Autonomno istraživanje 3D prostora uz kompenzaciju greške nastale optimizacijom trajektorije

Pušnik, Filip

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 52

AUTONOMNO ISTRAŽIVANJE 3D PROSTORA UZ KOMPENZACIJU GREŠKE NASTALE OPTIMIZACIJOM TRAJEKTORIJE

Filip Pušnik

Zagreb, veljača 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 52

AUTONOMNO ISTRAŽIVANJE 3D PROSTORA UZ KOMPENZACIJU GREŠKE NASTALE OPTIMIZACIJOM TRAJEKTORIJE

Filip Pušnik

Zagreb, veljača 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 2. listopada 2023.

DIPLOMSKI ZADATAK br. 52

Pristupnik:	**Filip Pušnik (0036517904)**
Studij:	Informacijska i komunikacijska tehnologija
Profil:	Automatika i robotika
Mentor:	prof. dr. sc. Stjepan Bogdan

Zadatak: **Autonomno istraživanje 3D prostora uz kompenzaciju greške nastale optimizacijom trajektorije**

Opis zadatka:

Cilj rada je autonomno istražiti i planirati putanju robota na temelju podkarte prostora. Sigurno i učinkovito volumetrijsko istraživanje 3D prostora većih razmjera obavit će se pomoću bespilotne letjelice. Metoda autonomnog istraživanja prostora treba se temeljiti na detekciji fronte te na kombinaciji lokalnog (vremenski i prostorno) i globalnog kartiranja prostora. U obzir je potrebno uzeti i grešku nastalu uslijed kontinuirane optimizacije trajektorije. U sklopu rada potrebno je koristiti već razvijenu metodu za autonomno istraživanje 3D prostora te ju je potrebno prilagoditi novom simulacijskom okruženju i upravljačkom sustavu. Za simulacijsko ispitivanje metode koristit će se Gazebo simulator u robotskom operacijskom sustavu (ROS). Razvijenu metodu potrebno je ispitati u stvarnom okruženju.

Rok za predaju rada: 9. veljače 2024.

Zahvaljujem mentoru prof. dr. sc. Stjepanu Bogdanu na mentorstvu i savjetima tijekom provedenog školovanja te Ani Milas, mag. ing. na pomoći i savjetima potrebnim za izradu ovog rada.

Zahvaljujem obitelji, pogotovo roditeljima koji su mi sve ovo omogućili i bili podrška od početka do kraja.

Za kraj zahvaljujem prijateljima s kojima sam obilježio ovaj period života i stvorio brojne uspomene.

SADRŽAJ

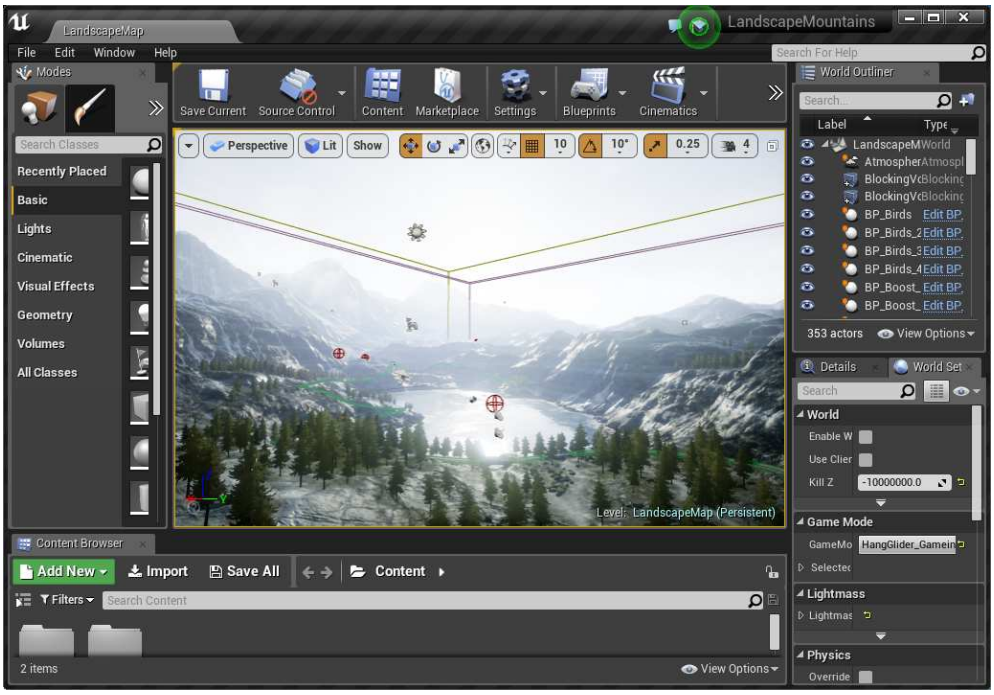
1. Uvod

Jedan od važnijih i popularnijih proizvoda današnje robotike predstavljaju bespilotne letjelice ili dronovi. Zbog širokog opsega mogućnosti koriste se u raznim industrijama poput vojne za izviđanje i nadzor terena te obavljanje operacija opasnih za čovjeka, filmske i zabavne za snimanje raznih sadržaja, zatim za nadzor prirodnih nepogoda, mjerenje terena i kartiranje i još mnogo drugih primjena. Na slici 1.1 može se vidjeti jedna od danas sve popularnijih primjena, a to je u području agronomije gdje se prate kvaliteta usjeva te različiti uvjeti koji prevladavaju. Istraživanje prostora predstavlja jednu od popularnijih mogućnosti za korištenje dronova, pogotovo u znanstvenoj zajednici, što dovodi do razvijanja raznih algoritama za autonomno istraživanje. Jedan od takvih predstavljen je u ovom radu gdje se objašnjava kako sam algoritam funkcionira, od kojih se dijelova sastoji te kako je implementiran, a za kraj ćemo pokušati prilagoditi sustav da funkcionira na različitim simulacijskim okruženjima ili na pravoj letjelici.

Slika 1.1: Primjena drona u agronomiji [13]

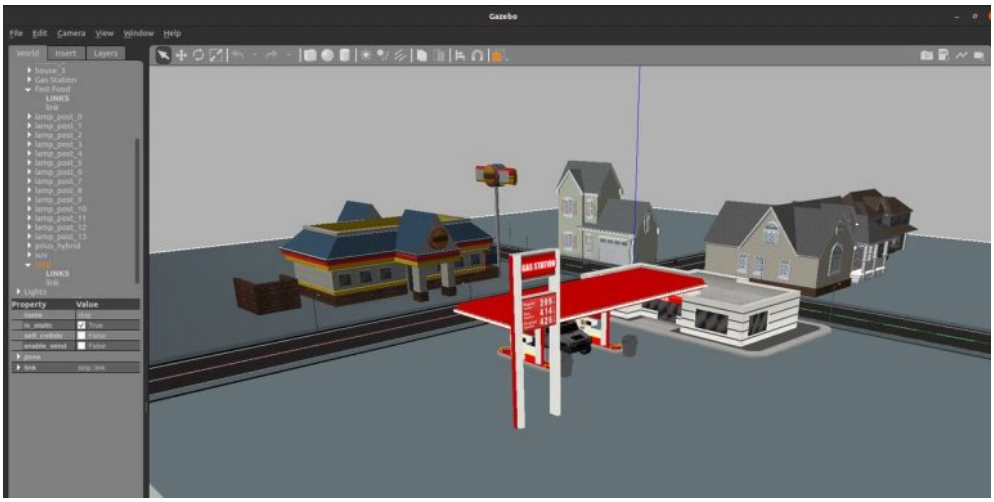
2. Programski alati

Korišteni sustav pretežito je baziran na C++ programskom jeziku te uz njega koristi i robotski operacijski sustav (*eng. Robot Operating System*), ROS [8], okruženje vrlo popularno u svijetu robotike zbog svojih mogućnosti te već postojećih biblioteka i alata. Kreatori GLocal sustava [1], koji se obrađuje u ovom radu, za demonstraciju koriste AirSim [14] simulacijsko okruženje koje je razvio Microsoft te je popularno za razvoj algoritama u područjima umjetne inteligencije, računalnog vida te autonomnih vozila. Samo simulacijsko okruženje bazirano je na Unreal Engineu, popularnom alatu za razvoj realne 3D grafike. Prikaz grafičkog sučelja simulatora i primjer okruženja može se vidjeti na slikama 2.1 i 2.2. Za potrebe ovog rada također će se koristiti vrlo popularno simulacijsko okruženje Gazebo simulator [10]. Primjer njegovog grafičkog sučelja može se vidjeti na slici 2.3. Iako je Gazebo simulator manje sofisticiran od Unreal Enginea koji je grafički puno superiorniji, jedna od njegovih prednosti je velika integriranost s ROS-om što omogućuje jednostavno kontroliranje i obradu informacija. Također, modeliranje objekta i izgradnja simulacijskog svijeta je dosta jednostavnije. S druge strane Unreal Engine osim uz vjerodostojan prikaz simulacijskog svijeta ima i mogućnost integracije s VR sustavima što može biti korisno u nekim slučajevima. Međutim, te stavke dolaze sa cijenom veće potrošnje računalnih resursa te je potrebno imati puno snažnije računalo nego u slučaju Gazebo simulatora.



Slika 2.1: Unreal grafičko sučelje [14]

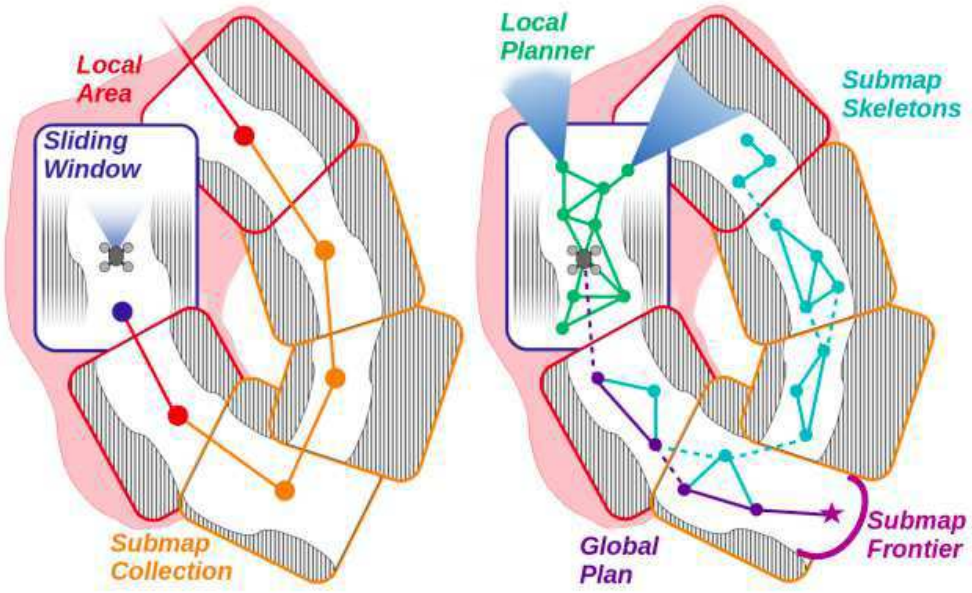
Slika 2.2: AirSim simulator [14]



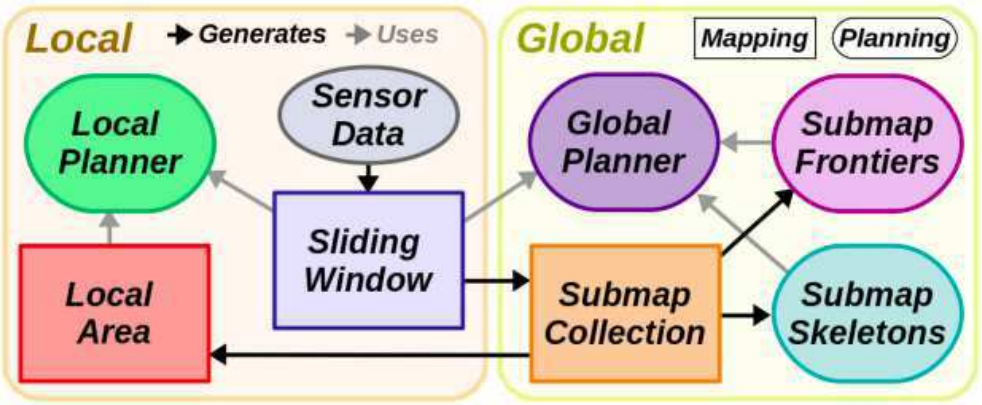
Slika 2.3: Gazebo simulator [15]

3. GLocal sustav autonomnog istraživanja prostora

GLocal sustav autonomnog istraživanja prostora razvijen je od strane istraživača Laboratorija autonomnih sustava s Tehničkog sveučilišta u Zürichu (ETH Zürich). Taj modularni sustav za globalno i lokalno istraživanje velikih okruženja, temeljen na podkartama prostora, kombinira različite slojeve kartiranja i planiranja kako bi se postigla robustnost na grešku odometrije koja se nakuplja pri prijeđenim velikim udaljenostima. To uključuje kombinaciju lokalnog (vremenski i prostorno) i globalnog kartiranja prostora, korištenje lokalnog planiranja za sigurno i brzo istraživanje i globalnog planiranja baziranog na detekciji fronte kako bi osigurali globalnu pokrivenost prostora. Na slici 3.1 može se vidjeti grafički prikaz pristupa kartiranja i planiranja. Najbitniji dijelovi kartiranja su lokalno područje (eng. *local area*), klizni prozor (eng. *sliding window*) i kolekcija podkarti (eng. *submap collection*) dok su u fazi planiranja to lokalni (eng. *local*) i globalni planer (eng. *global planner*), kostur podkarti (eng. *submap skeletons*) i fronte podkarti (eng. *submap frontiers*). Na slici 3.2 prikazan je shemtaski pregled cjelokupnog sustava. Prikazana je glavna podjela na lokalno i globalno područje te su kvadratima obilježene komponente kartiranja, a elipsama komponente sustava planiranja. Sustav se sastoji od lokalnog i globalnog planera, lokalnog područja, kliznog prozora za čije su generiranje potrebni podaci sa senzora (eng. *sensor data*), kolekcije podkarti (eng. *submap collection*) te kostura i fronti podkarata. Crne strelice označavaju da određena komponenta generira iduću dok sive obilježavaju uporabu određene komponente. Na primjer, podaci sa senzora se koriste za generiranje kliznog prozora dok on koristi lokalni i globalni planer u svojem dijelu. Navedene dijelove ćemo detaljnije razmotriti u sljedećem odjeljku.



Slika 3.1: Dijelovi kartiranja i planiranja sustava [1]

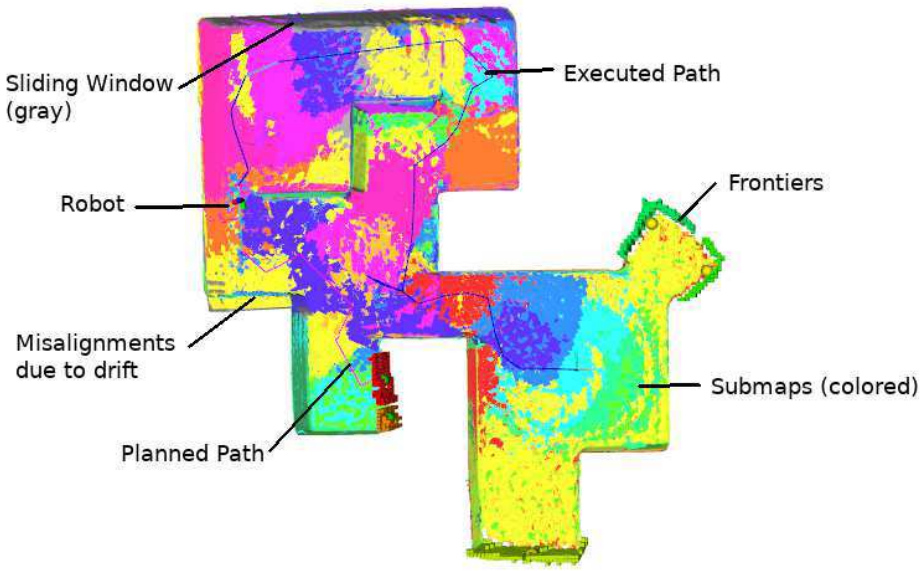


Slika 3.2: Shematski pregled sustava [1]

3.1. Dijelovi sustava

GLocal sustav se može podijeliti na šest glavnih komponenti. Kada se govori o kartiranju tu su ključni pojmovi klizni prozor i lokalna karta za definiranje lokalnog područja, ali isto tako imaju ulogu i u izgradnji globalne karte koja predstavlja treću komponetnu. U fazi planiranja ključne komponente sustava su lokalni i globalni planer te fronte podkarti. Vizualizirani prikaz tih komponenti može se vidjeti na slici 3.3 koja prikazuje jedan manji dio istraženog prostora u RViz grafičkom sučelju [9] koji koristimo za promatranje rada sustava. Prikazani su klizni prozor (eng. *sliding window*), podkarte (eng. *submaps*), planirani (eng. *planned*) i izvršeni put (eng. *executed path*), fronte (eng. *frontiers*) i neusklađenost zbog pomaka u odometriji (eng. *misalignments due to*

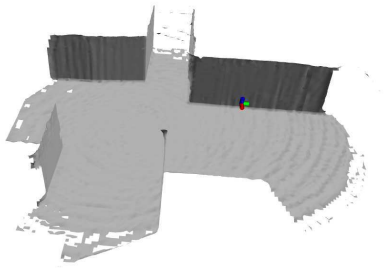
drift). U nastavku je dan opis navedenih komponenti.



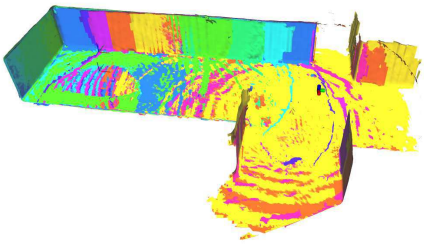
Slika 3.3: Vizualni prikaz GLocal sustava [1]

3.1.1. *Klizni prozor*

Kako se u istraživanju velikih okruženja s vremenom nakuplja greška u odometriji nije optimalno koristiti jednu globalnu kartu kao cjelinu već se ona dijeli na slojeve. Prvi sloj kartiranja čini tzv. karta kliznog prozora koja predstavlja privremenu lokalnu kartu prostora gdje se polazi od pretpostavke da je greška u odometriji dovoljno mala u kratkom vremenskom razdoblju što nam omogućuje kreiranje TSDF mreže. Za generiranje TSDF mreže koristi se *voxblox* sustav [4]. Funkcija skraćene udaljenosti s predznakom (eng. *truncated signed distance function*), TSDF, definira koliko je neka točka (unutar definirane skraćene udaljenosti) udaljena od površine promatranog objekta. Za točke koje se nalaze van objekta udaljenost je pozitivnog predznaka dok je za one unutar objekta negativnog predznaka. Glavna uloga TSDF-a je predstavljanje oblika i površina u prostoru na temelju podataka sa senzora poput RGB-D kamera ili LiDARA-a. Njihova izgradnja temelji se na emitiranju točki zraka senzora te usrednjavanjem novih projekcijskih mjerenja u postojeće ćelije. Nakon što relativna nesigurnost između trenutne poze i poze u kojoj su napravljena mjerenja pređe zadanu granicu mjerenja se uklanjaju te time završava ovaj korak. Na slici 3.4 prikazan je primjer kliznog prozora i usporedba s podkartama prostora sa slike 3.5. Vidi se kako klizni prozor predstavlja samo jedan manji dio okoline koja je trenutno u fokusu.



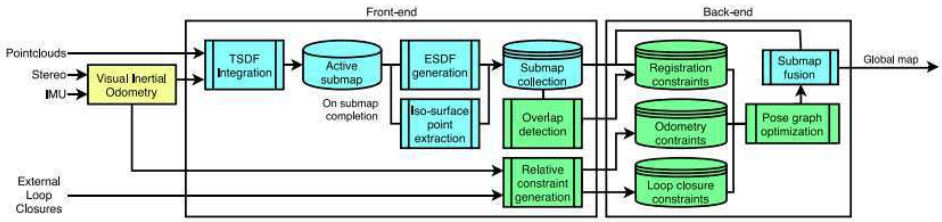
Slika 3.4: Klizni prozor



Slika 3.5: Skupina podkarti

3.1.2. Globalna karta

Kako bi se na kraju ipak dobila globalna karta, klizni prozor se periodično sprema kao podkarta prostora te sve podkarte zajedno predstavljaju globalnu kartu. Za globalnu optimizaciju, odnosno poravnanje podkarti prostora koristi se *voxgraph* algoritam [3] koji podkarte organizira u graf pozi zajedno s odometrijom kao apriori vrijednosti te poravnanjem površine kao ograničenjem. Pregled sustava prikazan je na slici 3.6.



Slika 3.6: Voxgraph pregled [3]

Glavna je podjela na *frontend* i *backend* sučelja. U *frontend* dijelu se koriste mjerenja sa senzora poput LiDAR senzora za dobivanje oblaka točaka (eng. *pointcloud*) te kamera ili inercijske mjerne jedinice (eng. *inertial measurement unit*), IMU, za

vizualnu i/ili inercijsku odometriju. Ti podaci se zatim koriste za kreiranje podkarti definiranom frekvencijom pomoću *voxblox* sustava [4] na principu integracije TSDF-a i generacije ESDF-a (eng. *Euclidian Signed Distance Function*). Također u ovom dijelu se određuju ograničenja koja se prosljeđuju *backend* sustavu za fuziju podkarti. Postoje tri vrste ograničenja, odometrijska, zatvaranje petlje (na temelju ponovno posjećenih lokacija) i registracijska. Najveću ulogu imaju registracijska ograničenja koja se odnose na geometrijsko poravnanje preklapajućih (eng. *overlap detection*) podkarti koristeći granične okvire poravnate s osima (eng. *Axis Aligned Bounding Boxes*), AABB [7].

Na slici 3.7 dan je prikaz okruženja nakon završenog procesa istraživanja na temelju izgrađenih podkarti sustava.

Slika 3.7: Globalna karta pomoću *voxgraph* sustava

3.1.3. Lokalno područje

Lokalno područje ili još zvana prostorno lokalna karta predstavlja zadnji sloj u postupku kartiranja. Ono daje ključne informacije lokalnom planeru kako bi se efikasno istražio prostor. Sastoji se od jedne TSDF karte koja se gradi na temelju podkarti koje se preklapaju s područjem kliznog prozora, odnosno proširuje se i smanjuje sukladno tome kako se postojeće podkarte nalaze u blizini ili napuštaju prostor kliznog prozora.

3.1.4. Lokalni planer

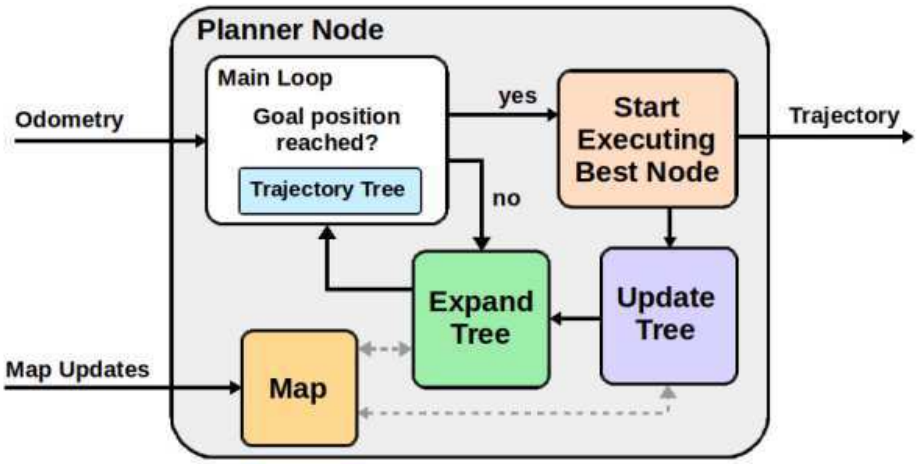
Kako bi se iskoristio puni potencijal i dobilo što efikasnije planiranje puta koriste se tzv. informativni putevi odnosno pokušava se riješiti problem informativnog planiranja puta (eng. *informative path planning*), IPP, u kojem je cilj dobiti put koji maksimizira dobivene informacije iz okoline. Danas su u tom području vrlo popularne metode bazirane na uzorkovanju. Tu je vrlo poznat algoritam brzorastućih slučajnih stabala (eng. Rapidly exploring random trees), RRT, od kojeg proizlaze razni noviji i efikasniji algoritmi. U GLocal sustavu koristi se planer temeljen na algoritmu predloženom u radu [2]. Taj algoritam je inspiriran RRT* metodom te rješava neke od problema iz originalnog RRT algoritma. Jedan od njih je to što se u RRT algoritmu veliki dijelovi stabla odbacuju prilikom svake iteracije što može dovesti do veće računalne potrošnje. Također RRT algoritam ima tendenciju odabira lokalno optimalnih rješenja što dovodi do suboptimalnih situacija. Stoga se u ovom algoritmu uvodi ponovno povezivanje čvorova prema njihovoj korisnosti te se neizvršeni čvorovi i njihova podstabla čuvaju i osvježavaju zajedno s ostatkom stabla. Time se postiže mogućnost rasta i održavanja beskonačno rastućeg stabla. Princip rada se temelji na uzorkovanju točke gledišta *v* u području kliznog prozora i povezivanjem sa svim točkama gledišta u blizini. Kao što je navedeno u radu [2] računa se koeficijent dobiti *g(v)* za svako gledište *v* na temelju nepromatranih ćelija (eng. *voxels*) i koeficijent troška *c(e)* za svaku vezu temeljen na njegovoj duljini. Zatim se svakom gledištu pridodaje aktivna veza *a(v)* koje se osvježavaju računanjem globalne normalizirane vrijednosti tj. računanjem maksimuma odnosa između akumuliranog koeficijenta dobiti i troška u stablu:

$$a(v)^{(i+1)} = \underset{e \in E(v)}{\operatorname{argmax}} \ \underset{v_j \in subtree(e)}{\operatorname{argmax}} \frac{\sum_{v_k \in path(v_j)} g(v_k)}{\sum_{v_k \in path(v_j)} c(a^{(i)} v_k)}$$

$$path(v) = \{v, a^{(i)}(v), a^{(i)}(a^{(i)}(v)), ..., root\}.$$

Iteriranjem tog procesa dobije se najbolji put te se izvršava prva veza, a zatim i

osvježavaju koeficijenti. Vrijednost koeficijenta dobiti bitna je za definiranje prelaska u globalno planiranje jer jednom kada niti jedan čvor nema koeficijent dobiti dovoljno velik sustav prelazi na globalno planiranje putanje. Cjelokupni prikaz sustava lokalnog planera dan je na slici 3.8 dok je na slici 3.9 dan vizualni prikaz stabla lokalnog planera. Plavom bojom su prikazane izvršene veze između čvorova dok su ostalim bojama prikazane trenutno neizvršene veze.



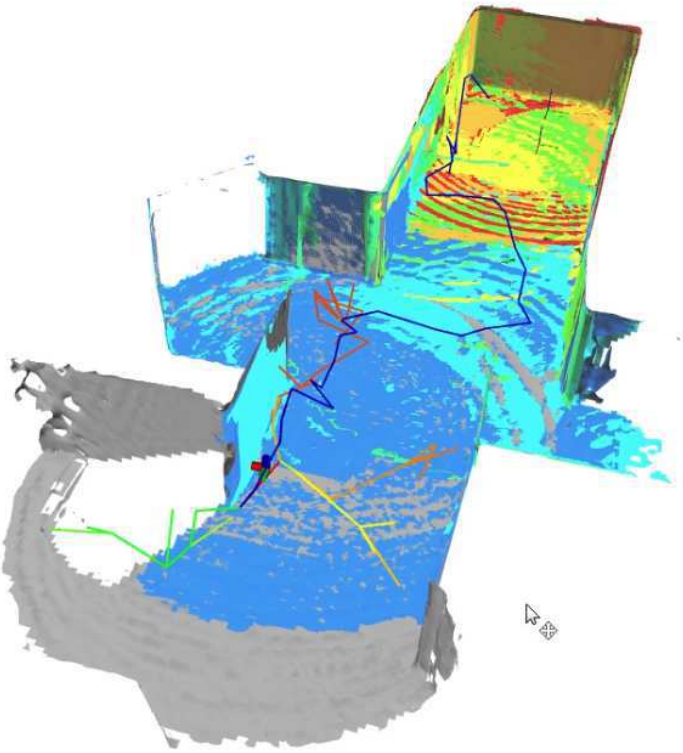
Slika 3.8: Pregled lokalnog planera [2]

3.1.5. Fronte podkarti

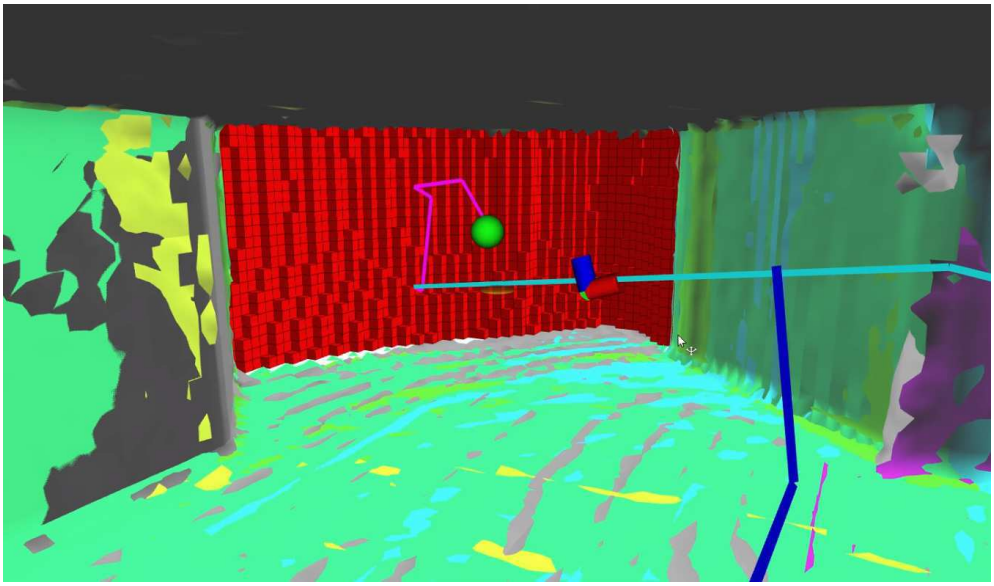
Fronte ili granice podkarte predstavljaju glavni izvor za provođenje globalnog planiranja u ovom algoritmu. Fronta je definirana kao nepoznata ćelija odnosno ćelija za koju ne znamo je li slobodna ili zauzeta, a nalazi se u susjedstvu barem jedne promatrane slobodne ćelije. Prilikom kreiranja podkarte, procijenjuju se fronte te se vezuju uz pripadajuću podkartu. Kako bi se na kraju identificirale globalne fronte, mogući kandidati se transformiraju u globalne koordinate i grupiraju u 3D prostorni *hash* set prilikom čega se uklanjaju duplikati. Također, vrši se provjera je li neka nepromatrana ćelija promatrana u drugoj podkarti što ju čini neaktivnom frontom te se ona također uklanja. Na slikama 3.10 i 3.11 može se vidjeti prikaz detektiranih fronti te njihovo grupiranje u više cijelina.

3.1.6. Globalni planer

Kao što je već navedeno, globalni planer se temelji na detekciji fronte odnosno njih uzima kao cilj za računanje nove putanje. U ovom slučaju za odabir ciljne fronte

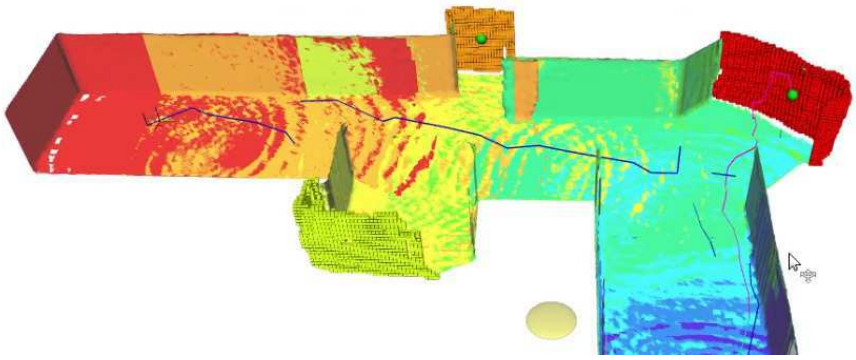


Slika 3.9: Stablo lokalnog planera



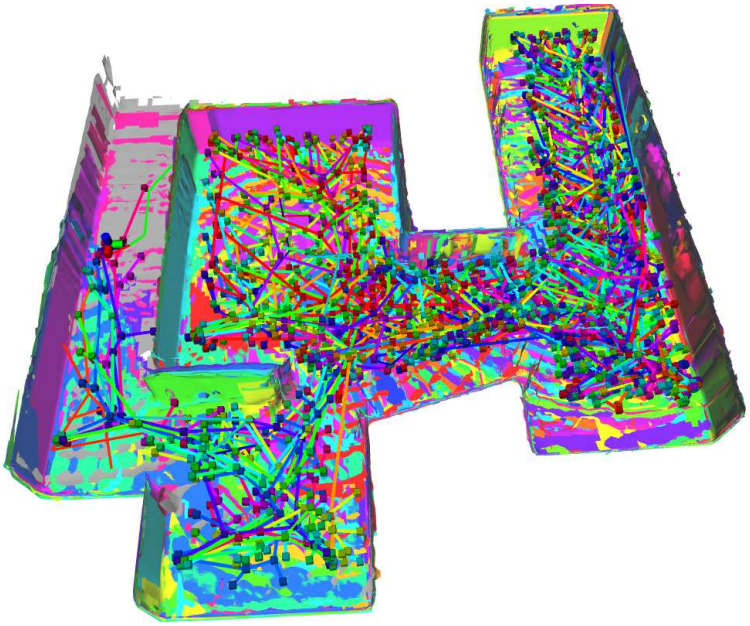
Slika 3.10: Detektirana fronta

uzima se u obzir ona koja je najbliža te se za duljinu puta uzima euklidska udaljenost. Potom se računa put do cilja, a tu opet nastupaju podkarte prostora. Naime, prilikom izgradnje podkarte ona se skeletonizira odnosno računa se graf prohodnosti te se pove-

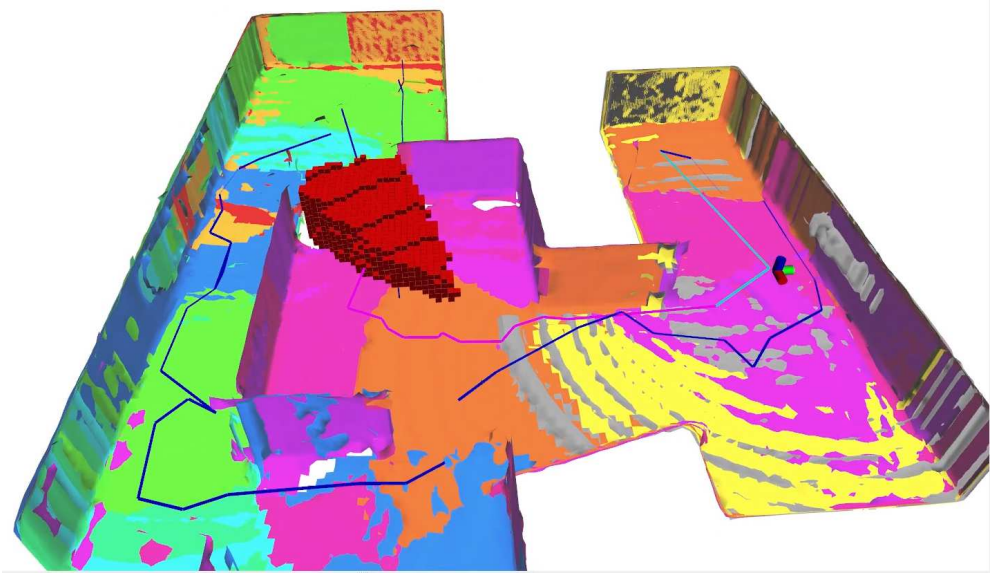


Slika 3.11: Različite grupe fronti

zuju grafovi međusobno preklapajućih podkarti. Put koji se zatim odabire za izvođenje može se pronaći pomoću raznih algoritma, a u ovom slučaju koristi se jedan od danas poznatijih i popularnijih algoritama A* algoritam. Na slici 3.12 dan je vizualni prikaz rezultata *skeleton* planera odnosno mogu se uočiti brojni čvorovi koji su međusobno povezani te se koriste u A* algoritmu za računanje optimalnog globalnog puta. Na slici 3.13 prikazan je put koji je dobiven provođenjem A* algoritma. Svijetlo plavom bojom prikazan je dio globalnog puta koji je izveden i segment koji se trenutno izvodi dok roza boja predstavlja dio koji je još potrebno proći.



Slika 3.12: Čvorovi i veze *skeleton* planera



Slika 3.13: Globalni put dobiven A* algoritmom

3.2. Procjena kvalitete algoritma

Kao što je već ranije navedeno, kreatori ovog algoritma za simulaciju i predstavljanje svojih rezultata koriste Microsoftov AirSim simulator. Svoje testiranje su proveli u dva različita okruženja, labirint veličine 40x40x3 metra uz nešto uže prolaze te okruženje zvano tuneli veličine 123x106x10 metara koje predstavlja dosta veliki i u dijelovima prostran prostor. Njihov prikaz može se vidjeti na slici 3.14. Dobivene rezultate uspoređuju s *Active3D* [2] metodom koju su također razvili istraživači s ovog instituta te trećom *GBPlanner* [5] metodom. Provedeno je deset eksperimenata za različite razine odnosno stupnjeve greške u odometriji te se u svakom slučaju koriste identične postavke i početni uvjeti. Neki od važnijih parametara i njihove vrijednosti dani su u tablici 3.1.

Maks. brzina	1	m/s	Maks. trajanje misije	15	min
LiDAR vidno polje	45	deg	LiDAR doseg	10	m
LiDAR frekvencija uzorkovanja	10	Hz	LiDAR nagib	15	deg
Trajanje kliznog prozora	10	s	Rezolucija karte	0.2	m

Tablica 3.1: Parametri eksperimenta [1]

Maksimalna brzina predstavlja maksimalnu brzinu gibanja letjelice te njena vrijednost utječe na vrijeme unutar kojeg letjelica treba doći do zadanog cilja prije nego se

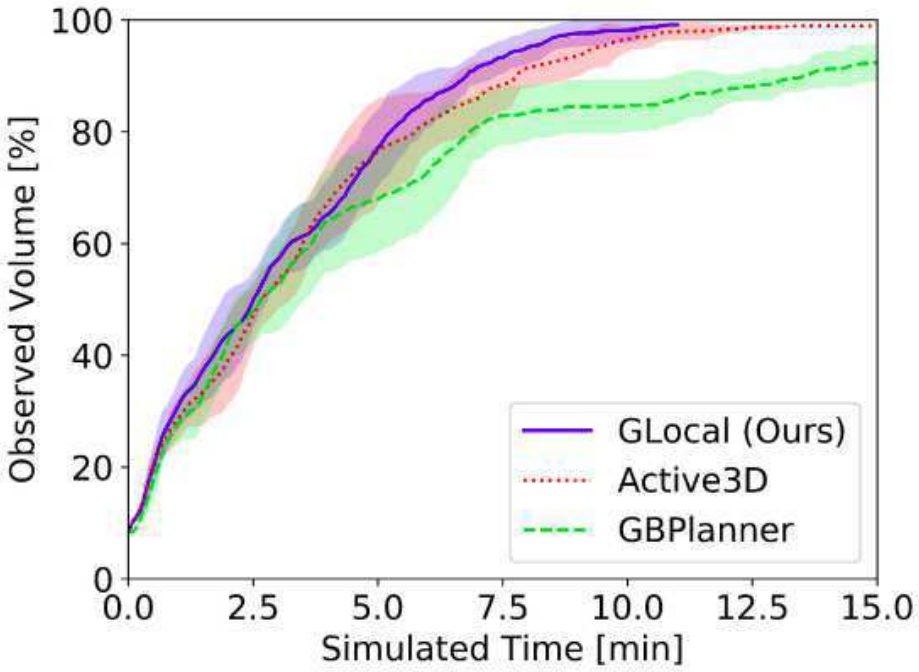
generira nova točka. Maksimalno trajanje misije definira dopušteno vrijeme trajanja misije. Ukoliko se dosegne zadana vrijednost misija će se automatski prekinuti. Sljedeći su neki od parametara LiDAR senzora poput vidnog polja senzora (eng. *field of view*, FOV), dosega lasera, frekvencije uzorkovanja podataka i nagib pod kojim je senzor postavljen u odnosu na bazu robota. Također su navedena dva parametra korištena u procesu kartiranja, trajanje kliznog prostora te rezolucija karte odnosno veličina ćelije. Svi navedeni parametri će biti detaljnije opisani kasnije.

Slika 3.14: Simulacijska okruženja korištena u testiranju [1]

3.2.1. Performanse istraživanja

Za usporedbu performansi sustava korisite se rezultati dobiveni u slučajevima bez odometrijske greške. Na slici 3.15 prikazan je graf s rezultatima količine promatranog prostora u odnosu na proteklo vrijeme.

Vidi se kako ovaj sustav ima mogućnost brzog istraživanja na razini *Active3D* sustava koji je orijentiran na performanse te se učinkovitost lokalnog planiranja može primijetiti u prvih pet minuta na prikazanom grafu. Mogućnosti i učinkovitost globalnog planera vidi se u dijelu nakon proteklih pet minuta gdje su veliki dijelovi labirinta već istraženi. Pretpostavka ovog sustava je da bilo koja fronta može dovesti do proizvoljno i potencijalno velikih dobiti te se stoga *GLocal* sustav bazira na brzom odabiru i planiranja puta do najbliže fronte. Upravo to u ovom slučaju predstavlja prednost nad *Active3D* i *GBPlanner* sustavima čiji se globalni planeri temelje na računanju koeficijenta dobiti odnosno kompromisu između dobiti i troška koji može favorizirati daljnje ciljeve koji potencijalno dovode do suboptimalnih trajektorija. Ono što također daje veliku prednost ovom sustavu je računanje svih fronti u karti što omogućuje prekid izvedbe nakon što su sve fronte istražene. Upravo navedena brojka od 98% istraženog prostora prikazuje uspješnost kompletne pokrivenosti, a osim toga i vrijeme izvedbe u odnosu na *Active3D* sustav je gotovo 20% brži.



Slika 3.15: Stanje istraženog prostora u situaciji bez odometrijske greške [1]

3.2.2. Robustnost na grešku odometrije

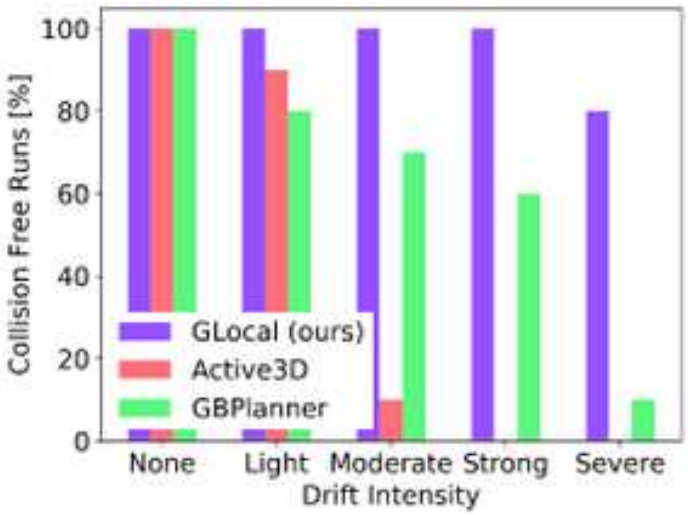
Za provjeru robustnosti sustava provodi se više različitih eksperimenata u kojima se dodaje šum na stanje brzine drona čime se simuliraju različite razine grešaka u odometriji dobivenih suvremenim metodama estimatora odometrije. Greške su podijeljene u četiri kategorije kao što je prikazano u tablici 3.2.

Greška pomaka	Slaba	Srednja	Snažna	Veoma snažna
Pozicija [m]	0.31 ± 0.11	0.56 ± 0.23	1.14 ± 0.74	2.76 ± 1.39
Rotacija [°]	0.95 ± 0.27	2.10 ± 0.42	4.59 ± 1.35	8.85 ± 5.63

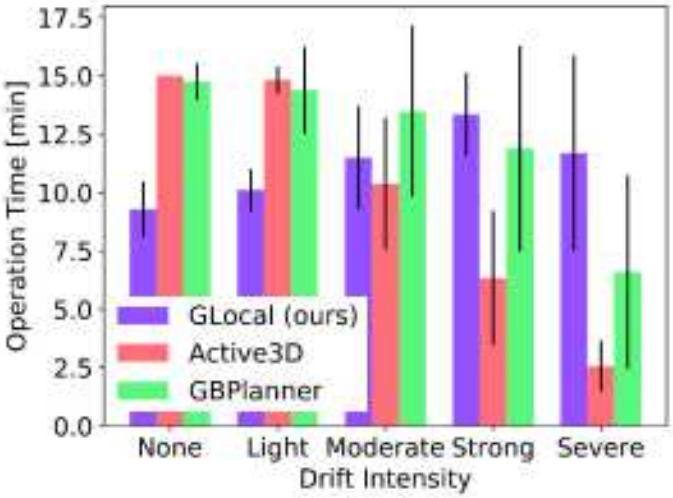
Tablica 3.2: Iznos odometrijske greške pri prijeđenih 100 metara [1]

Razina greške mjeri se u stanju pozicije drona te njegovoj orijentaciji nakon prođenih 100 metara. Rezultati srednje vrijednosti i standardne devijacije dobiveni su nakon deset provedenih eksperimenata za svaku kategoriju. Prema podacima iz istraživanja [16] navedeno je kako suvremeni estimatori akumuliraju oko 0.6 metara greške u poziciji te oko 3 stupnja u orijentaciji na prođenih 100 metara. U ovom eksperimentu to predstavlja drugi slučaj odnosno umjerenu grešku odometrije. Na slikama 3.16, 3.17 i

3.18 prikazani su rezultati sva tri sustava u različitim slučajevima greške odometrije.

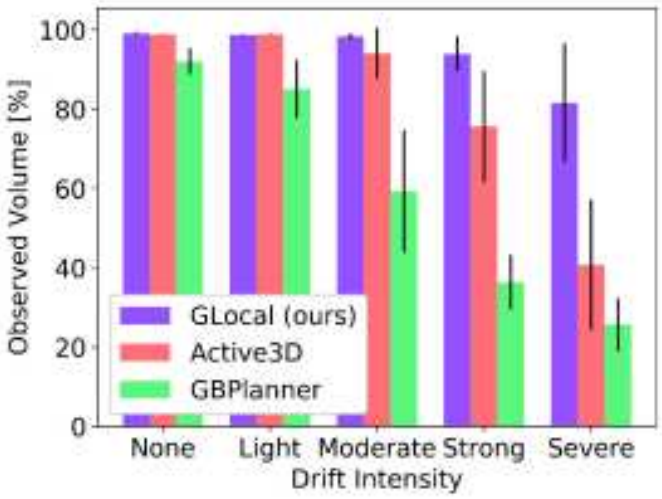


Slika 3.16: Uspješnost provedenih eksperimenata bez kolizije [1]



Slika 3.17: Ukupno vrijeme trajanja misije [1]

Na slici 3.16 prikazan je graf s rezultatima postotka uspješno provedenih eksperimenata bez kolizije. Već u situaciji s manjom greškom GLocal sustav daje bolje rezultate nego ostala dva, a pogotovo se ta razlika može vidjeti na sljedećoj, srednjoj, razini greške. Graf sa slike 3.17 prikazuje ukupno vrijeme potrebno za izvršavanjem zadatka te tu vidimo efikasnot GLocal sustava već i u situaciji bez ikakve odometrijske greške. Može se primijetiti kako ostala dva sustava imaju manje operacijsko vrijeme



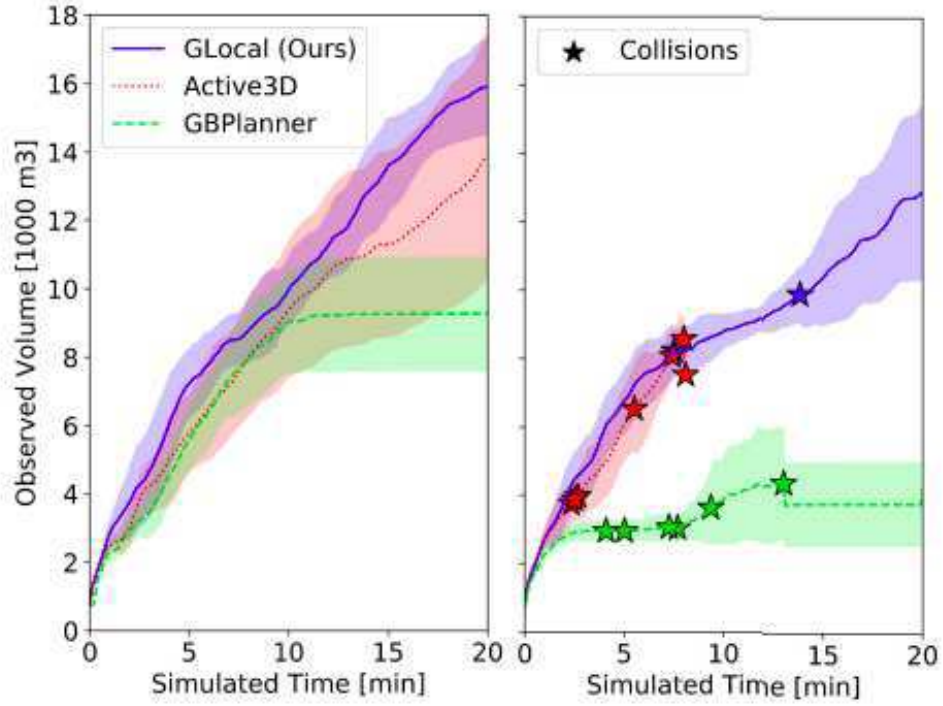
Slika 3.18: Postotak istraženog okruženja [1]

u slučajevima jačih inteziteta greške međutim to je rezultat kolizija te zbog toga prijevremenog završetka misije. Zadnji graf sa slike 3.18 prikazuje postotak istraženog prostora te su tu pri manjoj odometrijskoj greški sva tri sustava relativno sumjerljiva, ali pri većeme intezitetu opet vidimo superiornost GLocal sustava. Vidi se kako u svim slučajevima *GLocal* sustav ima prednost nad ostala dva. Štoviše, čak i prilikom malo veće greške i dalje uspješno izvršava zadanu misiju, a i u slučaju sa poprilično velikim stupnjem greške pokazuje se vrlo kvalitetno. Bolja usporedba je prikazana u slučaju istraživanja tunela koji su poprilično dugi i razgranati te je korištenje globalnog planera dosta relevantnije i potrebitije nego u slučaju labirinta. Na slici 3.19 prikazana su dva slučaja, jedan bez greške (lijevi) i drugi s umjerenom greškom odometrije (desni).

Primjetno je kako u oba slučaja *GBPlanner* sustav nakon nekog vremena dođe probleme te ne uspijeva istražiti ni približnu količinu prostora kao *GLocal* sustav. Navodi se kako najviše problema nastupa pri prijelazu iz jednog ogranka tunela u drugi. Kao i u slučaju s labirintom i ovdje *GLocal* sustav ima vrlo dobre performanse u početku za što je zaslužan lokalni planer. Osim tog dijela, ovaj sustav se vrlo dobro snalazi i kasnije kada naglasak stoji na globalnom planeru, a upravo je to zbog činjenice da je u mogućnosti identificirati fronte u različitim dijelovima karte.

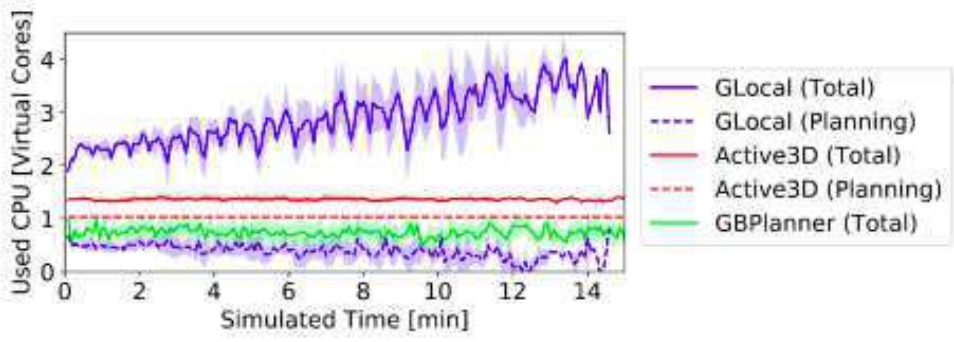
3.2.3. Troškovi računalnih resursa

Potrošnja računalnih resursa predstavlja veoma važnu ulogu u operacijama s robotskim mobilnim platformama te je bitno da sustav koji se koristi nije prezahtjevan za



Slika 3.19: Istraženi volumen i količina kolizija [1]

dostupne mogućnosti jer u suprotnom može doći do problema s izvedbom i provođenjem misije. Kada se promatraju potrebni resursi za pokretanje ovog sustava dobivaju se nešto lošiji rezultati u usporedbi s ostalim algoritmima. U svojim simulacijskim eksperimentima autori koriste Intel i9 9900K procesor (snaga procesora je u ovom slučaju najbitnija, odnosno njegovi resursi su ključni za pokretanje navedenih operacija) te se ispostavlja da GLocal kao cjelina troši najviše resursa dok najbolje rezultate daje GBPlanner sustav. Na slici 3.20 može se vidjeti kako za potrebe planiranja GLocal daje vrlo dobre rezultate, međutim, rad cjelokupnog sustava, prvenstveno optimizacija grafa poze karti, uvelike uvećava potrošnju resursa te se to ipak treba uzeti u obzir.



Slika 3.20: Korištenje CPU resursa [1]

Zanimljiva je još jedna predstavljena analiza, a to je odnos potrebnog vremena za detekciju i pronalaženje globalnih fronti sukladno napretku istraživanja odnosno proteklom vremenu istraživanja. Tu je dan prikaz metode temeljene na frontama podkarti koja je implementirana u GLocal sustavu u usporedbi s metodom pronalaska fronti na trenutnim kolekcijama podkarti iz rada [6], te treća opcija koja se svodi na spajanje svih podkarti u globalnu kartu na kojoj se zatim mogu ispitati globalne fronte. U ovom slučaju primjetna je ogromna prednost metode implementirane u GLocal sustavu.

4. Prilagodba na Gazebo simulacijsko okruženje

Kao što je već navedeno jedan od glavnih zadataka ovog rada je prilagoditi sustav na rad u Gazebo simulacijskom okruženju. Za to je potrebno dobro se upoznati s korištenim GLocal sustavom te raspoznati koje su promjene potrebne kako bi zadatak bio uspješno izvršen. Također, potreban je i novi simulacijski sustav za upravljanje dronom te simuliranje senzora potrebnih za rad sustava.

4.1. UAV Gazebo simulalcija

Za model drona i simuliranje njegovih senzora, kontrolera, motora i ostalog koristi se *uav_ros_simulation* repozitorij koji se može preuzeti s LARICS-ove (Laboratorij za robotiku i inteligentne upravljačke sustave) Github stranice [11]. On sadrži potrebnu kolekciju paketa od kojih su najvažniji RotorS i Ardupilot koji služe za modeliranje i upravljanje konkretnim lejelicama. Za lakšu instalaciju koristi se postojeća Docker slika te se ostatak rada temelji na njoj što olakšava cjelokupni proces, uklanja mogućnost problema i grešaka pri instalaciji te omogućuje jednostavniji daljnji razvoj. Uz navedene pakete koristi se još i paket *velodyne_simulator* koji sadrži URDF opis i Gazebo dodatke za simuliranje vrlo poznatih i popularnih Velodyne LiDAR senzora. U ovom radu kao letjelica se koristi model Kopterworx drona prikazanog na slici 4.1 te se njegov Gazebo model može vidjeti na slikama 4.2 i 4.3.

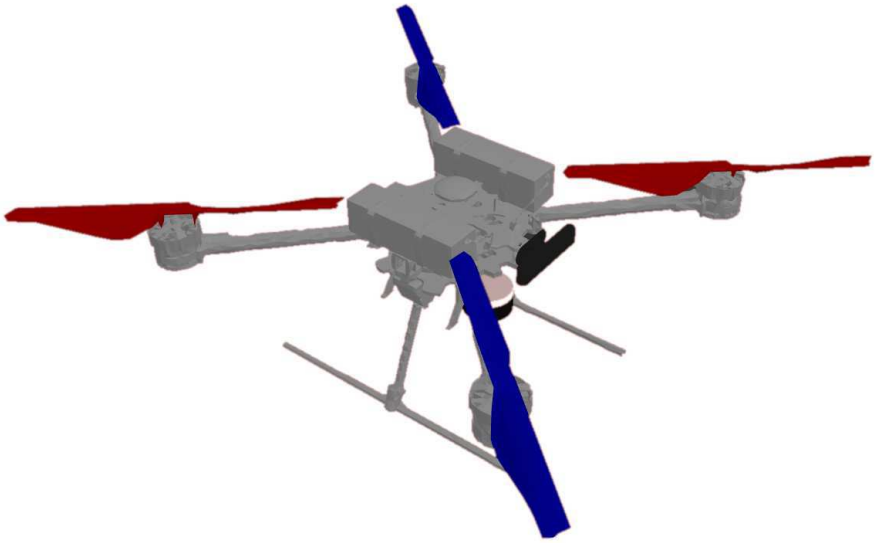
Uz to se koristi i Velodyne LiDAR senzor montiran s donje strane letjelice koji je detaljnije prikazan na slici 4.3. Simulacijski parametri LiDAR senzora mogu se jednostavno mijenjati unutar URDF datoteke. U nastavku su dana dva isječka URDF datoteke s nekim od parametara. Prvi isječak je iz URDF datoteke za model drona te u njemu definiramo definiramo položaj senzora u odnosu na bazu drona

```
<xacro:property name="horizontal_velodyne_origin">
  <origin xyz="0.08 0. -0.1578" rpy="0. 0. 0." />
</xacro:property>
```

dok je drugi isječak dio URDF datoteke za opis modela LiDAR senzora.

```
<xacro:macro name="lidar-x" params="
  *origin
  parent:=base_link
  name:=leddar
  topic:=/lidar_points
  hz:=10
  lasers:=32
  samples:=200
  collision_range:=0.3
  min_range:=0.1
  max_range:=10.0
  noise:=0.008
  min_horizontal_angle:=-3.14159
  max_horizontal_angle:=3.14159
  min_vertical_angle:=-0.392699082
  max_vertical_angle:=0.392699082
  gpu:=true">
```

Slika 4.1: Kopterworx Hammer X8B

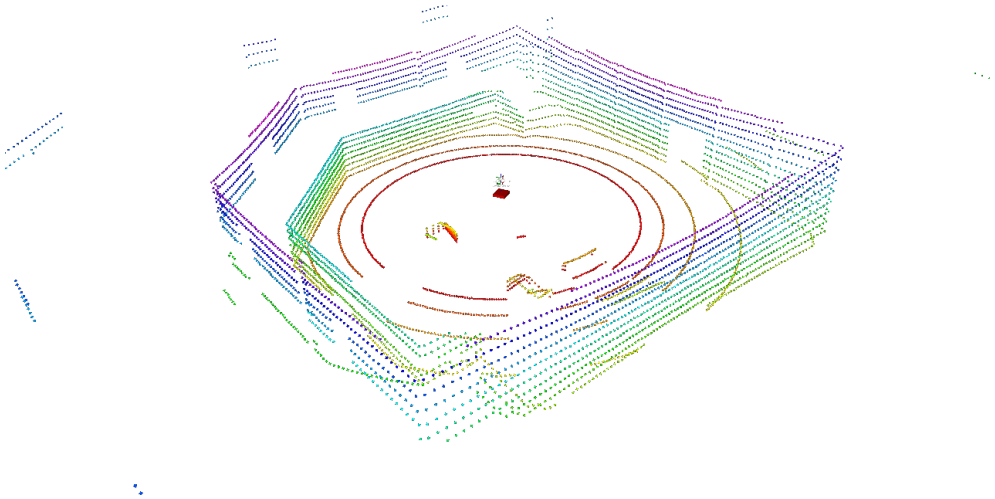


Slika 4.2: Model drona

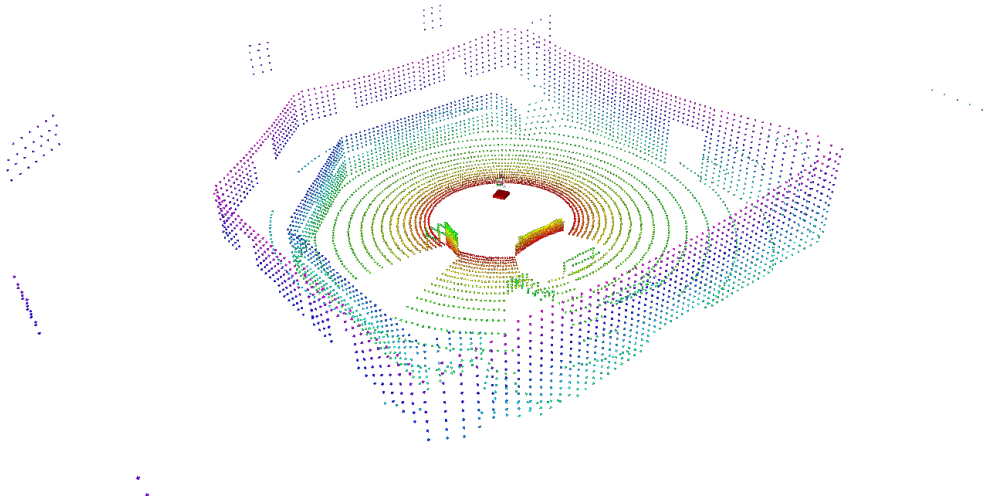
Slika 4.3: Prednji prikaz modela drona i LiDAR senzora

Neki od bitnijih parametara koji opisuju općeniti 3D LiDAR senzor su:

– **Broj laserskih zraka:** temelj svakog LiDAR senzora su upravo laserske zrake pomoću kojih se detektira prepreka, a što je veći njhov broj dobije se bolja vertikalna rezolucija te time i bolji rezultati. Kada se govori o količini laserskih zraka prvenstveno se misli na njihovu vertikalnu raspodjelu. Moderni 3D LiDAR senzori uglavnom koriste 16 ili 32 laserske zrake u vertikalnoj ravnini. Na slikama 4.4 i 4.5 može se vidjeti razlika između LiDAR senzora sa 16 zraka i onoga s 32. Primjetno je kako se u slučaju s 32 lasera dobije puno gušća raspodjela te stoga i veća količina korisnih informacija. Veličina uzorka je povećana zbog lakše vizualizacije.



Slika 4.4: LiDAR senzor sa 16 laserskih zraka

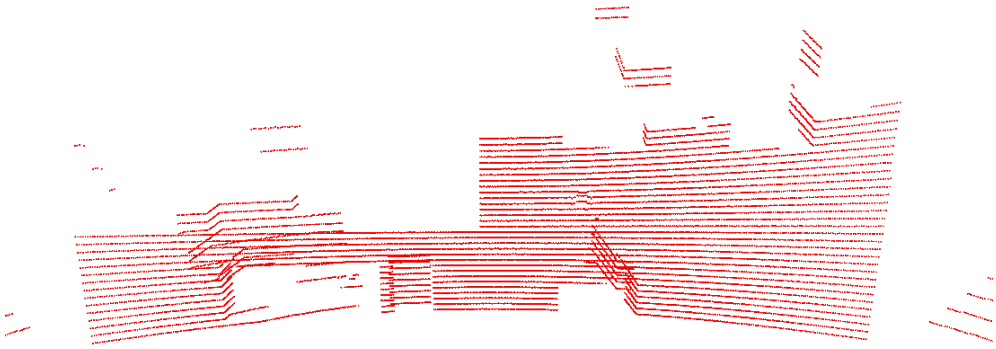


Slika 4.5: LiDAR senzor sa 32 laserskih zraka

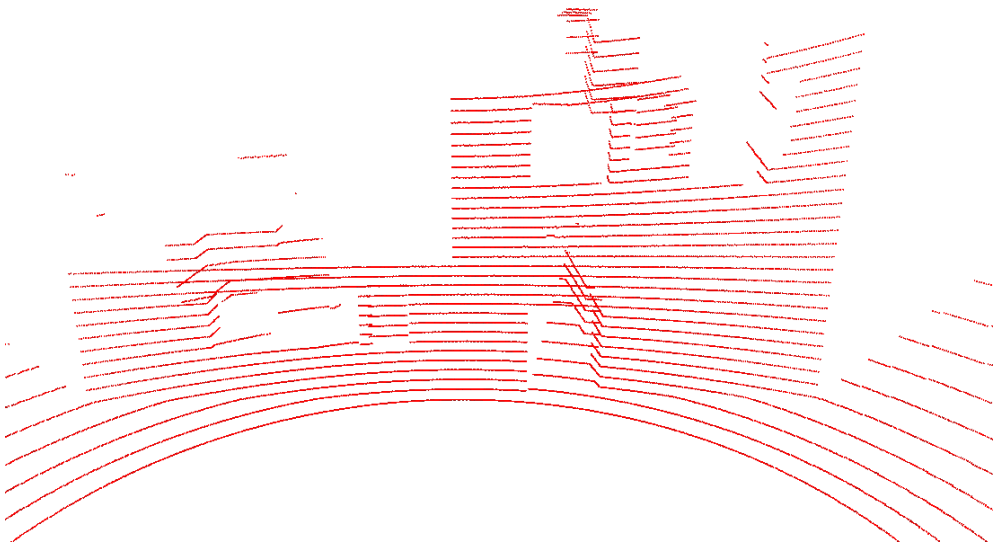
– **Broj uzoraka:** ovaj broj nam govori koliko uzoraka dobijemo od svakog lasera odnosno broj uzoraka po horizontalnoj ravnini te veći broj uzoraka dovodi do bolje horizontalne rezolucije.

– **Vidno polje:** kao što i sam naziv govori ovaj parametar definira prostor koji obuhvaćaju laserske zrake, a defininran je u horizontalnoj i vertikalnoj ravnini. 3D LiDAR senzori uglavnom imaju 360 stupnjeva horizontalnog vidnog polja dok je vertikalno vidno polje poprilično manje, a uglavnom se kreće između 20 i 50 stupnjeva. Naravno, skuplji i kompleksniji senzori mogu imati i šire vidno polje. Na slikama 4.6 i 4.7 vidi se vizualizacija laserskih zraka u slučaju

LiDAR senzora s vertikalnim vidnim poljem od 25 te 45 stupnjeva.



Slika 4.6: Vertikalno vidno polje od 25 stupnjeva



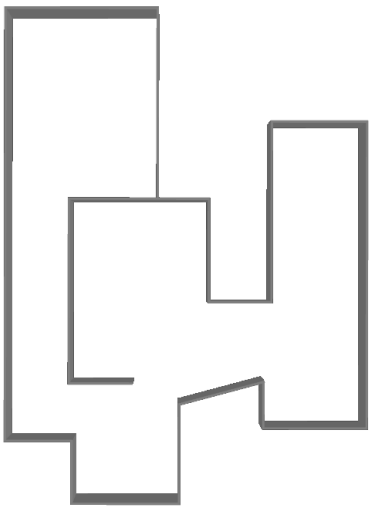
Slika 4.7: Vertikalno vidno polje od 45 stupnjeva

Može se uočiti kako se u slučaju vidnog polja od 45 stupnjeva prikuplja dosta više informacija iz prostora odnosno dobiva se šira slika okoline. To je osobito bitno u primjenama drona pri istraživanju i inspekciji prostora jer fokus nije

usredotočen na neki manji dio okoline već je cilj prikupiti što više informacija odjednom.

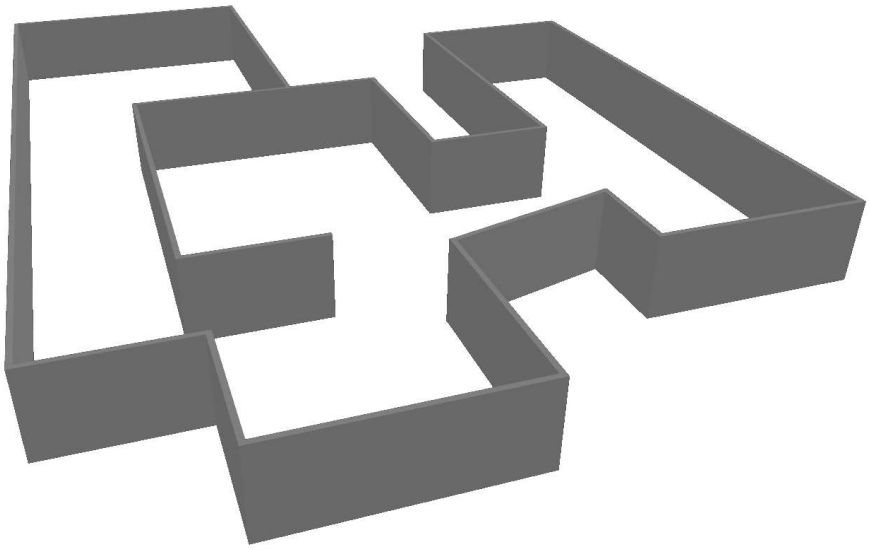
– **Udaljenost:** svaki LiDAR senzor je ograničen s udaljenošću na kojoj može detektirati prepreku, a te brojke se kreću od nekoliko desetaka metara pa čak i do stotinjak metara. Također, postoji i donja granica koja definira minimalnu udaljenost za sigurno i točno detektiranje prepreke.

Od ostalih postavki za pokretanje Gazebo simulatora mogu se još navesti uključivanje i modificiranje šuma na senzoru za odometriju pomoću čega se mogu simulirati različiti uvjeti koji su mogući u realnom svijetu. Također je potrebno definirati okruženje u kojemu se letjelica nalazi odnosno potrebno je definirati i modelirati Gazebo svijet. Za te potrebe napravljena su dva modela inspirirana onima navedenim u pripadajućem radu, labirint i tuneli, čiji se prikazi mogu vidjeti na slikama 4.9 i 4.11. Prilikom provođenja eksperimenata u navedenim modelima postoji i strop, ali je zbog bolje vizualizacije na ovim slikama uklonjen.

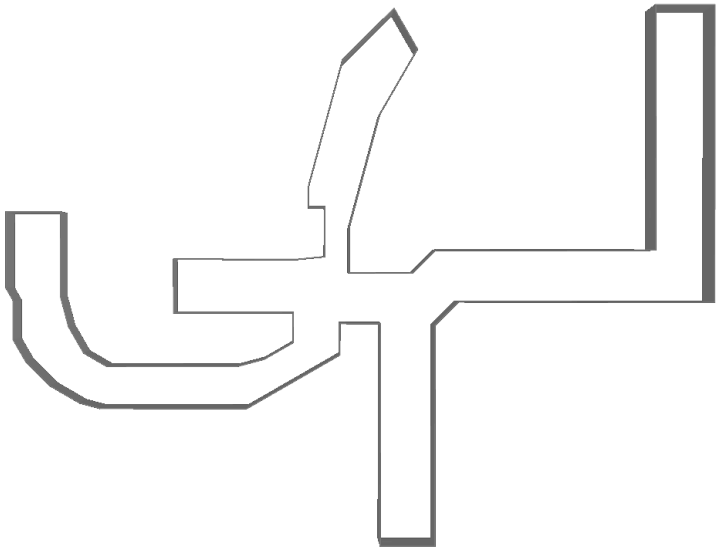


Slika 4.8: Labirint odozgo

Nakon što su definirani potrebni parametri simulacije ona se može pokrenuti pomoću pripadajuće Python skripte koja zatim pokreće sve potrebne ROS pakete.



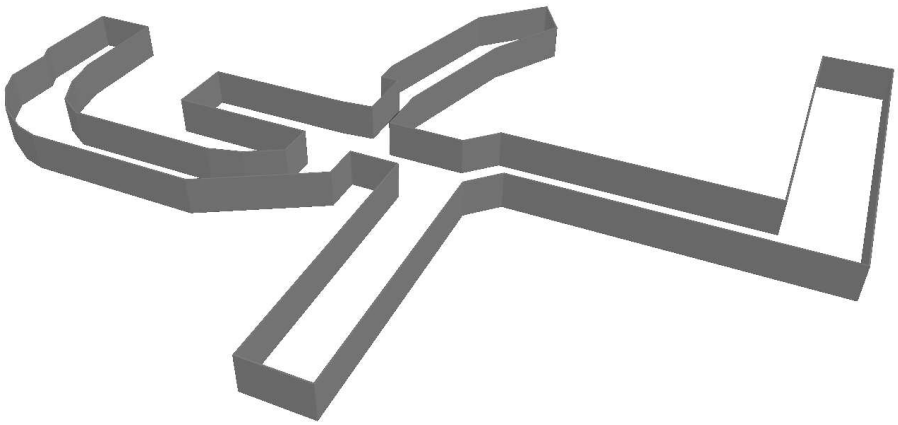
Slika 4.9: Labirint sprijeda



Slika 4.10: Tuneli odozgo

4.2. GLocal paket

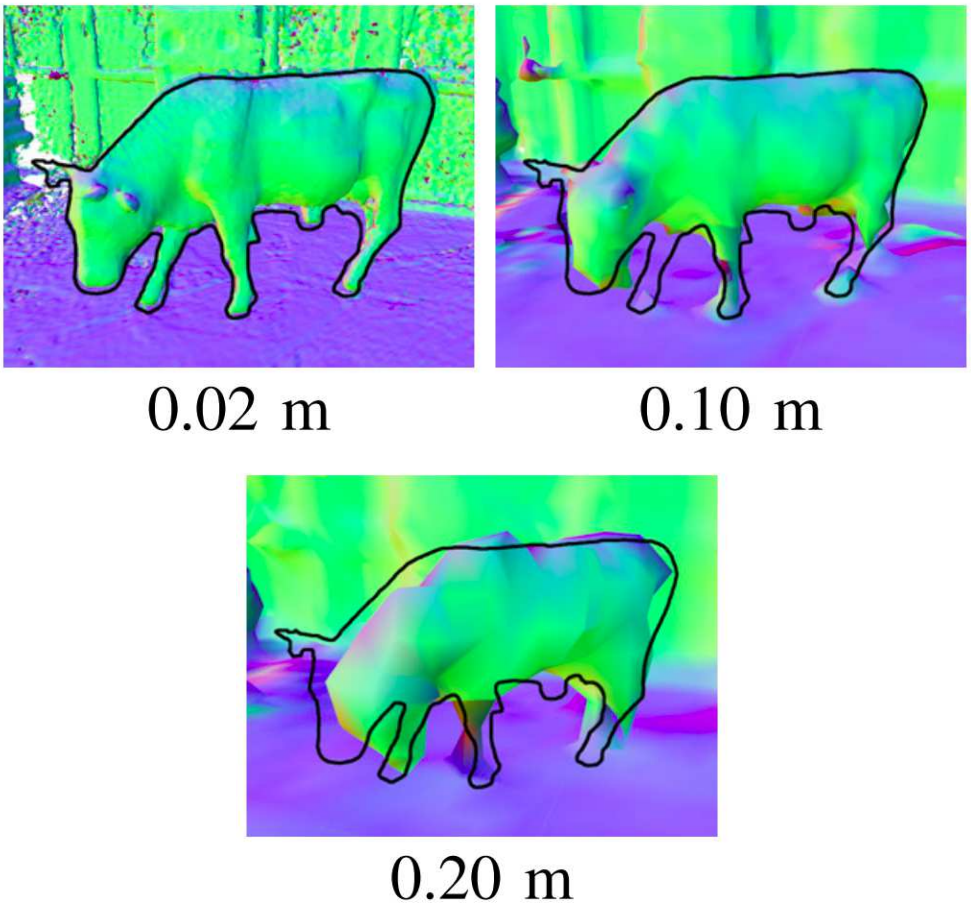
GLocal paket preuzet je s Github stranice Laboratorija za autonomne sustave s fakulteta ETH Zurich [12]. Sadrži pripadajuće pakete za kartiranje i navigaciju na kojima je sustav temeljen te još nekoliko paketa potrebnih za instalaciju i pokretanje. Kao što je već navedeno ovaj paket se temelji na AirSim simulatoru te su određene stvari pri-



Slika 4.11: Tuneli sprijeda

lagođene njemu, a uz to postoji i konfiguracijska datoteka koju je potrebno izmijeniti sukladno potrebama. U toj datoteci definirani su promjenjivi parametri koji se koriste u algoritmima za kartiranje i planiranje putanje i navigaciju kroz prostor. U nastavku su navedeni neki od parametara iz konfiguracijske datoteka koji su važniji i za koje postoji najveća mogućnost potrebe za modificiranjem:

- **Veličina ćelije (eng. *voxel_size*):** ovaj parametar definira kolika će biti veličina jedne ćelije u procesu kartiranja prostora. Ćelija predstavlja jedan uzorak u trodimenzionalnom polju te u ovom slučaju predstavlja rezultat detektirane prepreke od strane LiDAR senzora. Smanjivanjem veličine ćelije moguće je dobiti vrlo precizan i realističan prikaz okoline, međutim, to dovodi do velike potrošnje računalnih resursa te sporijeg izvođenja algoritma. Stoga je potrebno paziti na tu vrijednost jer ukoliko korišteno računalo nije dovoljno snažno sustav se neće kvalitetno izvoditi odnosno doći će do preopterećenja i čak mogućeg pada programa. U dalje provedenim eksperimentima veličina ćelije je postavljena na 0.2 metara što daje dovoljno vjeran prikaz okoline za trenutne potrebe, a ne utječe na otežano izvođenje programa. Na slici 4.12 može se vidjeti prikaz prostora s obzirom na različito definirane veličine ćelije.
- **Radijus prohodnosti (eng. *traversability_radius_**):** ovaj parametar je ključan za provjeru prohodnosti prilikom planiranja i izvođenja putanje robota. On definira područje u odnosu na prepreku u kojem se može nalaziti zadani cilj kretanja. To je veoma važno kako ne bi došlo do neželjene kolizije drona s



Slika 4.12: Prikaz detalja s obzirom na veličinu ćelije [4]

okolinom. Ovaj parametar je ostavljen na zadanoj vrijednosti, ali ga je bitno spomenuti zbog njegove važnosti.

– **Područje interesa (eng. *region_of_interest*):** još jedan parametar koji ima ulogu ograničavanja prostora kretanja, ali u ovom slučaju on služi kako bi se definiralo područje istraživanja. Definira se zadavanjem koordinata u x, y i z smjeru čime se definira kvadar interesa u prostoru. Sve ono što se nalazi izvan tog prostora bit će ignorirano te letjelica neće niti dobiti misiju izvan tog područja. Ovo je posebno korisno u velikim i otvorenijim okruženjima gdje se usredotočuje samo na jedan dio. Na slici 4.13 prikazan je rezultat misije u kojoj je područje interesa definirano na manjem području od veličine okruženja. Ako se ova slika usporedi sa slikom 4.11 može se uočiti da nedostaje desni dio odnosno da to područje nije istraženo. Upravo je to rezultat toga da je istraživanje okoline ograničeno u tom dijelu karte.

– **Maksimalna duljina zrake (eng. *max_ray_length*):** kao što i sam naziv go-

Slika 4.13: Istraživanje tunela s definiranim granicama

vori, pomoću ovog parametra definira se duljina zrake LiDAR senzora. Dakle, čak iako senzor možda ima laserske zrake veličine od npr. 50 metara pomoću ovog parametra moguće je smanjiti udaljenost koja će se uzimati u obzir. S definiranim većim udaljenostima odnosno većom duljine zrake malo se povećava kompleksnost izvođenja te je u ovom slučaju parametar postavljen na 10 metara. Autori ovog sustava u svojim eksperimentima također koriste tu duljinu, a i prilikom testiranja s različito definiranim veličinama došlo je do zaključka da je ta vrijednost korektna za kvalitetno izvođenje misije to jest nema potrebe za većim udaljenostima.

– **Parametri LiDAR senzora:** ovdje postoji nekoliko parametara koji su prvenstveno vezani za korišteni sustav odnosno LiDAR senzor. Potrebno je definirati parametre koji su navedeni ranije prilikom objašnjavanja LiDAR senzora stoga su ovdje samo navedeni. To su *vertical_fov* i *horizontal_fov* za definiranje vid-

nog polja, *vertical_resolution* i *horizontal_resolution* koji definiraju broj laserskih zraka u vertikalnog ravnini i broj uzoraka po laseru. Također osim ovih parametara potrebno je definirati i transformacijsku matricu između senzora i baze (eng. *baselinka*) robota. To je potrebno zbog algoritma koji se koristi za optimiziranje orijentacije drona u slučaju da se LiDAR senzor nalazi postavljen pod kutem jer tada laserske zrake s jedne strane imaju niži nagib, a s druge viši što uzrokuje različito prikupljanje informacija iz okoline. U provedenim eksperimentima senzor je postavljen u ravnini te je u kodu napravljena promjena tako da se prilikom zadavanja cilja uvijek zadaje ista orijentacija čime se ukljanjaju nepotrebne rotacije drona.

– **Parametri lokalnog planera:** na raspolaganju je nekoliko parametara koji utječu na funkcionalnost lokalnog planera. Prvo se definira koji tip planera se koristi. Taj parametar je ostavljen nepromijenjen te se koristi planer temeljen na RRT* algoritmu. Tu su još *min_path_length* i *max_path_lenght* koji definiraju minimalne i maksimalne udaljenosti za sljedeći cilj. U provedenim ekperimentima ti su parametri ostavljeni po već zadanim postavkama, ali u većim okruženjima slobodno se mogu povećati njihove vrijednosti. Još jedan važniji parametar je *termination_max_gain* koji definira granicu prilikom evaulacije kvaliteta rezultata lokalnog planera. Ukoliko je dobiveni koeficijent manji od zadanog razmotrit će se prelazak na globalni planer. Prikaz svih parametara može se vidjeti u sljedećem isječku iz konfiguracijske datoteke:

```
local_planner:
  type: rh_rrt_star
  verbosity: 4

  # RH-RRT-Star params
  sampling_range: 7 # m
  min_path_length: 1.0 # m
  min_sampling_distance: 1.0 # m
  max_path_length: 3.0  # m
  path_cropping_length: 0.3 # m
  max_number_of_neighbors: 30
  maximum_rewiring_iterations: 100
  traversability_radius: *traversability_
                          radius_local_planner
```

```
# Termination
reconsideration_time: 2 # s
termination_max_gain: 1000
terminaton_min_tree_size: 0
DEBUG_number_of_iterations: -1  # -1 to
                                # turn off

# Lidar Model
ray_length: *max_ray_length  # m
vertical_fov: 45  # deg
horizontal_fov: 360 # deg
vertical_resolution: 32 # 64 / 32
horizontal_resolution: 200 # 1024 / 200
ray_step: 0.4 # m
downsampling_factor: 1
num_yaw_samples: 4
T_baselink_sensor:  # 15 deg pitch
 - [1.0,  0.0000000,  0.0000000, 0.0800000]
 - [0.0,  1.0000000,  0.0000000, 0.0000000]
 - [0.0,  0.0000000,  1.0000000, -0.157800]
 - [0.0,        0.0,        0.0,       1.0]

# Visualization
visualize_tree: true
visualize_gain: true
visualize_text: true
visualize_visible_voxels: true
visualize_executed_path: true
```

– **Parametri globalnog planera:** kao i kod lokalnog planera i ovdje se definira vrsta algoritma koji se koristi, a to je *skeleton* planer te uz njega A* algoritam za određivanje optimalne trajektorije. Uz to je moguće definirati parametre vezane uz različite udaljenosti kao i broj iteracije prilikom izvođenja A* algortima. Još jedan vrlo koristan parametar je *min_frontier_sze*, a njime se definira broj detektiranih fronti unutar jedne skupine da bi se ona uzela u obzir prili-

kom planiranja putanje. Naime, ukoliko se koristi premali broj, to bi dovelo do ogromnog broja fronti, a možda se radi samo o nekoliko ćelija za koje nema potrebe da se uzimaju u obzir. Prikaz svih parametara može se vidjeti u sljedećem isječku iz konfiguracijske datoteke:

```
global_planner:
  type: "skeleton"
  verbosity: 4

  # Skeleton Planner
  use_centroid_clustering: true
  centroid_clustering_radius: 1.0
  use_path_verification: true
  safety_distance: 0.3
  path_verification_min_distance: 1.0
  goal_search_steps: 5.0
  goal_search_step_size: 1.0
  backtracking_distance_m: 5.0  # Set to -1
                                # to disable
  max_replan_attempts_to_chosen_frontier: 3
  max_closest_frontier_search_time_sec: 10

  # Submap Frontiers Evaluator
  min_frontier_size: 100
  submaps_are_frozen: true

  # Visualization
  visualize_frontiers: true
  visualize_frontier_text: true
  visualize_inactive_frontiers: true
  visualize_executed_path: true
  visualize_candidate_goals: true
  visualize_planned_path: true

  # A star search along linked sparse
  # skeleton graph submaps
```

```
skeleton_a_star:
  traversability_radius: *traversability_
                          radius_global_
                          planner
  max_num_a_star_iterations: 6000
  max_num_start_vertex_candidates: 20
  max_num_end_vertex_candidates: 5
  linking_num_nearest_neighbors: 3
  linking_max_distance: 5.0
```

4.3. Problemi i potrebne modifikacije

Iako autori ovog rade tvrde kako se njihov sustav može bez problema koristiti na bilo kojem drugom simulatoru ili stvarnoj letjelici to ipak nije baš tako jednostavno. Potrebno je uočiti neke stvari koje je potrebno modificirati kako bi sve radilo kako treba:

– **Promjene u *launch* datoteci za pokretanje:** pošto se ne koristi AirSim simulator potrebno je zakomentirati ili obrisati dijelove koji se odnose na to. U nastavku je prikazan navedeni isječak iz *launch* datoteke.

```
<!-- airsim client -->
<node name="airsim_simulator" pkg="unreal_
                                   airsim"
    type="airsim_simulator_node"
    required="true" output="screen"
    args="-alsologtostderr">
  <rosparam
        file="$(find glocal_exploration_
        ros)/config/$(arg airsim_config)"/>
  <rosparam
        file="$(find glocal_exploration_
        ros)/config/$(arg drift_config)"/>
</node>
```

Treba imati na umu da za pokretanje misije istraživanja GLocal sustav očekuje poruku *true* na *topicu* pod nazivom */simulation_is_ready*, a kako se ta poruka dobiva od strane AirSim simulatora koji je ugašen potrebno je ili napraviti mo-

difikacije u UAV Gazebo simulatoru da se objavi navedena poruka kada je simulator pokrenut i spreman za rad ili se može jednostavno objaviti ta poruka preko terminala naredbom *rostopic pub /simulation_is_ready std_msgs/Bool ̈data: true¨*. Unutar *launch* datoteke potrebno je i definirati nazive tri *topic-a - pointcloud* za rezultate dobivene s LiDAR senzora, *odometry* za rezultate odometrije i *command/pose* koji služi za zadavanje ciljeva kretanja drona:

```
<remap from="~pointcloud" to="/red/velodyne_
                                points"/>
<remap from="odometry" to="/red/odom_world"/>
<remap from="command/pose" to="/red/goal_pose"/>
```

– **Tip poruke za zadavanje poze drona:** kako bi se dron kretao koristi se *topic /red/tracker/input_pose* na koji GLocal sustav objavljuje ciljne točke. Međutim, u kodu za GLocal sustav autori koriste tip poruke *geometry_msgs/Pose.h* dok UAV Gazebo simulator očekuje tip poruke *geometry_msgs/PoseStamped.h*. Tip poruke *PoseStamped* u sebi sadrži tip poruke *Pose*, ali uz to još i zaglavlje odnosno *Header* poruku. Stoga je napravljena jednostavna promjena u kodu gdje je navedni tip poruke zamijenjen i objavljuje se pripadajuća vrsta. Dio koda zadužen za taj dio vidimo u sljedećem isječku:

```
void GlocalSystem::publishTargetPose() {
  // publish the target pose.
  geometry_msgs::PoseStamped msg;
  tf2::Quaternion q;
  q.setRPY(0, 0, target_yaw_);
  msg.position.x = target_position_.x();
  msg.position.y = target_position_.y();
  msg.position.z = target_position_.z();
  msg.orientation.x = q.x();
  msg.orientation.y = q.y();
  msg.orientation.z = q.z();
  msg.orientation.w = q.w();
  target_pub_.publish(msg);
```

– **Kretanje drona neusklađeno s zadanim ciljevima:** prilikom testiranja uočene su čudne kretnje drona odnosno letjelica nikada ne bi išla prema zadanom cilju već u nekom drugom smjeru. Stoga je isprobano ručno zadavanje cilja na

topic-u /red/tracker/input_pose te je uočeno kako su određene koordinate obrnute npr. zadavanjem pozitivne x koordinate letjelica bi se kretala u negativnom y smjeru. Prvenstveno je taj problem bio riješen tako da su u dijelu koda gdje se objavljuju koordinate sljedećeg cilja modificirane njihove vrijednosti s obzirom na prikupljena saznanja. Međutim, daljnjim testiranjem uočeno je da do greške dolazi zbog neusklađenosti u koordinatnom sustavu koji koristi GLocal paket za definiranje ciljne točke s koordinatnim sustavom za zadavanje ciljne točke u paketu za simulaciju drona i Gazebo svijeta. Naime u GLocal paketu glavni koordinatni sustavi koji se koriste su */odom*, */mission* i */world* te su oni međusobno poravnati. U paketu za simulaciju drona naredba za zadavanje točke kretanja i mjerenje odometrije zadaje se u odnosu na koordinatni sustav čije je ishodište u početnoj poziciji drona pri pokretanju simulacije. Stoga je napravljena mala skripta koja prima dvije poruke, odometrijska iz paketa za simulaciju drona i naredba ciljne točke iz GLocal paketa, te se one transformiraju u drugi, pripadajući koordinatni sustav i zatim prosljeđuju dalje. U nastavku je prikazan kratki isječak kako otprilike izgleda pripadajuća skripta.

```
def __init__(self):
    rospy.init_node('transformation')
    self.pub_odom = rospy.Publisher('/red/odom_
                world', Odometry, queue_size=0)
    self.pub_goal = rospy.Publisher('/red/tracker/
                input_pose', PoseStamped,
                queue_size=0)
    rospy.Subscriber("/red/mavros/global_position/
                local", Odometry, self.cb_odom)
    rospy.Subscriber("/red/goal_pose", PoseStamped,
                self.cb_goal)

def cb_odom(self, odom_uav):
    odom_glocal = Odometry()
    tf_transfrom(tf_glocal, tf_uav)
    self.pub_odom.publish(odom_glocal)

def cb_goal(self, goal_glocal):
    goal_uav = PoseStamped()
```

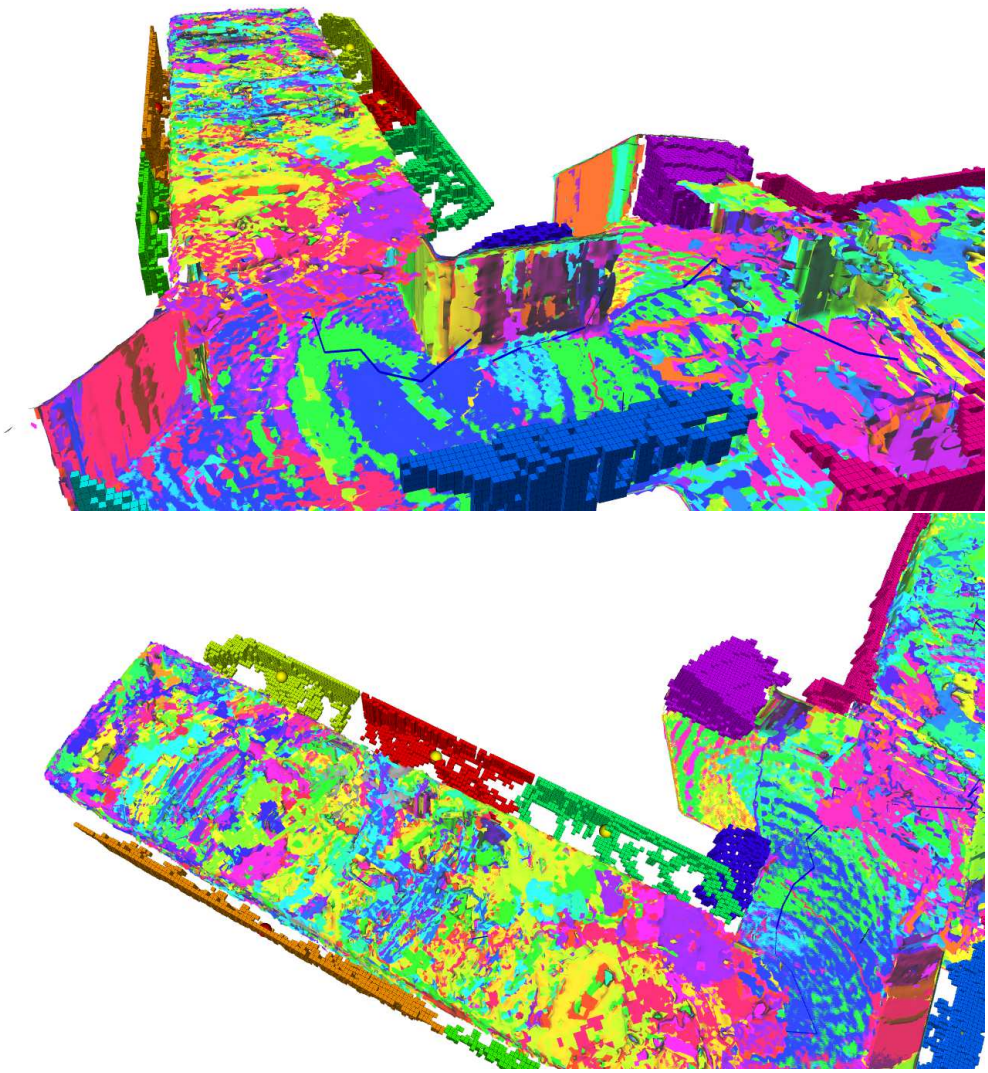
```
tf_transform(tf_glocal, tf_uav)
self.pub_goal.publish(goal_uav)
```

– **Poteškoće s izvođenjem programa:** pokretanje i pravilno izvođenje simulacije i sustava za istraživanje prostora ipak iziskuje malo snažnije računalne resurse. Zbog djelomičnih ograaničenja u tom segmentu uočeni su problemi s izvođenjem. Gazebo simulator bi se izvodio s faktorom stvarnog vremena manjim od 0.6 što bi uzrokovalo velike probleme sa samom simulacijom i kretnjom drona. Kontroler za letjelicu više ne bi funkcionirao pravilno te bi došlo do nepredvidivih kretnji te u konačnici nemogučnošću izvođenja misije. Uz dosta istraživanja i pronalaženja problema na kraju je donešen zaključak da je u ovom slučaju potrebno isključiti prikaz grafičkog sučelja Gazebo simulatora što se pokazalo uspješnim jer je nakon toga misija uspješno pokrenuta i izvođena. Navedeno se može jednostavno promijeniti u *launch* datoteci za simulator gdje se modificira *gui* parametar.

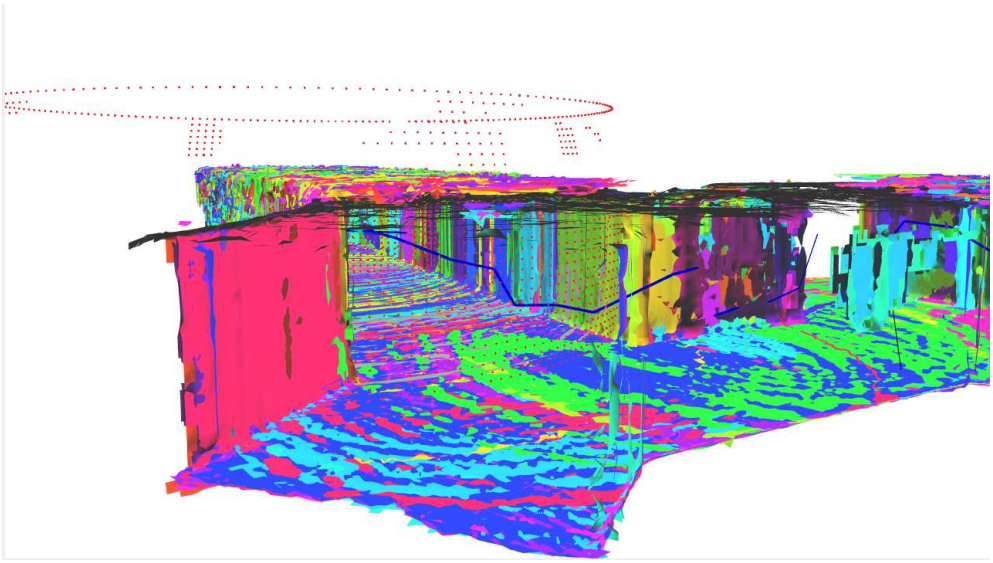
– **Pogrešna detekcija i pozicioniranje fronti:** nakon što je većina problema detektirano i uklonjeno izgledalo je kako sustav napokon normalno funkcionira. Međutim, u trenucima kada dolazi do promjene s lokalnog planera na globalni uočene su nepravilnosti. Program nas obavještava kako niti jedna od detektiranih fronti nije dohvatljiva odnosno nije moguće planiranje putanje do nje. Pogledom na RViz uočeno je kako su fronte u potpunosti krivo pozicionirane te se nalaze izvan ograđenog područja što nikako ne može biti moguće. Navedeno se može vidjeti na slici 4.14.

  Ovaj problem prouzročio je dosta razmišljanja i *debugiranja* da bi se na kraju ispostavilo kako postoji parametar zvan *use_missing_points_for_clearing* koji je postavljen u *true*, a pregledom koda utvrđeno je da on uključuje određene funkcionalnosti napravljene za AirSim simulator. To je utjecalo na pogrešnu reprodukciju laserskih zraka korištenih prilikom kartiranja kao što se vidi na slici 4.15. Može se uočiti kako dio laserskih zraka izlazi van zatvorenog područja modela okoline iako to ne bi smjelo biti moguće.

  Isključivanjem navedenomg parametra odnosno zadavanjem *false* vrijednosti problem je uklonjen te taj dio funkcionira normalno.



Slika 4.14: Pogrešno detektirane fronte

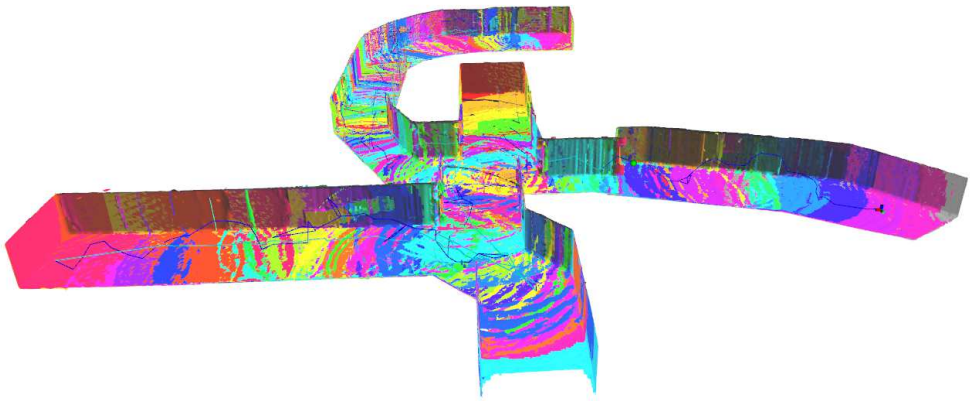


Slika 4.15: Greška prilikom reproduciranja laserskih zraka

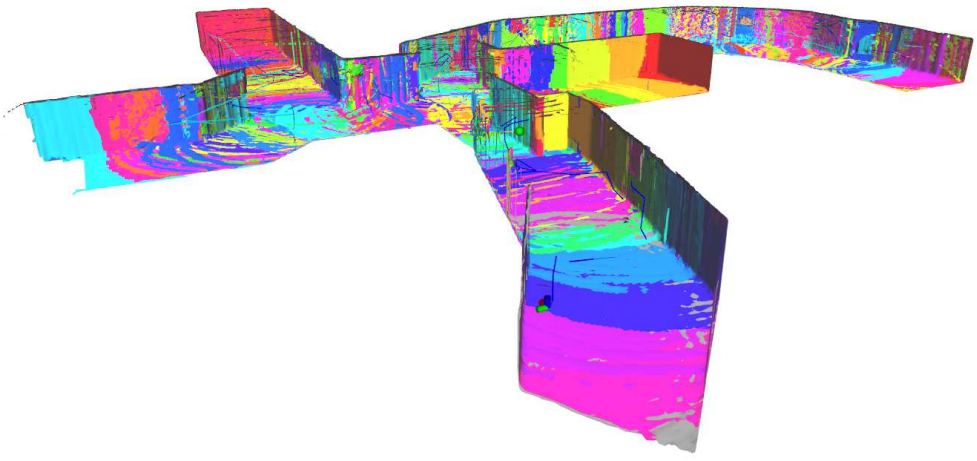
5. Eksperimentalni rezultati

Nakon što su napravljene sve promijene pokrenuta je simulacija i GLocal paket te su provedeni eksperimetni u tri slučaja. Dva u već spomenutim zatvorenim okruženjima te je još isproban slučaj u otvorenom okruženju. Eksperiment s otvorenim prostorom nije prošao uspješno već bi se letjelica previše zadržavala na određenom prostoru. Autori rada su svoje eksperimente proveli u zatvorenim prostorima te izložili rezultate u takvim slučajevima. Nije navedeno kako bi se sustav trebao ponašati u velikom otvorenom prostoru te koje bi potencijalno bile potrebne izmjene za bolje rezultate stoga su rezultati ovog eksperimenta zanemareni. U druga dva slučaja nije bilo većih problema pri izvedbi. Eksperimenti su provedeni na dva različita računala, prvo s grafičkom karticom NVIDIA GeForce GTX 1060 6GB i Intel i5-6500 3.20GHz četverojezgrenim procesorom te drugo, slabije, s grafičkom karticom NVIDIA GeForce MX150 2GB i Intel i5-8250U 1.6GHz procsorom. Na prvom računalu nije bilo nikakvih problema pri izvedbi te je simulacija bila izvođena faktorom realnog vremena približno 1. Uspješno je proveden eksperiment i za manje okruženje labirinta te ono veće u slučaju tunela. Izvedba sustava na drugom, slabijem, računalu imala je problema u slučaju s velikim okruženjem tunela gdje bi, nakon što je istražen velik dio mape, faktor realnog vremena izvođenja pao na manje od 0.55 što bi dovelo do korupcije prilikom kartiranja i izvođenja algoritma. Eksperiment s manjim okruženjem je prošao bez problema iako je i tu faktor realnog vremena izvođenja simulacije bio dosta manji od 1. Na slikama 5.1 i 5.2 prikazani su rezultati nakon završenog istraživanja velikog okruženja tunela. Na slikama se vidi ukupna karta izgrađena od podkarti dobivenih tijekom izvođenja sustava.

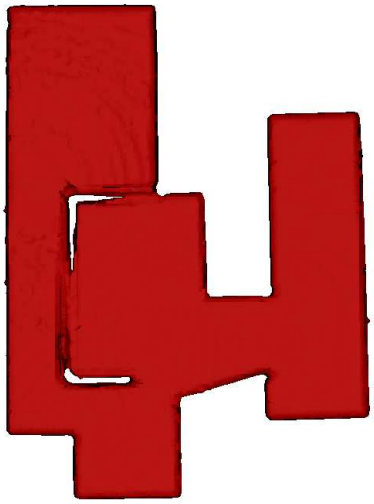
Također, prilikom trajanja izvedbe istraživanja postoji mogućnost spremanja trenutnog stanja karte u *ply* format koji se kasnije može otvoriti u nekom od programa koji podržavaju taj format kako bi vizualizirali dobiveni model. Na slikama 5.3, 5.3 i 5.5 može se vidjeti dobiveni model labirinta pri čemu je korišten *CloudCompare* programski alat. Na slici 5.6 vidi se prikaz tunela u RViz alatu poput onog prethodnog za tunele.



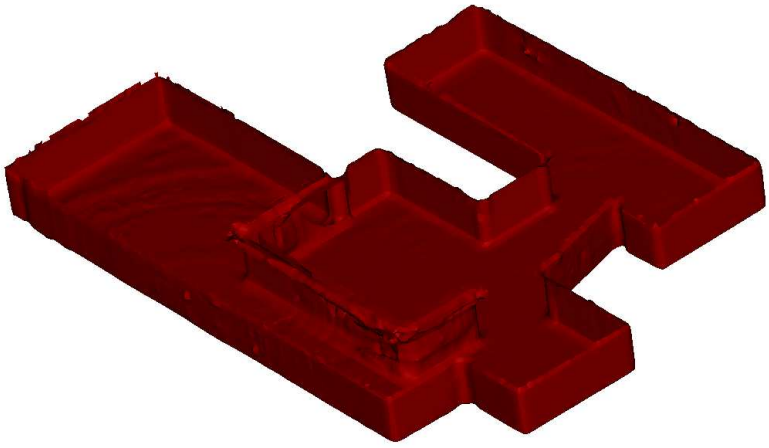
Slika 5.1: Rezultat istraživanja tunela



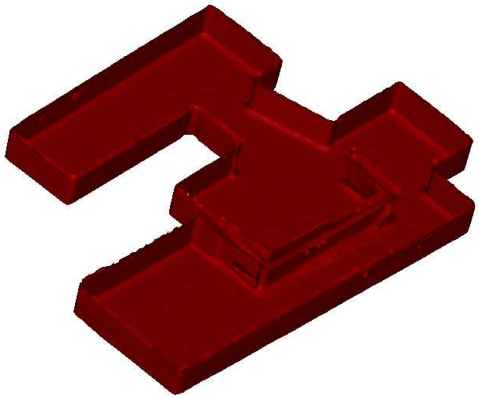
Slika 5.2: Rezultat istraživanja tunela



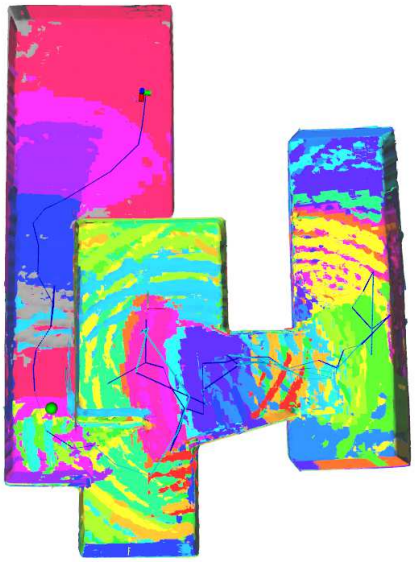
Slika 5.3: Dobiveni model labirinta



Slika 5.4: Dobiveni model labirinta



Slika 5.5: Dobiveni model labirinta



Slika 5.6: Rezultat istraživanja labirinta

6. Zaključak

Istraživanje i inspekcija prostora bespilotnom letjelicom predstavlja jednu od popularnijih vrsta upotreba dronova u današnjoj robotici. Kroz ovaj rad upoznali smo se s jednim od mnogih sustava za autonomno istraživanje prostora dronom, *GLocal*. Predstavljene su najvažnije komponente za kartiranje prostora te planiranje puta i objašnjene njihove uloge pri izvršavanju misije. Prezentirani su originalni rezultati te je dana usporedba s još dva različita sustava. Također, uz manje probleme, provedena je uspješna prilagodba navedenog sustava za rad na drugom simulacijskom okruženju te su navedene moguće poteškoće prilikom implementacije. Implementirane promjene su uspješno testirane u Gazebo simulacijskom okruženju u dva različita slučaja te su oba eksperimenta uspješno izvedena. Zbog nemogućnosti resursa, a i određenih ograničenja u simulaciji nije bilo moguće dobiti rezultate ukoliko postoji nakupljena grešku u odometriji koja bi uzrokovala pomak te se to može navesti kao jedno od mogućih daljnjih poboljšanja. Također, sljedeći veliki korak bio bi isprobati sustav za autonomno istraživanje na pravoj letjelici kako bi se dobili rezultati u realnim uvjetima te tako najbolje ocijenio rad sustava.

LITERATURA

[1] L. Schmid, V. Reijgwart, L. Ott, J. Nieto, R. Siegwart and C. Cadena, "A Unified Approach for Autonomous Volumetric Exploration of Large Scale Environments Under Severe Odometry Drift," in IEEE Robotics and Automation Letters, vol. 6, no. 3, pp. 4504-4511, July 2021, doi: 10.1109/LRA.2021.3068954.

[2] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart and J. Nieto, "An Efficient Sampling-Based Method for Online Informative Path Planning in Unknown Environments," in IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 1500-1507, April 2020, doi: 10.1109/LRA.2020.2969191.

[3] V. Reijgwart, A. Millane, H. Oleynikova, R. Siegwart, C. Cadena and J. Nieto, "Voxgraph: Globally Consistent, Volumetric Mapping Using Signed Distance Function Submaps," in IEEE Robotics and Automation Letters, vol. 5, no. 1, pp. 227-234, Jan. 2020, doi: 10.1109/LRA.2019.2953859.

[4] Helen Oleynikova, Zachary Taylor, Marius Fehr, Juan Nieto, and Roland Siegwart, "Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning", in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017.

[5] T. Dang, S. Khattak, F. Mascarich, and K. Alexis, "Explore locally, plan globally: A path planning framework for autonomous robotic exploration in subterranean environments," in Proc. 19th Int. Conf. Adv. Robot., 2019, pp. 9–16.

[6] B. Ho, P. Sodhi, P. Teixeira, M. Hsiao, T. Kusnur, and M. Kaess, "Virtual occupancy grid map for submap-based pose graph slam and planning in 3D environments," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2018, pp. 2175–2182.

[7] A. Millane, Z. Taylor, H. Oleynikova, J. Nieto, R. Siegwart, and C. Cadena, "C-blox: A scalable and consistent TSDF-based dense mapping approach," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Madrid, Spain, 2018, pp. 995–1002.

[8] ROS,
https://github.com/larics, pristupljeno 15. siječnja 2024.

[9] RViz,
https://www.ros.org/, pristupljeno 15. siječnja 2024.

[10] RViz,
https://gazebosim.org/, pristupljeno 15. siječnja 2024.

[11] LARICS GitHub,
http://wiki.ros.org/rviz, pristupljeno 15. siječnja 2024.

[12] ETHZ GitHub,
https://github.com/ethz-asl, pristupljeno 15. siječnja 2024.

[13] UAV inspekcija u agronomiji,
https://pixabay.com/photos/dji-drone-plant-protection-drone-4204801/, pristupljeno 15. siječnja 2024.

[14] AirSim simulator,
https://microsoft.github.io/AirSim, pristupljeno 15. siječnja 2024.

[15] Gazebo simulator,
https://automaticaddison.com/useful-world-files-for-gazebo-and-ros-2-simulations/, pristupljeno 15. siječnja 2024.

[16] Greška odometrije,
https://rpg.ifi.uzh.ch/docs/ICRA18_Delmerico.pdf, pristupljeno 15. siječnja 2024.

[17] Kopterworx UAV,
https://www.kopterworx.com/, pristupljeno 15. siječnja 2024.

Autonomno istraživanje 3D prostora uz kompenzaciju greške nastale optimizacijom trajektorije

Sažetak

Cilj ovog rada je opisati *GLocal*, modularni sustav za efikasno autonomno istraživanje prostora većih razmjera bespilotnom letjelicom. Pristupom temeljenim na izgradnji podkarti kombiniraju se višestruki slojevi kartiranja i planiranja kako bi se postigla otpornost na odometrijsku grešku u pomaku uz zadržavanje efikasnosti izvođenja misije. Praktični dio zadatka uključuje razvoj i implementaciju promjena kako bi se umjesto AirSim simulatora baziranog na Unreal Engineu moglo koristiti Gazebo simulacijsko okruženje zajedno s robotskim operacijskim sustavom (ROS). U radu su objašnjeni parametri korišteni u simulaciji te su navedene upute i promjene koje je potrebno napraviti za ispravno korištenje. Implementirane promjene su testirane u eksperimentima provedenim u dva različita simulacijska svijeta, tuneli i labirint.

Ključne riječi: dron, UAV, robot, istraživanje, simulacija, navigacija, kartiranje, ROS

Autonomous exploration of 3D environment with trajectory optimization error compensation

Abstract

The goal of this paper is to describe GLocal, a modular system for autonomous exploration of large scale environments with an unmanned areal vehicle (UAV). The submap-based approach combines multiple layers of mapping and planning to achieve robustness to odometry drift while maintaining mission efficiency. The practical part of the task involves the development and implementation of changes to use the Gazebo simulation environment together with the Robotic Operating System (ROS) instead of AirSim simulator based on the Unreal Engine. The paper explains the parameters used in the simulation, as well as the instructions and changes that need to be made for correct use. The implemented changes were tested in experiments conducted in two different simulation worlds, tunnels and a maze.

Keywords: drone, UAV, robot, exploration, simulation, navigation, mapping, ROS