan as a set of polygons. The paper briefly explains the theoretical concepts that were the starting point for the research – the art gallery problem, stereoscopic cameras, point cloud registration, RANSAC segmentation, and others. The process of recording the space using the TurtleBot3 mobile robot and depth cameras in a simulation environment is described, followed by the details of the creation of an environment model. The effectiveness of the algorithm for creating an abstract environment model from a given point cloud was verified by comparing the obtained results with the ground truth, where a similarity of 98 – 99% was calculated for simple environments, and 92 – 96% for complex environments.

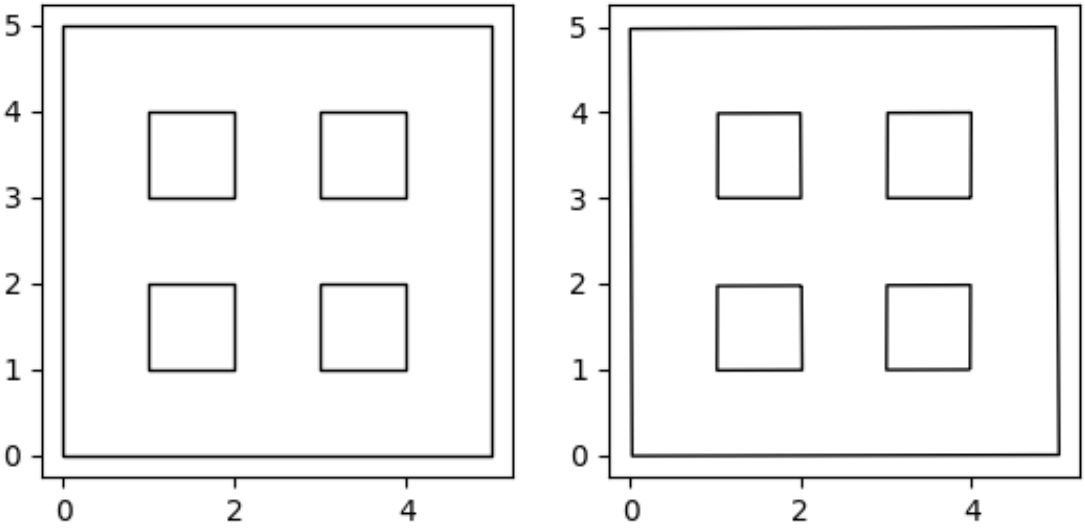
The obtained example of a more complex environment was used for the analysis of the sensor placement problem, which tries to find the optimal number and placement of sensors needed to achieve complete coverage of the closed environment. This example is significant because it can provide an insight into the effectiveness of the sensor placement model in real environments that we encounter in everyday life. An experiment was conducted using six metaheuristic optimization algorithms and up to 50 sensors. Based on the ANOVA statistical analysis of the data, conclusions were drawn about the influence of the selection of the optimization algorithm and the number of sensors on the achieved relative coverage of space and execution time. It was observed that the artificial bee colony algorithm achieves the best results, and the particle swarm optimization algorithm also gives good results.

Keywords: point cloud, environment model, spatial sensor distribution, TurtleBot3, RANSAC segmentation, metaheuristics.

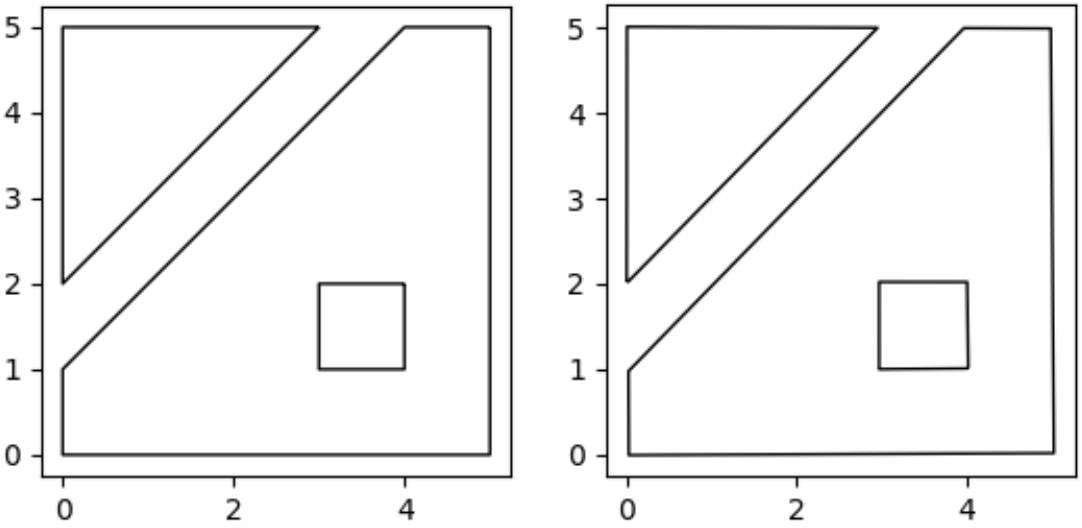
PRILOG A



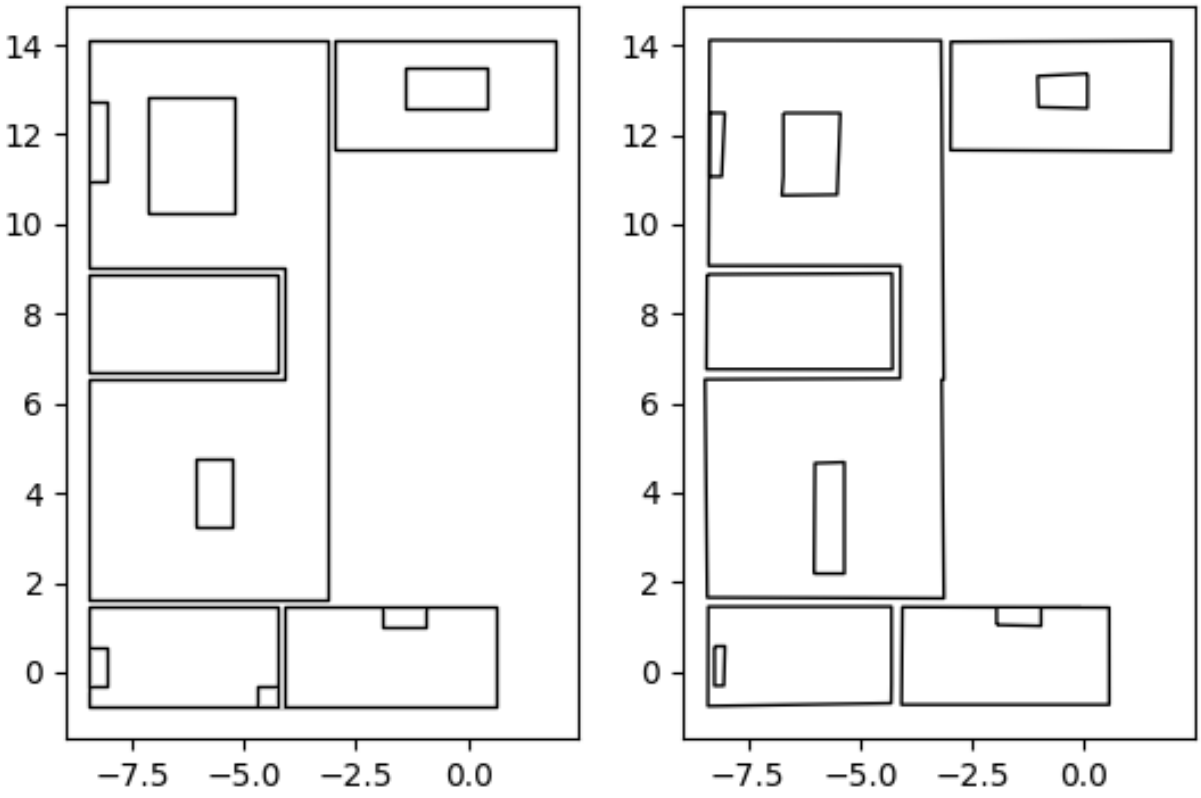
Slika A.1. Usporedba polazišnog modela i rezultata izrade modela (env1)



Slika A.2. Usporedba polazišnog modela i rezultata izrade modela (env2)



Slika A.3. Usporedba polazišnog modela i rezultata izrade modela (env3)



Slika A.4. Usporedba polazišnog modela i rezultata izrade modela (TurtleBot House)

PRILOG B

Tablica B.1. Srednje vrijednosti relativne pokrivenosti prostora

Broj osjetila	ABC	PSO	HCA	FWA	SA	NMM
1	0,0386	0,0387	0,0388	0,0379	0,0308	0,0269
2	0,0766	0,0767	0,0750	0,0707	0,0548	0,0546
3	0,1103	0,1097	0,1064	0,0990	0,0806	0,0829
4	0,1421	0,1413	0,1350	0,1235	0,1079	0,0985
5	0,1732	0,1697	0,1603	0,1476	0,1294	0,1204
6	0,2031	0,1977	0,1866	0,1725	0,1520	0,1410
7	0,2322	0,2253	0,2099	0,1953	0,1725	0,1615
8	0,2609	0,2505	0,2313	0,2148	0,1916	0,1774
9	0,2876	0,2782	0,2503	0,2314	0,2115	0,1983
10	0,3136	0,2996	0,2694	0,2502	0,2311	0,2160
11	0,3378	0,3252	0,2899	0,2642	0,2577	0,2324
12	0,3616	0,3460	0,3060	0,2789	0,2710	0,2506
13	0,3849	0,3673	0,3246	0,2946	0,2769	0,2630
14	0,4059	0,3879	0,3397	0,3149	0,2993	0,2843
15	0,4269	0,4055	0,3564	0,3285	0,3055	0,2988
16	0,4460	0,4260	0,3705	0,3438	0,3293	0,3157
17	0,4661	0,4415	0,3852	0,3551	0,3378	0,3239
18	0,4846	0,4591	0,3969	0,3726	0,3594	0,3349
19	0,5021	0,4725	0,4140	0,3812	0,3698	0,3492
20	0,5195	0,4921	0,4248	0,3933	0,3891	0,3608
21	0,5349	0,5064	0,4394	0,4073	0,3928	0,3764
22	0,5501	0,5246	0,4488	0,4195	0,4062	0,3880
23	0,5649	0,5368	0,4589	0,4315	0,4201	0,4085
24	0,5793	0,5505	0,4688	0,4364	0,4332	0,4085
25	0,5924	0,5622	0,4824	0,4501	0,4409	0,4271
26	0,6059	0,5790	0,4943	0,4620	0,4564	0,4421
27	0,6186	0,5894	0,5028	0,4761	0,4715	0,4412
28	0,6318	0,6011	0,5158	0,4809	0,4817	0,4600
29	0,6439	0,6116	0,5281	0,4944	0,4805	0,4813
30	0,6558	0,6178	0,5365	0,5003	0,5009	0,4829
31	0,6676	0,6320	0,5442	0,5077	0,5017	0,4922
32	0,6777	0,6452	0,5519	0,5150	0,5174	0,5004
33	0,6881	0,6539	0,5615	0,5331	0,5276	0,5154
34	0,6979	0,6631	0,5682	0,5421	0,5322	0,5225
35	0,7074	0,6761	0,5792	0,5469	0,5469	0,5323
36	0,7166	0,6793	0,5845	0,5589	0,5534	0,5433
37	0,7258	0,6939	0,5950	0,5657	0,5620	0,5429
38	0,7342	0,7005	0,6005	0,5711	0,5703	0,5634
39	0,7432	0,7035	0,6110	0,5855	0,5773	0,5604
40	0,7512	0,7127	0,6194	0,5836	0,5856	0,5755
41	0,7590	0,7225	0,6229	0,5930	0,5924	0,5825
42	0,7668	0,7277	0,6317	0,6027	0,6038	0,5891
43	0,7741	0,7348	0,6406	0,6071	0,6108	0,5933
44	0,7808	0,7404	0,6454	0,6203	0,6182	0,6057
45	0,7876	0,7503	0,6492	0,6249	0,6250	0,6097
46	0,7941	0,7547	0,6599	0,6340	0,6327	0,6208
47	0,8007	0,7571	0,6628	0,6366	0,6376	0,6255
48	0,8068	0,7707	0,6712	0,6376	0,6423	0,6338
49	0,8131	0,7758	0,6791	0,6462	0,6479	0,6395
50	0,8189	0,7766	0,6788	0,6561	0,6592	0,6489

Tablica B.2. Post hoc analiza relativne pokrivenosti po broju osjetila za svaki algoritam

		ABC	PSO	HCA	FWA	SA	NMM
1	2	< ,001***	< ,001***	< ,001***	< ,001***	< ,001***	< ,001***
2	3	< ,001***	< ,001***	< ,001***	< ,001***	< ,001***	< ,001***
3	4	< ,001***	< ,001***	< ,001***	< ,001***	< ,001***	0,355
4	5	< ,001***	< ,001***	< ,001***	< ,001***	< ,001***	< ,001***
5	6	< ,001***	< ,001***	< ,001***	< ,001***	< ,001***	< ,001***
6	7	< ,001***	< ,001***	< ,001***	< ,001***	< ,001***	< ,001***
7	8	< ,001***	< ,001***	< ,001***	< ,001***	0,002**	0,207
8	9	< ,001***	< ,001***	0,002**	0,093	< ,001***	< ,001***
9	10	< ,001***	< ,001***	0,002**	0,003**	< ,001***	0,017*
10	11	< ,001***	< ,001***	< ,001***	1,000	< ,001***	0,118
11	12	< ,001***	< ,001***	0,178	1,000	1,000	0,008**
12	13	< ,001***	< ,001***	0,004**	0,289	1,000	1,000
13	14	< ,001***	< ,001***	0,629	< ,001***	< ,001***	< ,001***
14	15	< ,001***	0,019*	0,073	1,000	1,000	1,000
15	16	0,002**	< ,001***	1,000	0,503	< ,001***	0,059
16	17	< ,001***	0,388	1,000	1,000	1,000	1,000
17	18	0,005**	0,021*	1,000	0,025*	< ,001***	1,000
18	19	0,022*	1,000	0,038*	1,000	1,000	1,000
19	20	0,025*	< ,001***	1,000	1,000	0,001**	1,000
20	21	0,460	1,000	1,000	1,000	1,000	0,325
21	22	0,574	0,007**	1,000	1,000	1,000	1,000
22	23	1,000	1,000	1,000	1,000	1,000	< ,001***
23	24	1,000	1,000	1,000	1,000	1,000	1,000
24	25	1,000	1,000	1,000	1,000	1,000	0,004**
25	26	1,000	0,071	1,000	1,000	0,387	0,750
26	27	1,000	1,000	1,000	1,000	0,705	1,000
27	28	1,000	1,000	1,000	1,000	1,000	0,003**
28	29	1,000	1,000	1,000	1,000	1,000	< ,001***
29	30	1,000	1,000	1,000	1,000	< ,001***	1,000
30	31	1,000	1,000	1,000	1,000	1,000	1,000
31	32	1,000	1,000	1,000	1,000	0,308	1,000
32	33	1,000	1,000	1,000	0,010**	1,000	0,776
33	34	1,000	1,000	1,000	1,000	1,000	1,000
34	35	1,000	1,000	1,000	1,000	1,000	1,000
35	36	1,000	1,000	1,000	1,000	1,000	1,000
36	37	1,000	1,000	1,000	1,000	1,000	1,000
37	38	1,000	1,000	1,000	1,000	1,000	< ,001***
38	39	1,000	1,000	1,000	1,000	1,000	1,000
39	40	1,000	1,000	1,000	1,000	1,000	0,635
40	41	1,000	1,000	1,000	1,000	1,000	1,000
41	42	1,000	1,000	1,000	1,000	1,000	1,000
42	43	1,000	1,000	1,000	1,000	1,000	1,000
43	44	1,000	1,000	1,000	1,000	1,000	1,000
44	45	1,000	1,000	1,000	1,000	1,000	1,000
45	46	1,000	1,000	1,000	1,000	1,000	1,000
46	47	1,000	1,000	1,000	1,000	1,000	1,000
47	48	1,000	1,000	1,000	1,000	1,000	1,000
48	49	1,000	1,000	< ,001***	1,000	1,000	1,000
49	50	1,000	1,000	1,000	1,000	1,000	1,000

* p < ,05, ** p < ,01, *** p < ,001

Note. P-value adjusted for comparing a family of 50

Tablica B.3. Efekt interakcije odabira algoritma i broja osjetila na relativnu pokrivenost

		Mean Difference	SE	t	Cohen's d	p_{bonf}
5 ABC	5 FWA	0,026	0,004	7,175	1,853	< ,001***
	5 HCA	0,013	0,004	3,620	0,935	0,195
	5 NMM	0,053	0,004	14,802	3,822	< ,001***
	5 PSO	0,003	0,004	0,979	0,253	1,000
	5 SA	0,044	0,004	12,278	3,170	< ,001***
10 ABC	10 FWA	0,063	0,004	17,769	4,588	< ,001***
	10 HCA	0,044	0,004	12,371	3,194	< ,001***
	10 NMM	0,098	0,004	27,346	7,061	< ,001***
	10 PSO	0,014	0,004	3,919	1,012	0,060
	10 SA	0,082	0,004	23,118	5,969	< ,001***
20 ABC	20 FWA	0,126	0,004	35,384	9,136	< ,001***
	20 HCA	0,095	0,004	26,556	6,857	< ,001***
	20 NMM	0,159	0,004	44,488	11,487	< ,001***
	20 PSO	0,027	0,004	7,696	1,987	< ,001***
	20 SA	0,130	0,004	36,543	9,435	< ,001***
30 ABC	30 FWA	0,155	0,004	43,564	11,248	< ,001***
	30 HCA	0,119	0,004	33,415	8,628	< ,001***
	30 NMM	0,173	0,004	48,435	12,506	< ,001***
	30 PSO	0,038	0,004	10,641	2,747	< ,001***
	30 SA	0,155	0,004	43,394	11,204	< ,001***
40 ABC	40 FWA	0,168	0,004	46,972	12,128	< ,001***
	40 HCA	0,132	0,004	36,952	9,541	< ,001***
	40 NMM	0,176	0,004	49,255	12,718	< ,001***
	40 PSO	0,039	0,004	10,792	2,786	< ,001***
	40 SA	0,166	0,004	46,423	11,986	< ,001***
50 ABC	50 FWA	0,163	0,004	45,562	11,764	< ,001***
	50 HCA	0,140	0,004	39,197	10,121	< ,001***
	50 NMM	0,171	0,004	47,818	12,347	< ,001***
	50 PSO	0,042	0,004	11,807	3,048	< ,001***
	50 SA	0,160	0,004	44,831	11,575	< ,001***
5 FWA	5 HCA	-0,013	0,004	-3,555	-0,918	0,249
	5 NMM	0,027	0,004	7,627	1,969	< ,001***
	5 PSO	-0,022	0,004	-6,196	-1,600	< ,001***
	5 SA	0,018	0,004	5,103	1,318	< ,001***
10 FWA	10 HCA	-0,019	0,004	-5,398	-1,394	< ,001***
	10 NMM	0,034	0,004	9,577	2,473	< ,001***
	10 PSO	-0,049	0,004	-13,850	-3,576	< ,001***
	10 SA	0,019	0,004	5,349	1,381	< ,001***
20 FWA	20 HCA	-0,031	0,004	-8,828	-2,279	< ,001***
	20 NMM	0,032	0,004	9,104	2,351	< ,001***
	20 PSO	-0,099	0,004	-27,688	-7,149	< ,001***
	20 SA	0,004	0,004	1,159	0,299	1,000
30 FWA	30 HCA	-0,036	0,004	-10,149	-2,620	< ,001***
	30 NMM	0,017	0,004	4,872	1,258	< ,001***
	30 PSO	-0,117	0,004	-32,923	-8,501	< ,001***
	30 SA	-0,604	0,004	-0,169	-0,044	1,000
40 FWA	40 HCA	-0,036	0,004	-10,020	-2,587	< ,001***
	40 NMM	0,008	0,004	2,283	0,590	1,000
	40 PSO	-0,129	0,004	-36,180	-9,342	< ,001***

	40 SA	-0,002	0,004	-0,549	-0,142	1,000
50 FWA	50 HCA	-0,023	0,004	-6,365	-1,643	< ,001***
	50 NMM	0,008	0,004	2,257	0,583	1,000
	50 PSO	-0,120	0,004	-33,755	-8,716	< ,001***
	50 SA	-0,003	0,004	-0,730	-0,189	1,000
5 HCA	5 NMM	0,040	0,004	11,182	2,887	< ,001***
	5 PSO	-0,009	0,004	-2,641	-0,682	1,000
	5 SA	0,031	0,004	8,658	2,235	< ,001***
10 HCA	10 NMM	0,053	0,004	14,976	3,867	< ,001***
	10 PSO	-0,030	0,004	-8,451	-2,182	< ,001***
	10 SA	0,038	0,004	10,748	2,775	< ,001***
20 HCA	20 NMM	0,064	0,004	17,932	4,630	< ,001***
	20 PSO	-0,067	0,004	-18,860	-4,870	< ,001***
	20 SA	0,036	0,004	9,988	2,579	< ,001***
30 HCA	30 NMM	0,054	0,004	15,020	3,878	< ,001***
	30 PSO	-0,081	0,004	-22,774	-5,880	< ,001***
	30 SA	0,036	0,004	9,979	2,577	< ,001***
40 HCA	40 NMM	0,044	0,004	12,303	3,177	< ,001***
	40 PSO	-0,093	0,004	-26,160	-6,754	< ,001***
	40 SA	0,034	0,004	9,471	2,445	< ,001***
50 HCA	50 NMM	0,031	0,004	8,621	2,226	< ,001***
	50 PSO	-0,098	0,004	-27,390	-7,072	< ,001***
	50 SA	0,020	0,004	5,634	1,455	< ,001***
5 NMM	5 PSO	-0,049	0,004	-13,822	-3,569	< ,001***
	5 SA	-0,009	0,004	-2,524	-0,652	1,000
10 NMM	10 PSO	-0,084	0,004	-23,427	-6,049	< ,001***
	10 SA	-0,015	0,004	-4,228	-1,092	0,016*
20 NMM	20 PSO	-0,131	0,004	-36,792	-9,500	< ,001***
	20 SA	-0,028	0,004	-7,944	-2,051	< ,001***
30 NMM	30 PSO	-0,135	0,004	-37,795	-9,759	< ,001***
	30 SA	-0,018	0,004	-5,041	-1,302	< ,001***
40 NMM	40 PSO	-0,137	0,004	-38,463	-9,931	< ,001***
	40 SA	-0,010	0,004	-2,832	-0,731	1,000
50 NMM	50 PSO	-0,128	0,004	-36,012	-9,298	< ,001***
	50 SA	-0,011	0,004	-2,987	-0,771	1,000
5 PSO	5 SA	0,040	0,004	11,299	2,917	< ,001***
10 PSO	10 SA	0,069	0,004	19,199	4,957	< ,001***
20 PSO	20 SA	0,103	0,004	28,847	7,448	< ,001***
30 PSO	30 SA	0,117	0,004	32,754	8,457	< ,001***
40 PSO	40 SA	0,127	0,004	35,631	9,200	< ,001***
50 PSO	50 SA	0,118	0,004	33,025	8,527	< ,001***

* p < ,05, ** p < ,01, *** p < ,001

Note. P-value adjusted for comparing a family of 36

Tablica B.4. Srednje vrijednosti vremena izvršavanja

Broj osjetila	ABC	PSO	HCA	FWA	SA	NMM
1	628	587	621	540	547	574
2	2 491	2 416	2 382	2 217	2 175	2 447
3	5 487	5 390	5 299	5 002	4 763	5 362
4	9 871	9 712	9 341	8 868	8 786	9 864
5	15 366	14 946	14 421	13 693	13 451	14 555
6	21 897	21 592	20 927	19 861	20 126	21 673
7	29 748	29 475	28 428	26 989	27 192	27 532
8	38 310	37 740	36 643	35 408	35 534	37 233
9	48 594	48 324	46 218	44 016	44 736	48 650
10	60 111	59 374	56 818	54 074	55 322	57 333
11	72 089	72 119	68 230	65 109	67 272	69 889
12	86 030	85 290	80 912	78 250	79 350	81 785
13	99 779	100 285	95 441	92 612	91 684	97 704
14	115 204	114 668	110 771	107 153	107 710	112 724
15	133 838	131 645	127 016	122 091	121 760	128 421
16	152 504	150 441	144 810	138 509	142 089	145 976
17	169 523	170 166	163 851	158 049	157 757	167 163
18	191 882	190 765	182 256	177 167	180 625	187 427
19	213 756	211 037	203 089	197 330	199 170	203 772
20	234 695	235 311	225 275	216 562	222 322	230 038
21	259 049	261 238	248 800	239 428	242 955	252 391
22	284 850	286 711	270 496	265 509	267 614	275 719
23	313 798	309 744	293 900	288 527	291 678	304 750
24	341 214	338 279	321 136	316 195	317 759	330 029
25	369 504	367 604	348 718	344 151	343 904	357 014
26	396 088	393 545	378 645	369 484	370 967	385 671
27	428 608	423 360	405 274	403 034	403 306	417 910
28	461 858	453 290	438 357	428 482	432 679	443 134
29	492 257	489 311	469 824	460 763	457 088	473 684
30	523 449	523 362	497 602	493 010	496 126	513 231
31	557 094	559 759	531 141	520 552	524 718	540 874
32	595 050	595 590	563 057	552 201	553 189	574 107
33	629 962	627 786	600 714	588 611	588 003	613 608
34	670 308	668 449	635 191	623 922	631 407	655 847
35	713 550	707 695	676 548	664 948	664 123	693 593
36	748 699	746 052	711 533	707 542	706 164	728 667
37	784 946	790 252	750 536	742 380	742 134	765 019
38	836 874	823 269	792 769	778 180	786 624	801 151
39	884 622	879 187	834 067	822 223	822 887	863 531
40	927 568	910 060	876 371	863 151	862 928	888 699
41	970 124	958 131	912 950	914 893	911 283	946 002
42	1 017 662	1 016 624	964 726	952 306	961 463	991 626
43	1 068 796	1 063 748	1 016 594	1 001 803	1 003 109	1 030 082
44	1 106 464	1 111 215	1 059 021	1 055 848	1 049 595	1 086 047
45	1 166 101	1 156 245	1 110 816	1 088 335	1 098 873	1 142 399
46	1 221 521	1 212 223	1 157 824	1 145 914	1 146 862	1 183 361
47	1 271 226	1 259 854	1 216 386	1 196 358	1 204 364	1 254 535
48	1 327 995	1 317 098	1 257 471	1 251 932	1 241 449	1 299 176
49	1 387 961	1 380 177	1 316 268	1 302 507	1 302 153	1 338 545
50	1 446 653	1 421 049	1 369 917	1 348 808	1 365 981	1 402 887

Tablica B.5. Post hoc analiza za vrijeme izvršavanja

		Mean Difference	SE	t	Cohen's d	p_{bonf}
1	2	-1 771,817	10 559,269	-0,168	-0,097	1,000
2	3	-2 862,697	10 559,269	-0,271	-0,157	1,000
3	4	-4 189,545	10 559,269	-0,397	-0,229	1,000
4	5	-4 998,635	10 559,269	-0,473	-0,273	1,000
5	6	-6 607,275	10 559,269	-0,626	-0,361	1,000
6	7	-7 214,552	10 559,269	-0,683	-0,394	1,000
7	8	-8 583,953	10 559,269	-0,813	-0,469	1,000
8	9	-9 944,987	10 559,269	-0,942	-0,544	1,000
9	10	-10 415,804	10 559,269	-0,986	-0,570	1,000
10	11	-11 945,870	10 559,269	-1,131	-0,653	1,000
11	12	-12 818,110	10 559,269	-1,214	-0,701	1,000
12	13	-14 314,923	10 559,269	-1,356	-0,783	1,000
13	14	-15 120,649	10 559,269	-1,432	-0,827	1,000
14	15	-16 090,313	10 559,269	-1,524	-0,880	1,000
15	16	-18 259,411	10 559,269	-1,729	-0,998	1,000
16	17	-18 696,694	10 559,269	-1,771	-1,022	1,000
17	18	-20 602,240	10 559,269	-1,951	-1,126	1,000
18	19	-19 672,103	10 559,269	-1,863	-1,076	1,000
19	20	-22 674,818	10 559,269	-2,147	-1,240	1,000
20	21	-23 276,084	10 559,269	-2,204	-1,273	1,000
21	22	-24 506,358	10 559,269	-2,321	-1,340	1,000
22	23	-25 249,927	10 559,269	-2,391	-1,381	1,000
23	24	-27 035,967	10 559,269	-2,560	-1,478	1,000
24	25	-27 713,602	10 559,269	-2,625	-1,515	1,000
25	26	-27 250,815	10 559,269	-2,581	-1,490	1,000
26	27	-31 182,129	10 559,269	-2,953	-1,705	1,000
27	28	-29 384,642	10 559,269	-2,783	-1,607	1,000
28	29	-30 854,612	10 559,269	-2,922	-1,687	1,000
29	30	-33 975,435	10 559,269	-3,218	-1,858	1,000
30	31	-31 226,224	10 559,269	-2,957	-1,707	1,000
31	32	-33 176,122	10 559,269	-3,142	-1,814	1,000
32	33	-35 914,874	10 559,269	-3,401	-1,964	0,956
33	34	-39 406,577	10 559,269	-3,732	-2,155	0,288
34	35	-39 222,267	10 559,269	-3,714	-2,145	0,308
35	36	-38 033,215	10 559,269	-3,602	-2,080	0,467
36	37	-37 768,474	10 559,269	-3,577	-2,065	0,511
37	38	-40 600,162	10 559,269	-3,845	-2,220	0,187
38	39	-47 941,499	10 559,269	-4,540	-2,621	0,011*
39	40	-37 043,125	10 559,269	-3,508	-2,025	0,655
40	41	-47 434,703	10 559,269	-4,492	-2,594	0,013*
41	42	-48 503,656	10 559,269	-4,593	-2,652	0,008**
42	43	-46 620,967	10 559,269	-4,415	-2,549	0,018*
43	44	-47 343,042	10 559,269	-4,484	-2,589	0,014*
44	45	-49 096,478	10 559,269	-4,650	-2,684	0,007**
45	46	-50 822,952	10 559,269	-4,813	-2,779	0,003**
46	47	-55 836,151	10 559,269	-5,288	-3,053	< ,001***
47	48	-48 733,041	10 559,269	-4,615	-2,665	0,008**
48	49	-55 414,931	10 559,269	-5,248	-3,030	< ,001***
49	50	-54 613,939	10 559,269	-5,172	-2,986	< ,001***

* p < ,05, ** p < ,01, *** p < ,001

Note. P-value adjusted for comparing a family of 50