Kartiranje vanjskih prostora u stvarnom vremenu s primjenama u poljoprivredi

Prpić, Marco Davide

Master's thesis / Diplomski rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka, Faculty of Engineering / Sveučilište u Rijeci, Tehnički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:190:291052

Rights / Prava: Attribution 4.0 International/Imenovanje 4.0 međunarodna

Download date / Datum preuzimanja: **2024-05-04**

Repository / Repozitorij:

Repository of the University of Rijeka, Faculty of Engineering

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Diplomski sveučilišni studij računarstva

Diplomski rad

KARTIRANJE VANJSKIH PROSTORA U STVARNOM VREMENU S PRIMJENAMA U POLJOPRIVREDI

Rijeka, srpanj 2022.

Marco Davide Prpić
0069078573

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Diplomski sveučilišni studij računarstva

Diplomski rad

KARTIRANJE VANJSKIH PROSTORA U STVARNOM VREMENU S PRIMJENAMA U POLJOPRIVREDI

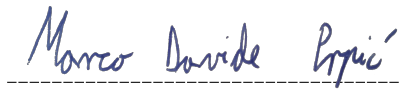
Mentor: Prof. dr. sc. Kristijan Lenac

Rijeka, srpanj 2022.

Marco Davide Prpić
0069078573

IZJAVA

Sukladno članku 43. Statuta Tehničkog fakulteta Sveučilišta u Rijeci o diplomskom radu, diplomskom ispitu i završetku diplomskih sveučilišnih studija, izjavljujem da sam samostalno, pod vodstvom Prof. dr. sc. Kristijana Lenca, primjenjujući znanja stečena tijekom studija, te koristeći navedenu literaturu izradio diplomski rad naslova "Kartiranje vanjskih prostora u stvarnom vremenu s primjenama u poljoprivredi".



Marco Davide Prpić

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 12. ožujka 2021.

Zavod: **Zavod za računarstvo**
Predmet: **Mobilna robotika**
Polje: **2.09 Računarstvo**

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Marco Davide Prpić (0069078573)**
Studij: Diplomski sveučilišni studij računarstva
Modul: Programsko inženjerstvo

Zadatak: **Kartiranje vanjskih prostora u stvarnom vremenu s primjenama u poljoprivredi / Real-time outdoor mapping with applications in agriculture**

Opis zadatka:

Istražiti postojeća rješenja za trodimenzionalno kartiranje vanjskih prostora u stvarnom vremenu u ROS (Robot Operating System) okruženju upotrebom Intel RealSense D455 kamere. Predložiti i testirati konfiguraciju sustava prilagođenu za izradu preciznih karata poljoprivrednog zemljišta.

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

Marco Davide Prpić

Zadatak uručen pristupniku: 15. ožujka 2021.

Mentor:

Izv. prof. dr. sc. Kristijan Lenac

Predsjednik povjerenstva za diplomski ispit:

Izv. prof. dr. sc. Kristijan Lenac

Sadržaj

1 UVOD **1**

2 KARTIRANJE VANJSKIH PROSTORA U STVARNOM VREMENU **2**
2.1 Pristupi kartiranju 2
2.2 Pristupi rješavanja SLAM problema 4
2.2.1 Pristupi temeljeni na korištenju filtera 4
2.2.2 Pristupi temeljeni na optimizaciji grafa 4
2.3 Podjela vizualne odometrije u odnosu na pogrešku koju minimizira 5
2.4 Vizualna odometrija temeljena na značajkama u slici 6
2.4.1 Vizualne značajke - Ključne točke 6
2.4.2 Vizualne značajke - Deskriptori 8
2.4.3 Estimacija pomaka kamere 10
2.5 Zatvaranje petlje i optimizacija grafa 12
2.5.1 Zatvaranje petlje 12
2.5.2 Optimizacija grafa 13

3 ROBOTSKI OPERACIJSKI SUSTAV(ROS) **15**

4 INTEL REALSENSE KAMERA **21**

5 ISTRAŽENA RJEŠENJA ZA KARTIRANJE U STVARNOM VREMENU OTVORENOG KODA **24**

6 PREDLOŽENA KONFIGURACIJA SUSTAVA ZA KARTIRANJE POLJOPRIVREDNOG ZEMLJIŠTA **31**
6.1 Instalacija programske podrške 31
6.2 Priprema i konfiguracija Intel Realsense D455 kamere 32
6.3 Konfiguracija inercijalne mjerne jedinice 34
6.4 Intel RealSense kamera i RTABMAP 36
6.4.1 Pokretanje RTABMAP čvorova 37
6.4.2 Dodatni senzori 38
6.5 Konfiguracija RTABMAP čvorova 38
6.6 Konfiguracija čvora za estimaciju položaja 40

7 TESTIRANJE I REZULTATI **43**

8 ZAKLJUČAK **51**

Literatura **52**

Popis slika **54**

Popis tablica **55**

9 SAŽETAK I KLJUČNE RIJEČI **56**

10 Dodatak A **57**

1 UVOD

Potreba za autonomnim strojevima u poljoprivrednom području sve je veća jer poljoprivrednici sve više prepoznaju njezin utjecaj u poljoprivredi. Roboti se sada koriste za različite zadatke kao što su sadnja, berba, praćenje okoliša, opskrba vodom i hranjivim tvarima itd. U tom kontekstu, nužno je razviti rješenja koja robotima omogućuju autonomno kretanje u poljoprivrednim područjima. Jedan od zadataka kojeg je potrebno riješiti je omogućiti robotu lokalizaciju u tim područjima. Najčešći pristup je korištenje samostalnih rješenja globalnog navigacijskog satelitskog sustava(GNSS[1]) [1]. Naime, sama lokalizacija daje odgovor na pitanje gdje se robot nalazi, a ne što se oko robota nalazi. Drugi bitan zadatak je omogućiti robotima predodžbu terena na kojem rade. Različite kulture biljaka zahtijevaju različite potrebe, te je prepoznavanje različitih kultura velika potreba autonomnih robota koji rade na poljoprivrednim terenima. Uz prepoznavanje različitih kultura korisno je i praćenje razvoja istih. Kako bi robotima omogućili predodžbu prostora oko njih potrebno je kreirati vizualnu kartu poljoprivrednog terena na kojem rade. Postupak kreiranja karte naziva se kartiranje.

Istovremena lokalizacija i kartiranje (Simultaneous Localization and Mapping (SLAM)) je tehnika koja se sastoji od procijene položaja robota pomoću ulaznih senzorskih podataka, pri čemu se istovremeno gradi karta prostora oko robota. Karta je višedimenzionalni prikaz koji se dobiva analizom podataka ugrađenih robotičkih senzora, a koji se, uz informiranje o okolini, koristi kao skup referenca u postupku lokalizacije. Kada se za rješavanje SLAM problema koristi kamera kao senzor onda se taj problem naziva vizualni SLAM (VSLAM). Sami postupak estimacije položaja kamere uz korištenje slika se još naziva i vizualna odometrija (VO). VO je potproblem VSLAM-a koji daje lokalnu estimaciju položaja kamere. Zadatak VSLAM-a je osigurati globalnu estimaciju položaja kamere, kao i graditi kartu prostora.

Nepravilnost terena dovodi do nestabilnijeg kretanja u poljoprivrednim terenima. Različita osvijetljenost terena kroz dan i iz različitih kuteva pogleda stvara veliki problem VSLAM-u koji se oslanja samo na sliku za estimaciju položaja. Također VO radi greške u procijeni položaja kamere. VO se može i izgubiti na mjestima gdje u slikama nema dovoljno različitih vizualnih značajki za procjenu pokreta, a poljoprivredni tereni su jednoličnog uzorka. Svi navedeni razlozi ukazuju na problematiku korištenja vizualnog pristupa u poljoprivredi. Stoga se ne može direktno osloniti na opremu i algoritme za kartiranje kako bi riješili problem, već treba postaviti problem tako da se eliminiraju neki od spomenutih vanjskih čimbenika. Na primjer, obavljati kartiranje kada imamo jednoliko osvjetljenje terena, postavljanje značajki po terenu, kako se vizualna odometrija ne bi gubila itd.

U sljedećem poglavlju započeti će se priča općenito o kartiranju vanjskih prostora, te usporediti dva pristupa: robotički pristup u stvarnom vremenu i fotogrametrijski pristup koji se ne izvodi u stvarnom vremenu. U istom će se poglavlju zatim obratiti pažnja na robotički pristup u stvarnom vremenu(VSLAM), napraviti podjela takvih pristupa u odnosu na pogrešku koju minimiziraju, te opisati korake koji se koriste u mnogim rješenjima VSLAM-a. U trećem poglavlju će se pričati o ROS-u, usporediti ROS sa ROS2 verzijama. Zatim u četvrtom poglalju će se pričati o Intel RealSense serijama kamera. U petom poglalju pričati će se o VSLAM rješenjima otvorenog koda dostupnima u ROS okruženju. U šestom poglavlju će se predstaviti postupak rada, koji uključuje instalaciju i pokretanje čvorova RTABMAP-a s Intel RealSense D455 kamerom u ROS2 okruženju. U sedmom poglavlju će biti opisano testiranje i rezultati te u konačnici zaključak.

[1]Globalni navigacijski satelitski sustav (GNSS) odnosi se na konstelaciju satelita koji daju signale iz svemira koji prenose podatke o položaju i vremenu do GNSS prijamnika. Primatelji zatim koriste te podatke za određivanje lokacije. Po definiciji, GNSS osigurava globalnu pokrivenost. Primjeri GNSS-a uključuju europski Galileo, američki NAVSTAR Global Positioning System (GPS), ruski Global'naya Navigatsionnaya Sputnikovaya Sistema (GLONASS) i kineski BeiDou Navigation Satellite System.

2 KARTIRANJE VANJSKIH PROSTORA U STVARNOM VREMENU

Kartiranje je proces dobivanja karte. Kao što je navedeno u uvodu karte prostora su potreba u autonomnim robotičkim aplikacijama u poljoprivredi, radi prepoznavanja biljaka i praćenja razvoja istih. Razne se karte koriste u robotici. U odnosu na dimenzionalnost imamo 2D i 3D karte, u odnosu na informaciju koju daju imamo metričke, topološke itd. Svaka karta ima svoje prednosti. Recimo topološka karta ne daje informaciju o prostornoj udaljenosti između dva čvora, već informaciju jesu li dva čvora povezana ili ne. Informacija o povezanosti dvaju čvorova može pomoći u rješavanju problema planiranja putanje robota. Metričke karte daju geometrijsku, prostornu predodžbu prostora, te se može prikazati u koordinatnom sustavu. Informacija o geometrijskoj predodžbi može pomoći u problemu lokalizacije robota. Kako postoji više vrsta karti, potrebno je sagledati koji problem želimo moći riješiti kartom te se odlučiti za onu koja rješava naš problem.

2.1 Pristupi kartiranju

Kako bi se kreirala karta, u robotici se koriste senzori, kao što su:

- Kamere (3D vizualno kartiranje)
- Laseri (2D/3D geometrijska predodžba prostora)
- Sonari (3D geometrijska predodžba prostora (npr. podvodno kartiranje))

Ovaj rad se fokusira na kartiranju sa kamerama, te će se u nastavku isključivo o tome pričati. Vizualno 3D kartiranje se može koristiti za izradu karti, modela objekata, te se naziva još i 3D rekonstrukcija. Ovim se problemom ne bavi samo robotika, već i druge grane kao fotogrametrija. Način kreiranja karte prostora dok se istovremeno pokušava lokalizirati robot u karti koju kreira naziva se SLAM. SLAM je problem kojeg robotika pokušava riješiti. Ako pogledamo kreiranje karte iz slika, kao glavni cilj onda i fotogrametrija ima svoje rješenje a naziva se struktura iz pokreta (eng. Structure from motion (SfM)). Oba rješenja imaju zadatak kreirati kartu iz slika, no nemaju oba iste ciljeve koje žele zadovoljiti.

Značajke SLAM-a:

- Cilj je procijeniti položaj senzora i kartu (slično 3D rekonstrukciji).
- Glavni senzori mogu biti kamera, laserski radari (LIDAR - Light Detection and Ranging), zvučni radari (sonar - Sound Detection and Ranging), ... i mogu imati dodatne senzore kao što su GNSS, inercijalna mjerna jedinica(IMU - Inertial Measurement Unit), ...
- Općenito, fokus je na stvarnom vremenu, ali se može koristiti i izvanmrežno(offline).
- Fokus je na kontinuitetu.
- Može se riješiti Kalmanovim filtrom ili pristupom prilagodbe paketa (eng. Bundle Adjustment (optimizacijski pristup)).
- Podaci gotovo uvijek dolaze inkrementalno.
- Dolazi iz zajednice robotike.
- Može biti i 2D problem.

Značajke SfM-a:

- Cilj je rekonstruirati 3D okruženje iz serije 2D slika. Struktura se odnosi na 3D okruženje, a gibanje dolazi iz činjenice da 2D slike moraju biti snimljene s različitih pozicija.
- Glavni senzor je uvijek kamera, a mogu se koristiti i drugi senzori kao što su IMU i GNSS.
- Problem se obično smatra izvanmrežnim postupkom, što znači da rješavate problem nakon što prikupite sve podatke.
- Fokus je na točnosti 3D rekonstrukcije.
- Uobičajeni način rješavanja ovog problema je prilagodba paketa.
- Kako su svi podaci dostupni, mogu se obraditi bilo kojim redoslijedom.
- Dolazi iz zajednice računalnog vida.

Ako se fokusiramo isključivo na VSLAM, onda nam je glavni senzor kamera. Također problem je 3D prirode. Može se vidjeti velika povezanost između VSLAM-a koji se rješava postupkom prilagodbe paketa i SfM-a. Njihova je razlika što VSLAM pokušava problem riješiti u stvarnom vremenu, inkrementalno, dok SfM prikupi podatke prije, te uzme vremena za procesiranje izvanmrežno. VSLAM brine o tome da se izvodi u stvarnom vremenu što vodi do toga da gubi na kvaliteti procesiranja slike, što se odnosi i na rezultantnu kartu. SfM se fokusira samo na kvalitetu izlazne karte i nije mu bitno vrijeme procesiranja, jer se izvodi izvanmrežno.

Kako je i SfM pristup često korišten u kartiranju poljoprivrednih površina i drugih vanjskih okruženja, ima mjesto u ovom radu iako se ne izvodi u stvarnom vremenu. Danas ima mnoštvo fotogrametrijskih rješenja dostupnih. Neki od njih su besplatni kao Meshroom[2], OpenDroneMap[3], COLMAP[4], te komercijalni kao Pix4D Mapper[5], DroneDeploy[6], 3DFlow [7].

Ako je cilj precizna i velika karta SfM je bolji izbor, no ako je cilj izvođenje u stvarnom vremenu onda je VSLAM rješenje. **U nastavku rada fokus će biti na VSLAM-u**, jer se od rada traži izvođenje u stvarnom vremenu. U sljedećem poglavlju će se dati opis koraka kojeg imaju VSLAM rješenja. Mnogi od tih koraka se također primjenjuju u SfM rješenjima također.

[2]Meshroom - `https://alicevision.org/#meshroom`
[3]OpenDroneMap - `https://www.opendronemap.org/`
[4]COLMAP - `https://colmap.github.io/index.html`
[5]Pix4D Mapper - `https://www.pix4d.com/product/pix4dmapper-photogrammetry-software`
[6]DroneDeploy - `https://www.dronedeploy.com/`
[7]3DFlow - `https://www.3dflow.net/`

2.2 Pristupi rješavanja SLAM problema

Mnogo je pristupa za rješavanje SLAM problema, od kojih se većina može kategorizirati u dvije glavne paradigme: pristupi temeljeni na filterima (Filter based) i pristupi temeljeni na optimizaciji grafa (Optimization (Graph) based).[2]

2.2.1 Pristupi temeljeni na korištenju filtera

Najpopularniji pristupi temeljeni na korištenju filtera su filteri bazirani na Kalmanovom filteru i filtru čestica. Oni bazirani na Kalmanovom filtru su KF (Kalman Filter), EKF(Extended Kalman Filter), UKF(Unscented Kalman Filter), EIF (Extended Information Filter), SEIF (Sparse Extended Information Filter). Svi ovi pristupi se baziraju na Gaussovu distribuciju za estimaciju položaja u svojoj pozadini.[3]

Filter čestica je metoda u kojoj se skup čestica koristi za predstavljanje vjerovanja robota, umjesto da se koriste parametarske vrijednosti (tj. srednja vrijednost i varijanca) za opisivanje distribucije kao što je Kalman filtar i prošireni Kalmanov filtar.[4]

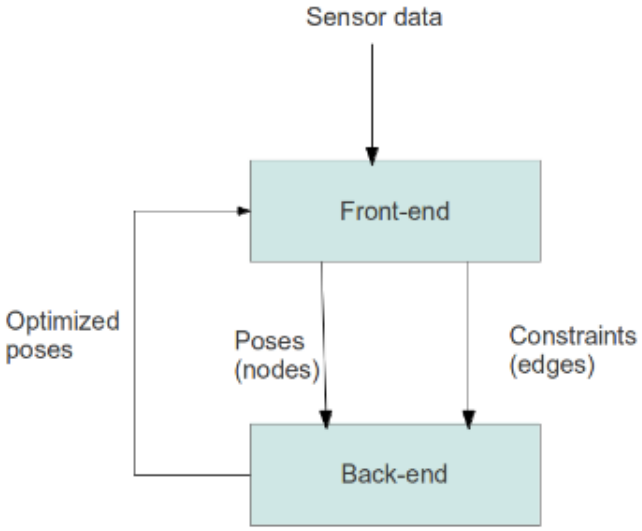
Pristupi bazirani na filterima spajaju mjerenja senzora uz uzastopno ažuriranje distribucije vjerojatnosti na osnovu interpretacije mjerenja iz senzora.[5] Korišteni su puno u prošlosti, a koriste se i dan danas. Njihova prednost je što jako dobro modeliraju greške iz nesavršenih senzora. Većina ovih pristupa nije prilagođena za SLAM velikih razmjera, kao što je VSLAM. Danas se sve više pridodaje pažnji pristupima temeljenim na optimizaciji grafa[6], zbog njihovog boljeg izvođenja u problemima velikih razmjera.[5] **U daljnjem radu će se više pažnje obratiti na optimizacijski pristup rješavanja SLAM problema**.

2.2.2 Pristupi temeljeni na optimizaciji grafa

Pristup temeljen na optimizaciji (Grafni pristup) obično koristi temeljnu strukturu grafa za predstavljanje mjerenja robota. Čvorovi grafa predstavljaju poze robota i mjerenje dobiveno na ovoj poziciji, a veze koje povezuju čvorove predstavljaju prostorno ograničenje koje se odnosi na dvije poze robota. Ograničenje se obično sastoji od relativnih transformacija između dvije poze. Ove transformacije su ili mjerenja odometrije između uzastopnih položaja robota ili se određuju usklađivanjem opažanja dobivenih na dvije lokacije robota. Nakon što je graf konstruiran, proces optimizacije počinje pronalaziti konfiguraciju poza robota koja najbolje zadovoljava ograničenja[2]. SLAM temeljen na optimizaciji se sastoji od 2 najbitnija zadatka:

- **Konstrukcija grafa**
  - Za novu pozu robota kreiraj novi čvor u grafu, poveži čvorove robota ograničenjima dobivenim estimacijom pomaka. Na taj se način kreira graf s pozama i ograničenjima između svake poze. U ovaj proces dolaze i ograničenja nastala zatvaranjem petlje, više o ovome kasnije. Konstrukcija grafa se odvija u *frontend* dijelu VSLAM-a, vidi sliku Slika 1.

- **Optimizacija grafa**
  - Optimizacija grafa se odnosi na optimiziranje kreiranog grafa u prethodnom koraku, tako da greška u pozama bude minimalna. Na ovaj se način ispravljaju položaji kamere kao i estimirane 3D točke u prostoru. Optimizacija grafa se odvija u *backend* dijelu VSLAM-a, vidi sliku Slika 1.



Slika 1: Shema optimizacijskog pristupa: *frontend* i *backend*

IZVOR: Doaa M. A.-Latif, Mohammed A-Megeed Salem, H. Ramadan and Mohamed I. Roushdy; 2014. Volume 8

2.3 Podjela vizualne odometrije u odnosu na pogrešku koju minimizira

Vizualna odometrija (VO) je potproblem VSLAM-a. VO želi estimirati položaj kamere koristeći samo slike kao ulaz. Većina gotovih rješenja za problem VO se mogu kategorizirati u sljedeće kategorije i to u odnosu na pogrešku koju minimiziraju:

- **Pristup temeljen na praćenju značajki iz slike**
  - uključuje izdvajanje značajki slike (kao što su kutovi, linije i mrlje[8]) između uzastopnih okvira slike, podudaranje ili praćenje izdvojenih značajki među slikama i konačno procjenu pokreta kamere na osnovu podudaranja. Ovaj pristup se temelji na minimiziranju pogreške geometrijske reprojekcije.

- **Direktni pristup**
  - Direktni pristup je započeo s idejom korištenja svih piksela u slici za rješavanje svijeta oko senzora, oslanjajući se na principe fotogrametrije. Umjesto izdvajanja značajki iz slike i praćenja tih značajki u 3D prostoru, izravne metode gledaju na neke ograničene aspekte piksela (boja, svjetlina, gradijent intenziteta) i prate kretanje tih piksela od okvira do okvira. Ovaj pristup mijenja problem koji se rješava od minimiziranja pogrešaka geometrijske reprojekcije, kao u slučaju pristupa s izvlačenjem značajki, na minimiziranje fotometrijskih pogrešaka.[7]

- **Hibridni pristup** (spoj prethodna dva)
  - Pristup temeljen na značajkama ima prednost u visoko teksturnim scenarijima, dok opada u područjima jednog uzorka. U takvim scenarijima bolje radi direktni pristup. Hibridni pristup je korištenje jednog i drugog pristupa u kombinaciji.[8]

Ovaj će se rad fokusirati na kategoriju koja koristi praćenje značajki u slikama kao metodu za procjenu vizualne odometrije. Mnogi algoritmi koriste ovaj pristup vizualne odometrije, a u nastavku slijedi detaljniji opis takvog pristupa.

[8]mrlje(eng. blobs) - regije na digitalnoj slici koje se razlikuju po svojstvima, kao što su svjetlina ili boja, u usporedbi s okolnim regijama.

2.4 Vizualna odometrija temeljena na značajkama u slici

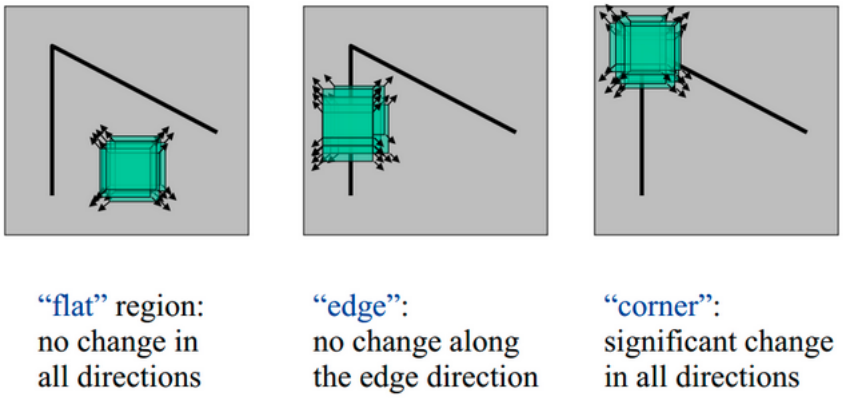
U ovom poglavlju obratit će se pažnja na dobivanje odometrije korištenjem slika i to s idejom izdvajanja značajki iz slike. U nastavku će biti opisani koraci za dobivanje odometrije.

2.4.1 Vizualne značajke - Ključne točke

Vizualne značajke su specifične točke u slici koje odstupaju na neki način od drugih, bilo da su rubne točke, kutevi ili mrlje. Želimo pronaći takve specifične točke u slici kako bismo ih mogli pratiti u drugim slikama. Vizualna značajka se sastoji od dva dijela, to su ključna točka(keypoint), i deskriptor ključne točke(descriptor).

$$Vizualna\ značajka = Ključna\ točka + Deskriptor$$

Ključna točka je lokacija specifične točke u slici, dok je deskriptor opis lokalne strukture oko ključne točke. U ovom će se poglavlju razgovarati o ključnim točkama i kako se pronalaze u slici.



Slika 2: Detekcija ključnih točaka: jednolični prostor, rub, kut

IZVOR: C. Harris, M. Stephens; 1988.

Kao što je do sada navedeno točke od interesa su one koje predstavljaju neki kut ili rub. Kutevi daju bolju predodžbu od rubova(stranica), pošto ih se može detektirati u raznim pomacima, dok rubovi pate od pomaka u smjeru ruba. Kako bi se pronašli kutevi mora se napraviti pretraga oko točke i pronaći promjena u intenzitetima piksela u više smjerova, vidi sliku Slika 2.

Detektori kuta koji koriste strukturnu matricu

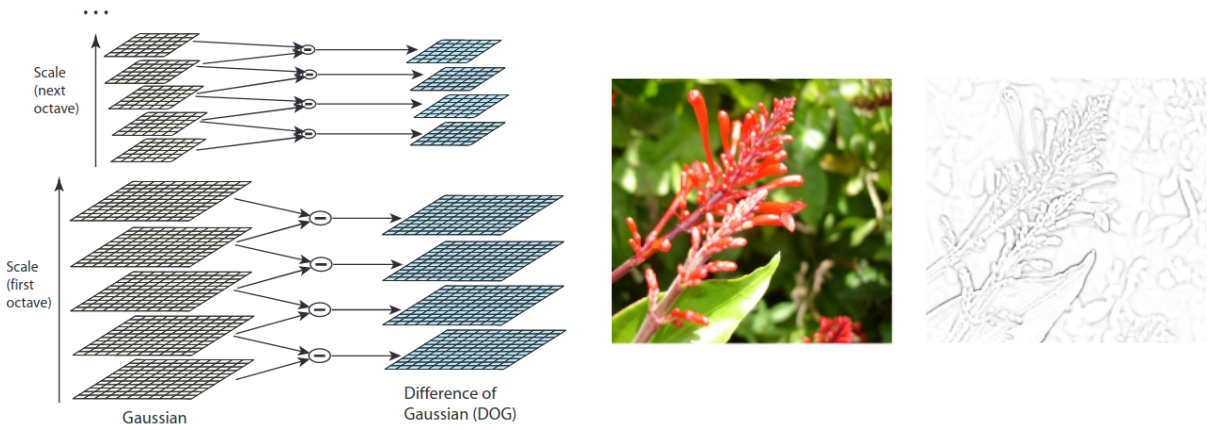
Strukturna matrica se koristi za pronalaženje rubova i kutova. Ona kodira promjene u intenzitetu slike u lokalnom području oko ključne točke. Neki od detektora kuteva koji se i danas koriste a na bazi su strukturne matrice su:

- Harris corner detector (1988)
- Shi Tomasi corner detector (GFTT - Good Features To Track)(1994)

Shi Tomasi je danas često korišten detektor kuteva. U raznim knjižnicama, kao što je OpenCV stoji kao zadani detektor kuteva.

Difference of Gaussians(DoG)

Difference of Gaussians(DoG) tehnika ne koristi strukturnu matricu. Ideja ove tehnike je uzeti sliku, napraviti konvoluciju s različitim Gaussovim kernelima kako bi se dobile slike različite zamućenosti, te pratiti točke u lokalnoj slici i između slika različite zamućenosti, vidi sliku Slika 3(lijevo) Oduzimanjem slika različite zamućenosti jedna od druge povećava vidljivost kutova, rubova i drugih detalja prisutnih na slici(mrlja), vidi sliku Slika 3(desno).

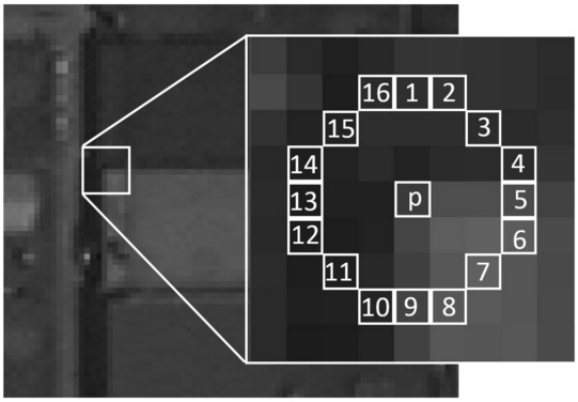


Slika 3: Metoda DoG: Shematski prikaz oduzimanja slika različite zamučenosti(lijevo). Rezultat oduzimanja(desno)

IZVOR: David G. Lowe, 2004. ; SadaraX(Wikipedija), 2008.

Features from accelerated segment test(FAST)

FAST kutni detektor izvorno su razvili Edward Rosten i Tom Drummond, a objavljen je 2006. godine. Prednost FAST-a je njegova brzina. FAST detektor kuta koristi krug od 16 piksela (Bresenhamov krug radijusa 3) za klasifikaciju je li kandidat točka p zapravo kut, vidi sliku Slika 4. Vrijednost svjetline svakog piksela u ovom krugu uspoređuje se sa središnjim pikselom. Područje je definirano kao ujednačena, rub ili kut na temelju postotka susjednih piksela sa sličnim intenzitetima kao središnji piksel. Poznato je da je FAST jedna od računski najučinkovitijih metoda ekstrakcije značajki. Međutim, vrlo je osjetljiv na šum. [9]



Slika 4: FAST detektor kuteva

IZVOR: Jingjin Huang, Guoqing Zhou, Xiang Zhou and Rongting Zhang, 2018.

2.4.2 Vizualne značajke - Deskriptori

U prethodnom potpoglavlju pričalo se o tehnikama pronalaženja ključnih točaka u slici. U ovo poglavlju će se govoriti na koji način se može opisati lokalna struktura oko značajne točke u slici, te na kraju kako će se moći uspoređivati na osnovu tog opisa. Deskriptor je neka numerička reprezentacija ključne točke, tj. njene okoline. Danas postoji mnoštvo deskriptora značajki samo neki od njih su:

- SIFT: Scale Invariant Feature Transform
- SURF: Speeded-Up Robust Features
- BRIEF: Binary Robust Independent Elementary Features
- ORB: Oriented FAST and rotated BRIEF

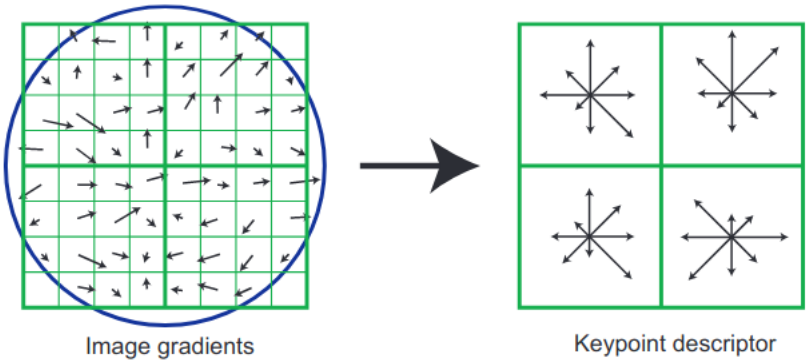
SIFT/SURF

SIFT je deskriptor značajki koji numerički opisuje ključnu točku i njenu okolinu. Ovaj deskriptor koristi DoG detektor za pronalaženje ključnih točaka i koristi neke informacije iz tog postupka kako bi opisao ključnu točku. SIFT deskriptor se sastoji od 4 dijela a to su:

- p - lokacija piksela u slici
- s - scale(na kojoj smo skali u DoG tehnici dobili najveću vrijednost)
- r - orijentacija(izračunaju se gradijenti u okolini slike i onaj smjer koji prevladava on određuje orijentaciju)
- f - 128-brojni deskriptor(opisuje gradijente oko ključne točke)

Koraci dobivanja 128-bojnog deskriptora su:

1. Uzmi lokaciju točke dobivenu DoG tehnikom za pronalazak ključne točke
2. Uzmi područje oko točke i izračunaj gradijente
3. Podijeli područje na 16 dijelova, svaki gradijent može imati 8 smjerova (16x8=128 vrijednosti gradijenata), vidi sliku Slika 5



Slika 5: SIFT: gradijenti

IZVOR: David G. Lowe, 2004.

SIFT metoda opisuje značajke tako da su otporne na translaciju, rotaciju i skaliranje. Također su djelomično otporne na razliku u osvjetljenju. SIFT deskriptor je pogodan za prepoznavanje značajki na različitim udaljenostima i iz različitih kuteva, te do neke razine i za slike s različitim osvjetljenjem. Dva SIFT deskriptora se uspoređuju računanjem *euklidske* udaljenosti.

SURF je tehnika nastala djelomično po uzoru na SIFT. SURF nadmašuje SIFT u smislu vremena i računalne učinkovitosti. Međutim, SIFT neznatno nadmašuje SURF u pogledu robusnosti.[9]

BRIEF/ORB

Kompleksne značajke koje dobivamo od SIFT-a rade super i do današnjeg dana su ostale zlatni standard u fotogrametriji, no najveći njihov problem je brzina izvođenja. Postoje izvedbe SIFT-a koje se izvode na grafičkoj kartici kako bi se ubrzalo računanje, no to nije rješenje za sustave koji ne raspolažu s velikom računskom snagom. Tu nastaju binarni deskriptori kojima je glavni cilj pojednostaviti značajke kako bi se one brže računale i uspoređivale.

Ideja iza binarnih deskriptora je trivijalna:

1. Uzmi neko područje oko ključne točke
2. Odaberi set pixela u odabranom području
3. Usporedi intenzitete odabranog para

$$b = \begin{cases} 1 & \text{if } I(s_1) < I(s_2) \\ 0 & \text{otherwise} \end{cases}$$

4. Ponavljaj 2. i 3. a rješenje b dodaji jedan iza drugoga u string

Ovisno koliko parova pixela se želi usporediti toliko će biti dužina binarnog stringa. Jako je bitno da jednom kad se definira strategija redoslijeda usporedbe parova piksela, da se ona ne mijenja. U protivnom usporedba ne bi bila relevantna. Razlike među različitim binarnim deskriptorima najčešće leži u strategiji odabira parova piksela za usporedbu. Usporedba dvaju deskriptora je sada također jednostavna i brza a to je izračun *Hamming*-ove udaljenosti:

$$d_{Hamming}(B_1, B_2) = sum(xor(B_1, B_2))$$

BRIEF je prvi binarni deskriptor predložen 2010. godine. Koristi 256 parova piksela za usporedbu. Izrazito je brz u računanju deskriptora za ključne točke, kao i za usporedbu dvaju deskriptora. Puno je brži od SIFT-a, no SIFT je robusniji. Naime SIFT je otporan na rotacije i skaliranje, dok to nije slučaj za BRIEF.

ORB(Oriented FAST and rotated BRIEF) kako mu ime kaže koristi FAST detektor kuteva za detekciju ključnih točaka u slici i opisuje ih sa proširenom verzijom BRIEF-a. ORB deskriptor uči na setu podataka za učenje kako bi odabrao "najbolju" strategiju odabira parova za usporedbu. Kao i BRIEF koristi 256 usporedbi. Uz usporedbe ORB računa centar mase i orijentaciju (u odnosu na intenzitete piksela) u području usporedbe piksela. S ovime pokušava riješiti problem neotpornosti na rotacije od kojeg pati BRIEF.

2.4.3 Estimacija pomaka kamere

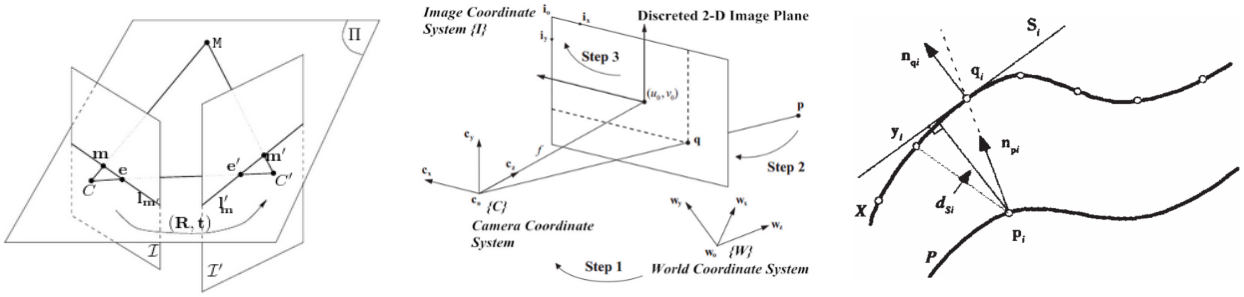
Do sada se govorilo o vizualnim značajkama, kako ih detektiramo u slici i kako ih opisujemo. Ovisno koju tehniku deskriptora značajki koristimo, koristit ćemo pripadajuću metodu usporedbe dvaju značajki. Ako je u pitanju SIFT to će biti *Euklidska udaljenost*, a ako je u pitanju neki binarni deskriptor tipa BRIEF ili ORB onda će to biti *Hammingova udaljenost*. Usporedbom dobivamo veze(podudaranja značajnih točaka) između slika. Na osnovi tih podudaranja može se estimirati pomak kamere. Ovisno o vrsti kamere koja se koristi, odnosno vrsti podataka koju dobivamo od senzora(kamere), estimacija pomaka kamere se može izračunati jednom od 3 tehnike:

- **2D-2D**
  Kada je kamera monokularna, znamo samo koordinate 2D piksela slike, pa je problem procijeniti kretanje prema dva skupa 2D točaka. Ovaj problem se rješava *epipolarnom geometrijom*. Kako bi se izračunala poza potrebno je izračunati esencijalnu matricu. Ona ima 5 stupnjeva slobode (Degrees of Freedom - DoF), što znači da je potrebno najmanje 5 podudaranja između dvije 2D slike kako bi se izračunao pokret kamere. Unutarnja svojstva esencijalne matrice su nelinearna što kod pristupa s 5 parova 2D točaka dovodi do komplikacija prilikom estimacije. *Eight-point-algorithm* koristi samo linearna svojstva esencijalne matrice, tako da se može riješiti u okviru linearne algebre, i koristi 8 parova 2D točaka(8 podudaranja).[10] Ovaj pristup pati od par problema kao što su problem s razmjerom, nemogućnost izračuna čiste rotacije kamere(potrebna je određena translacija), zahtjeva barem 8 ili više podudaranja značajki. Iz tog se razloga se problem estimacije s monokularnom kamerom svodi na: uzimanje dvije 2D slike, uz pomoć *eight-point-algorithm*-a izračuna početni pomak (ovo se zove inicijalizacija), na osnovu prvog pomaka, se metodom triangulacije mogu izračunati 3D točke iz dvije 2D slike i njihovim razmakom. Jednom kad imamo 3D točke u prostoru, svaka nova 2D slika se veže na prethodnu i koristi tehniku 3D-2D za estimaciju položaja. [10]

- **3D-2D**
  Perspective-n-Point(PnP) je metoda za rješavanje procjene 3D u 2D gibanja. Opisuje kako procijeniti položaj kamere kada je poznato n 3D prostornih točaka i njihove projekcijske pozicije. Kao što je ranije spomenuto, metoda 2D-2D epipolarne geometrije zahtijeva osam ili više parova točaka (npr. *eight point algorithm*), a postoje problemi s inicijalizacijom, čistom rotacijom i razmjerom. Međutim, ako je poznat 3D položaj značajki u slikama, tada su nam potrebna najmanje tri para (i barem jedna dodatna točka za provjeru rezultata) za procjenu kretanja kamere. 3D položaj točke značajke može se odrediti triangulacijom ili estimacijom dubine RGB-D kamerom. Stoga, u stereo ili RGB-D vizualnoj odometriji, možemo izravno koristiti PnP za procjenu kretanja kamere. Dok je u monokularnom slučaju, inicijalizacija se mora provesti prije korištenja PnP-a. 3D-2D metoda ne zahtijeva epipolarna ograničenja i može dobiti bolju procjenu kretanja u nekoliko podudarnih točaka. To je najvažnija metoda procjene poze.[10] PnP se može riješiti na više načina: linearno - često je podijeljena na mnogo koraka, kao što je prvo procjena položaja kamere, a zatim položaja točaka; nelinearno - tretira položaj kamere i točaka u prostoru kao varijable optimizacije i optimizira ih zajedno. Ovo je vrlo općenita metoda rješenja. Možemo ju koristiti za optimizaciju rezultata koje daje PnP. Ova vrsta problema, spajanje kamere i 3D točaka radi minimiziranja, općenito se naziva *bundle adjustment*.[10]

- **3D-3D**
  Kada je kamera binokularna, RGB-D, ili je udaljenost dobivena nekom metodom, tada je problem procijeniti kretanje prema dva skupa 3D točaka. Ovaj problem obično

rješava *Iterative Closest Point*(ICP). Transformacija između okvira kamere tada se procjenjuje minimiziranjem 3D euklidske udaljenosti između odgovarajućih 3D točaka.[9] Model kamere se ne pojavljuje u 3D-3D procjeni poze, što znači da kada se uzme u obzir samo transformacija između dva skupa 3D točaka, on nema nikakve veze s kamerom. Stoga je ICP također izvediv u LiDAR SLAM-u. No budući da značajke lidara nisu dovoljno bogate, teško je znati podudarni odnos između dva skupa točaka. Možemo samo pretpostaviti da su najbliže točke iste. Stoga se ova metoda naziva iterativnom najbližom točkom. Značajke nam pružaju bolji odnos podudaranja u računalnoj viziji, tako da cijeli problem postaje lakše riješiti.



Slika 6: Skice algoritama za estimaciju položaja kamere: epipolarna geometrija(lijevo), PnP(sredina), ICP(desno)

IZVOR: Zhengyou Zhang, 1996.; Xiao Xin LU, 2018.; Fang Wang, Zijian Zhao, 2018.

Svaki od gore navedenih pristupa može riješiti problem estimacije položaja kamere. Ovisno o kameri koju koristimo i strategiji vizualne odometrije, tako se i odlučujemo za neku od gore navedenih tehnika.

Dakle da sažmemo priču, ako smo koristili neku od tehnika za detektiranje ključnih točaka u slici tipa *Difference in Gaussians*(DoG) i SIFT deskriptor, te uzeli dvije slike na kojima se mogu detektirati iste značajke preko usporedbe SIFT deskriptora, pokret kamere se računa preko recimo PnP algoritma. Njemu je potrebno najmanje 3 para značajki u jednoj i drugoj slici koje se podudaraju. Ako se radi o uzastopnim slikama i rezolucije slika su visoke SIFT algoritam pronađe mnoštvo značajki i podudaranja s drugom slikom(puno više od samo 3, pogledaj sliku Slika 7). Naravno da SIFT algoritam nije savršen, tako da nisu sva podudaranja vjerodostojna. SIFT-om se dobiva dakle veliki skup podudaranja od kojih su neki dobra podudaranja(ona koja pokazuju na istu točku u 3D svijetu) i loša(koja ne spajaju istu točku u 3D svijetu). Kada bi se računala matrica transformacije pokreta kamere ne bi bilo dobro da odaberemo neke od tih loših podudaranja, jer bi dobili jednostavno krive rezultate. Kako bi se odvojila loša podudaranja iz skupa svih podudaranja može se iskoristi algoritam *Random Sample Consensus*(RANSAC).

RANSAC

RANSAC je iterativni algoritam koji traži optimalne parametre iz skupa parametara koji sadrže izvanredne parametre(anomalije) za izračun nekog matematičkog modela. Može se deklarirati kao algoritam za detekciju izvanrednih vrijednosti(outlier, anomalija). Ne deterministički je algoritam, što znači da daje razuman rezultat s određenom vjerojatnošću. Vjerojatnost raste s brojem iteracija postupka. Ideja je jednostavna i može se objasniti sljedećim koracima:

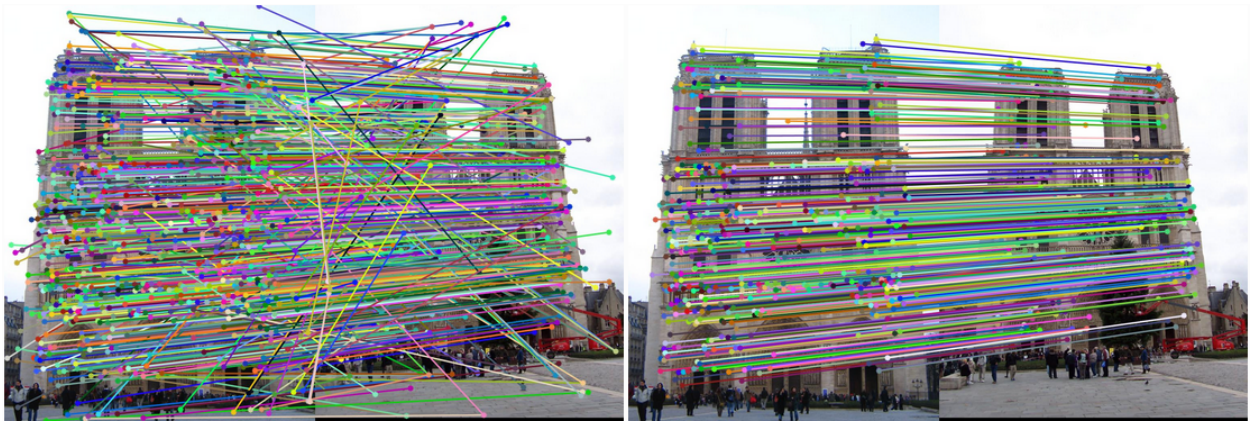
1. *Sample*: Slučajnim odabirom odaberi onaj broj podataka iz skupa svih podataka(koji sadrže anomalije) koliko ti je potrebno za izračun matematičkog modela.

2. *Compute*: Izračunaj matematički model s odabranim podatcima

3. *Score*: Ocijeni dobiveni matematički model tako da izbrojiš broj podataka koji je ostao u skupu, a koji se slaže s dobivenim matematičkim modelom

4. *Repeat*: Ponovi postupak n puta, zadržavajući onaj model s najvišom ocjenom.

Ovaj se algoritam može primijeniti na skup podudaranja vizualnih značajki. Ako koristimo PnP, onda RANSAC nalaže da se uzme najmanje 3 slučajna podudaranja i izračuna se matrica transformacije između dvije poze kamere. Zatim se ocijeni dobivena matrica transformacije tako da se izbroji koliko još podudaranja se slaže s tom matricom transformacije. Postupak se ponovi n puta i uzme se najbolje ocijenjeno rješenje.



Slika 7: RANSAC: Cijeli skup podudaranaja(lijevo), skup u kojem su izbačeni outlieri(desno)

IZVOR: John Turner, 2015.

2.5 Zatvaranje petlje i optimizacija grafa

Sve do sada bilo je opisivanje vizualne odometrije metodom praćenja vizualnih značajki. VO je samo jedan od potproblema VSLAM-a. U ovom poglavlju će se govoriti o još par važnih koraka koji se koriste u VSLAM-u baziranog na optimizaciji grafa. To su detekcija petlje zatvaranja i globalna optimizacija grafa.

2.5.1 Zatvaranje petlje

Sama vizualna odometrija, kao i druge vrste odometrije, pati od akumulacije greške kroz vrijeme. Da bi spriječili rast akumulirane greške u nedogled, VSLAM koristi tehniku zvanu zatvaranje petlje(Loop Closure). Ideja zatvaranja petlje jest prepoznati ako smo se vratili na neko prethodno posječeno mjesto i koristiti tu informaciju za ispravljanje greške koja je nastala s vremenom.

Najosnovniji oblik detekcije zatvaranja petlje je uskladiti trenutni okvir sa svim prethodnim okvirima (pohlepni pristup) korištenjem tehnika podudaranja značajki. Ovaj pristup je računski vrlo skup, zbog činjenice da se broj okvira s vremenom povećava i da usklađivanje

trenutnog okvira sa svim prethodnim okvirima nije prikladno za izvođenje aplikacije u stvarnom vremenu. Rješenje ovog problema je definiranje ključnih okvira (podskup svih prethodnih okvira) i usporedba trenutnog okvira samo s ključnim okvirima. Najjednostavniji oblik odabira ključnog okvira je odabir svakog n-tog okvira. Kako bi ubrzali proces pretrage okvira može se koristiti pristup koji se temelji na stablu vokabulara u kojem su deskriptori značajki ključnih okvira kandidata hijerarhijski kvantizirani i predstavljeni "torbom vizualnih riječi" (bag-of-visual-words(BOVW)).[9]

Detekcija petlji igra veliku ulogu u optimizaciji poza koja te petlje zatvara. Detektiranje lažno pozitivnih petlji predstavlja veliki problem. Naime dobro detektirane petlje ispravljaju akumuliranu grešku odometrije te ispravljaju pogrešno estimirane poze. Ali zato loše detektirana petlja može uništiti cijelu putanju i kartu. [11]

Kao što je navedeno u potpoglavlju 2.2.2 (*Pristupi temeljeni na optimizaciji grafa*) prepoznate petlje tvore nova, jako bitna, ograničenja u grafu između poza. Dobiveni graf s pozama, odometrijskim ograničenjima i opaženim(prepoznavanje petlje) ograničenjima se šalje u "*backend*" kako bi se dobivene poze na osnovu ograničenja optimizirale.

2.5.2 Optimizacija grafa

VO je inkrementalna estimacija koja pokušava estimirati putanju kamere lokalno (iz slike u sliku) no gledano globalno ta putanja akumulira grešku kroz vrijeme. Kako bi VSLAM omogućio globalnu dosljednost potrebno je optimizirati poze kamere s vremena na vrijeme.

Bundle adjustment

U fotogrametriji, prilagodba paketa(eng. Bundle adjustment) je simultano preciziranje 3D koordinata koje opisuju geometriju scene, parametre relativnog gibanja i optičke karakteristike kamere(a) korištene za dobivanje slika, s obzirom na skup slika koje prikazuju nekoliko 3D točaka s različitih točaka gledišta. Njegovo ime odnosi se na geometrijske snopove svjetlosnih zraka koje potječu iz svake 3D značajke i skupljaju se u optičko središte svake kamere, a koje su optimalno podešene prema kriteriju optimalnosti koji uključuje odgovarajuće projekcije slike svih točaka. Prilagodba paketa se gotovo uvijek koristi kao posljednji korak svakog algoritma 3D rekonstrukcije temeljenog na značajkama. Izvorno je zamišljena u polju fotogrametrije tijekom 1950-ih, a posljednjih godina sve više je koriste istraživači računalnog vida, te u robotičkim aplikacijama kada se rješava problem VSLAM-a.

Prilagodba paketa svodi se na minimiziranje greške reprojekcije između lokacija slike promatranih i predviđenih točaka slike, koja se izražava kao zbroj kvadrata velikog broja nelinearnih funkcija sa stvarnim vrijednostima. Stoga se minimizacija postiže uporabom nelinearnih algoritama najmanjih kvadrata. Među njima, Levenberg-Marquardt se pokazao kao jedan od najuspješnijih zbog svoje jednostavnosti implementacije i upotrebe učinkovite strategije prigušenja koja mu daje mogućnost brze konvergacije iz širokog raspona početnih pretpostavki. Iterativnom linearizacijom funkcije koju treba minimizirati u blizini trenutne procjene, Levenberg-Marquardtov algoritam uključuje rješavanje linearnih sustava koji se nazivaju normalnim jednadžbama. Prilikom rješavanja problema minimizacije koji nastaju u okviru prilagodbe snopa, normalne jednadžbe imaju rijetku blok strukturu zbog nedostatka interakcije između parametara za različite 3D točke i kamere. Ovo se može iskoristiti za postizanje ogromnih računalnih prednosti upotrebom rijetke varijante Levenberg-Marquardt algoritma koji eksplicitno iskorištava prednost uzorka nula normalnih jednadžbi, izbjegavajući pohranjivanje i rad na nultim elementima.

Pretpostavimo da n 3D točaka se vide u m prikaza i neka je x_{ij} projekcija i-te točke na slici j. Neka v_{ij} označava binarnu vrijednost jednaku 1 ako se točka i vidi u slici j, u protivnom 0. Pretpostavimo da je svaka kamera j parametrizirana vektorom a_j i svaka 3D točka i sa vektorom b_i. Prilagodba paketa minimizira totalnu reprojekcijsku pogrešku u odnosu na sve 3D točke i parametre kamere, posebno gdje je $Q(a_j, b_i)$ predviđena projekcija točke i na slici j, a $d(x, y)$ označava Euklidsku udaljenost između točaka slike predstavljenim vektorom x i y.

$$\min_{\mathbf{a}_j,\, \mathbf{b}_i} \sum_{i=1}^{n} \sum_{j=1}^{m} v_{ij}\, d(\mathbf{Q}(\mathbf{a}_j,\, \mathbf{b}_i),\ \mathbf{x}_{ij})^2 \tag{1}$$

Prilagodba paketa je nelinearni problem optimizacije najmanjih kvadrata(least squares - LS). Nelinearni problem najmanjih kvadrata se može optimizirati korištenjem spomenute Levenberg–Marquardt ili Gauss-Newton metode. Neki od alata otvorenog koda koji nude implementacije ovih metoda za optimizaciju grafa su g2o, GTSAM, Ceres.[12]

3 ROBOTSKI OPERACIJSKI SUSTAV(ROS)

Robotski operacijski sustav(eng. Robot Operating System(ROS)) je metaoperativni sustav za robote, otvorenog koda. Pruža važne usluge koje pružaju drugi operacijski sustavi, uključujući hardversku apstrakciju, nisku kontrolu uređaja, implementaciju često korištenih funkcionalnosti, prijenos poruka između procesa i upravljanje paketima. Također nudi alate i biblioteke za dobivanje, izgradnju, pisanje i izvođenje koda na više računala [13].

Prva inačica ROS-a (u nastavku ROS1), stvorena je 2007. godine od laboratorija za istraživanje robotike *Willow Garage*, i ubrzo se proširila zajednicom za robotiku. Tim koji stoji iza izgradnje ROS1 naučio je, sa svim godinama iskustva, koje važne značajke nedostaju i što bi se moglo poboljšati. Nažalost, dodavanje svih tih izmjena u ROS1 zahtijevalo bi mnoge promjene i učinilo ROS1 prilično nestabilnim. Iz tog se razloga odlučilo razviti novi ROS(ROS2) od početka. Za sada ROS nije jako popularan u industriji, a i nedostaju mu neki od najvažnijih zahtjeva, kao što su izvođenje u stvarnom vremenu, sigurnost, certifikacija, zaštita. Jedan od ciljeva ROS2 je učiniti ga kompatibilnim s industrijskim aplikacijama.[14] U nastavku će biti navedene neke razlike između ROS1 i ROS2.

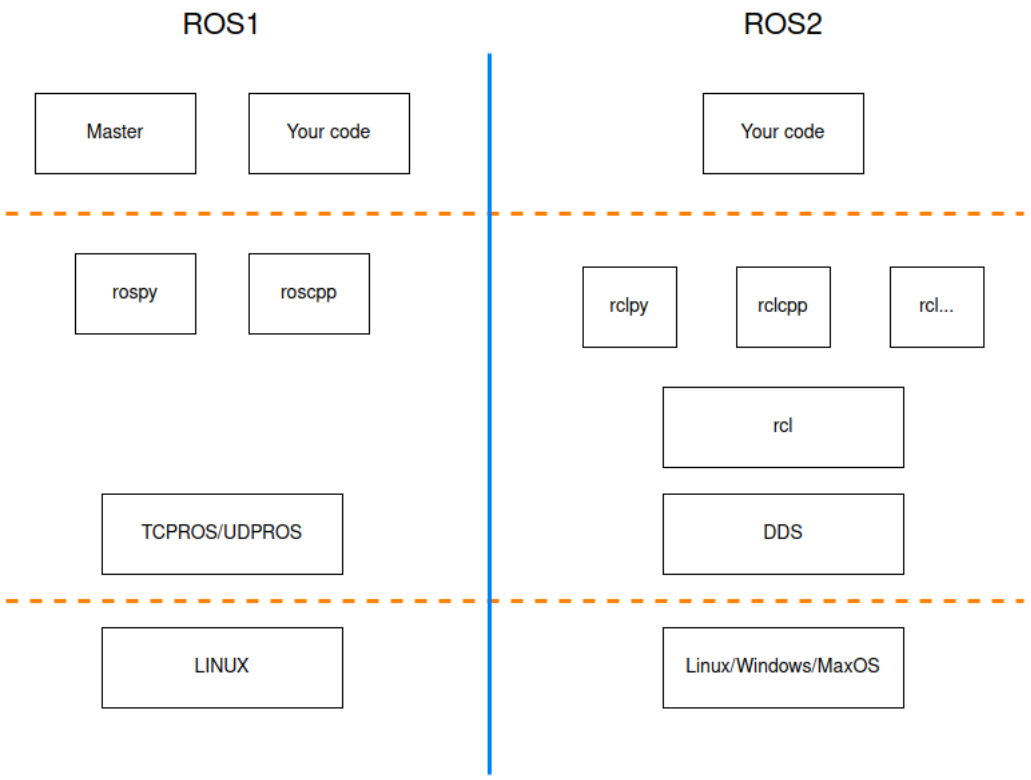
ROS API

ROS1 ima 2 API-a, jedan za cpp programski jezik - roscpp, i jedan za python programski jezik - rospy. Obje su knjižnice potpuno neovisne i izgrađene od nule. To znači da API nije nužno isti između roscpp i rospy, a neke značajke su razvijene za jedno, a ne za drugo.

ROS2 ima više slojeva. Postoji samo jedna osnovna knjižnica **"rcl"**, koja je implementirana u C programskom jeziku. Ona je temelj koji sadrži sve osnovne ROS2 funkcionalnosti. Dakle rcl je jezgrena knjižnica koja pruža sve funkcionalnosti koje pruža ROS. Ona se ne koristi direktno u programima, već se koristi neka druga klijentska knjižnica izgrađena povrh rcl-a. Na primjer "rclcpp" za Cpp, "rclpy" za Python. Super stvar kod ovog pristupa je ta da svaka nova funkcionalnost samo treba biti implementirana u rcl-u. Zatim, klijentske knjižnice na povrh rcl-a samo trebaju osigurati vezivanje.
Dodatni benefiti koje pruža ovaj pristup za programere su:

- API između rclcpp i rclpy bit će mnogo sličniji nego API između roscpp i rospy.
- Bit će lakše kreirati i koristiti klijentske knjižnice drugih jezika, na primjer rclnodejs, rcljava, itd. Nema potrebe za svaki jezik posebno pisati funkcionalnost, ona se napiše jednom u rcl-u, a sve druge klijentske knjižnice samo napišu vezivanje s njom na svom jeziku.
- I svi klijenti na svim jezicima imat će sličan API.
- Kada bude objavljena nova temeljna značajka, bit će dostupna prije na različitim jezicima, tako da se neće morati previše čekati.

Shematski prikaz razlike u arhitekturi ROS1 i ROS2 može se vidjeti na slici Slika 8.



Slika 8: Razlika između ROS1 i ROS2 arhitekture

Pisanje čvorova(OOP)

U ROS1 ne postoji posebna struktura koja govori kako biste trebali napisati funkcionalnost čvora. Možete odlučiti dodati funkcije povratnog poziva bilo gdje u svom programu ili koristiti OOP ako želite, ali svaka implementacija može biti jedinstvena.

U ROS2 stvari su drugačije. Postoji konvencija o tome kako napisati svoje čvorove. Morate stvoriti klasu koja nasljeđuje od objekta Node (na primjer: rclcpp::Node u Cpp-u, rclpy.node.Node u Pythonu). U ovom razredu imat ćete sve svoje ROS2 funkcije.

Ovo je sjajno jer će svima uštedjeti puno vremena iz razloga jer postoji modularna struktura za pisanje čvora. To će učiniti programe čišćima, a suradnja među programerima na različitim projektima bit će lakša.

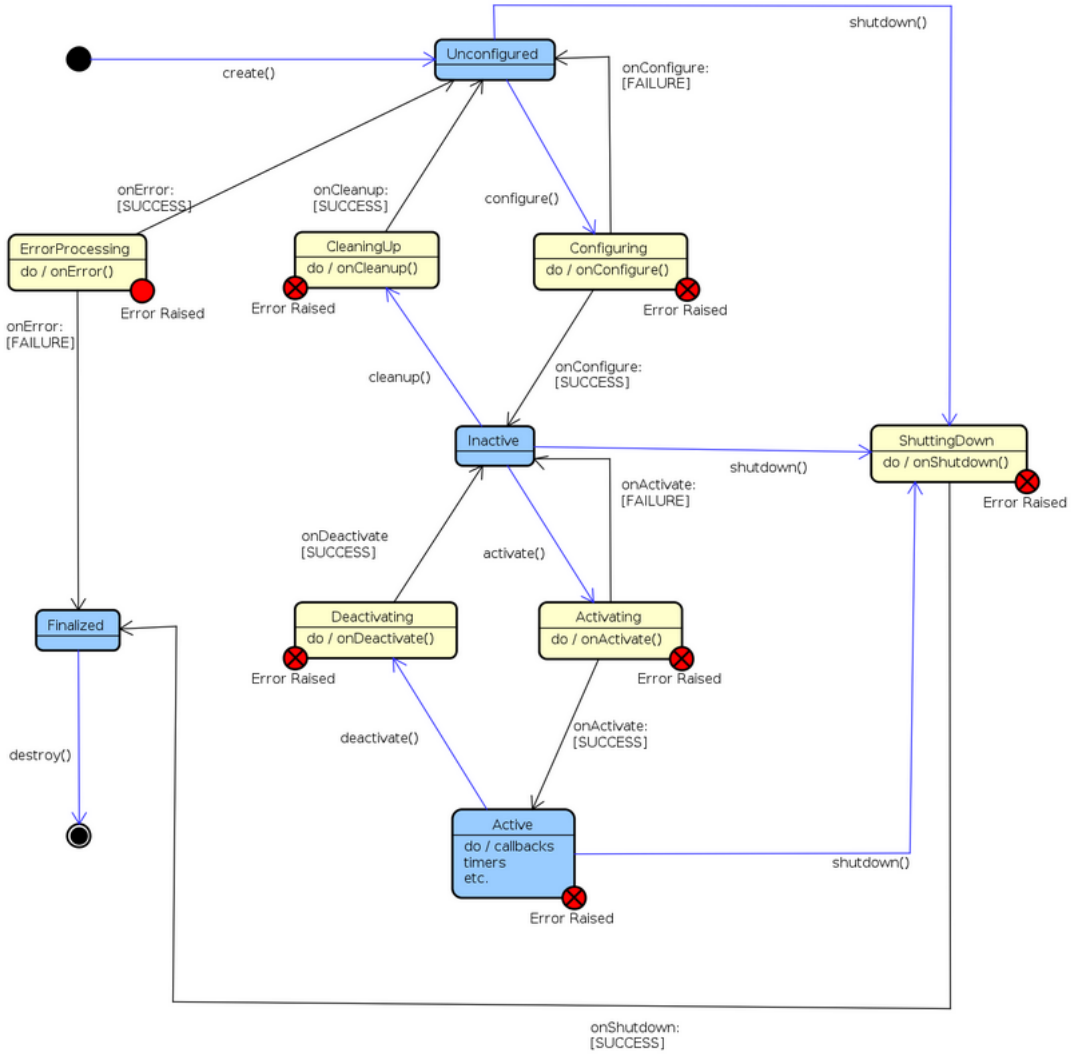
Čvorovi sa životnim ciklusom

ROS2 uvodi koncept čvorova sa životnim ciklusom. Čvor sa životnim ciklusom ima različita stanja: nekonfiguriran, neaktivan, aktivan, finaliziran. Ovo je vrlo korisno kada je potrebna faza postavljanja prije stvarnog pokretanja glavnih funkcionalnosti čvora.

Kada se pokrene takav čvor, on je u početku nekonfiguriran. Putem pruženog sučelja (ROS2 usluge) može se zatražiti prijelaz u drugo stanje. Kada se to učini, unutar čvora će se pokrenuti unaprijed definirani povratni poziv.

Recimo da imate čvor za senzor. Najprije morate provjeriti je li senzor otkriven i je li komunikacija uspješno pokrenuta. Zatim možete pokrenuti svoj ciklus čitanja i objaviti podatke. S čvorom sa životnim ciklusom ovo možete jasno odvojiti: prvo dodijelite memoriju za izdavače, pretplatnike i druge instancirane objekte. Zatim započinjete komunikaciju sa senzorom. I konačno pokrećete svoju petlju čitanja kako biste objavili podatke.

Grafički prikaz životnog siklusa može se vidjeti na slici Slika 9.



Slika 9: Čvor sa životnim ciklusom u ROS2

IZVOR: Geoffrey Biggs, Tully Foote, 2021(zadnji put uređivano)

Pisanje datoteke za pokretanje

Datoteke za pokretanje omogućuju pokretanje svih čvorova iz jedne datoteke. Mogu se pokrenuti standardni čvor, komponenta, čvor u životnom ciklusu. Mogu se dodati argumenti, parametri i mnoge druge opcije.

U ROS1 se za pisanje datoteka za pokretanje koristi XML. Moguće je datoteke za pokretanje također pisati i u python-u ali nije dobro dokumentirano za ROS1, te je za mnoge i nepoznanica. Općenito za ROS1 je u zajednici prihvaćeno pisanje datoteka za pokretanje u XML-u.

U ROS2 će se sada koristiti Python za pisanje datoteka za pokretanje. Postoji API koji omogućuje pokretanje čvorova, dohvaćanje konfiguracijskih datoteka, dodavanje parametara itd. I omogućit će prilagođavanje datoteke za pokretanje mnogo više nego prije. U ROS2 je također omogućeno pisanje datoteka za pokretanje u XML-u. Python donosi više modularnosti, programsku logiku, više je dokumentiran i postao je ROS2 konvencija za datoteke pokretanja.

ROS Master

ROS Master pruža usluge imenovanja i registracije za ostale čvorove u ROS sustavu. Prati izdavače i pretplatnike na teme i usluge. Uloga Mastera je omogućiti pojedinačnim ROS čvorovima da lociraju jedan drugog. Nakon što se ti čvorovi međusobno lociraju, međusobno komuniciraju peer-to-peer. Master također osigurava poslužitelj parametara. Master se najčešće pokreće pomoću naredbe "*roscore*", koja učitava ROS Master zajedno s ostalim bitnim komponentama. Ovo je praksa za ROS1, centraliziran sustav.

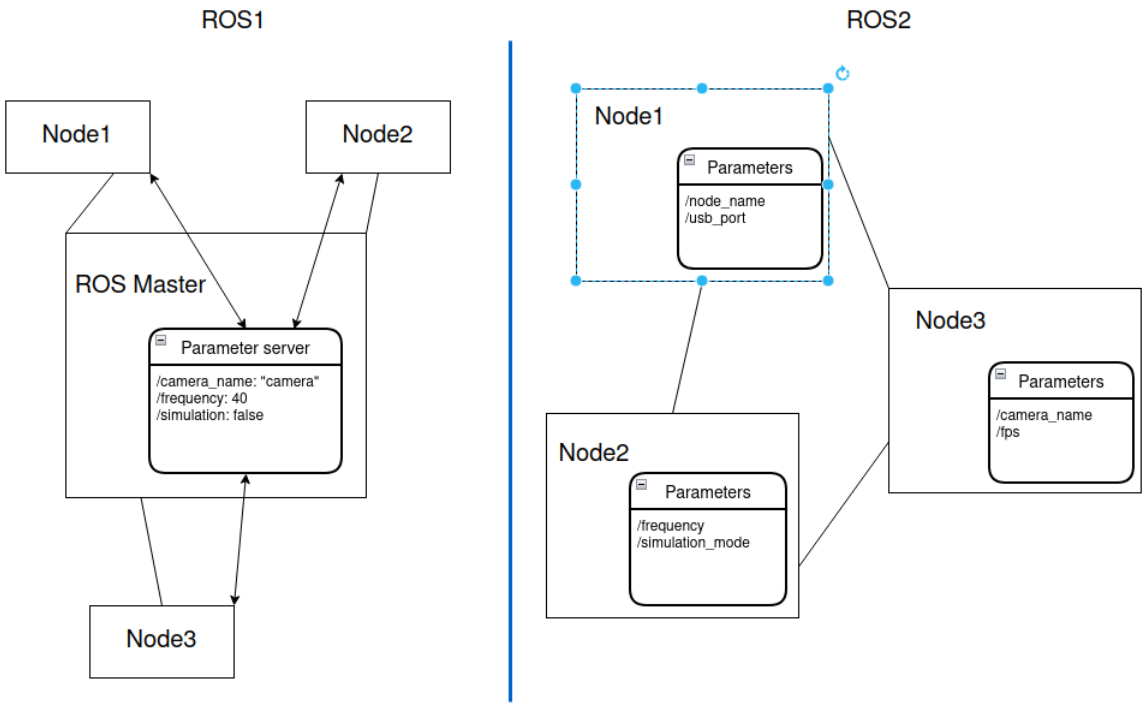
U ROS2, nema više ROS mastera! Ovo više nije centralizirani sustav. Svaki čvor ima sposobnost otkrivanja drugih čvorova. Može se jednostavno pokrenuti čvor bez brige da li postoji master koji radi ili ne. Ova promjena je sjajna jer omogućuje stvaranje potpuno distribuiranog sustava. Svaki čvor je neovisan i nije vezan za globalnog gospodara.

Prilikom izrade ROS2 aplikacije za više strojeva, ne mora se definirati jedan stroj kao "glavni". Svaki će stroj biti neovisan i moći će se samostalno pokretati, međusobno se povezivati i odspajati, s manje podešavanja nego u ROS1.

Parametri

Parametri su neka vrsta globalnih varijabli u ROS1. Parametrima upravlja poslužitelj parametara, kojim sam upravlja ROS Master. Unutar ROS Mastera se nalazi globalni server za parametre.

U ROS2 više nema ROS Mastera, također nema globalnog poslužitelja parametara. Koncept parametara je potpuno promijenjen. Više nema globalnog parametra, svaki je parametar specifičan za čvor. Čvor deklarira i upravlja svojim vlastitim parametrima, a ti se parametri uništavaju kada je čvor ubijen. Kao da svaki čvor ima svoj vlastiti poslužitelj parametara. Kada se pokrene čvor, kreira se nekoliko ROS2 usluga. Oni vam omogućuju interakciju s njegovim parametrima s terminala ili iz drugih čvorova. Osim toga, mogu se jednostavno izmijeniti parametri čvora nakon što su stvoreni, koristeći povratni poziv parametara.



Slika 10: ROS Master, server za parametre - usporedba ROS1 i ROS2

Definicije poruka, usluga i radnji (Messages, Services, Actions)

Način stvaranja definicija za poruke, usluge i radnje prilično je sličan u ROS1 i ROS2. I dalje se stavljaju u mape msg/, srv/ i action/. Razlika je to što u ROS2 nakon prevođenja koda poruci se dodaje namespace msg/, usluzi srv/, akciji action/.

Na primjer, recimo da imate paket pod nazivom *my_robot_msgs*, a unutar tog paketa ste stvorili poruku pod nazivom Temperature, plus uslugu pod nazivom ActivateButton. U kodu vašeg čvora morat ćete ih uvesti(importati) na sljedeći način:

ROS1

- my_robot_msgs/Temperature
- my_robot_msgs/ActivateButton

ROS2

- my_robot_msgs/**msg**/Temperature
- my_robot_msgs/**srv**/ActivateButton

Može se reći da ROS2 smanjuje zbrku i čini razdvajanje jasnijim između 3 vrste komunikacija.

Izgradnja(build) čvorova

Sustav izgradnje u ROS1 je *catkin*. Da bi izgradili i instalirali pakete koristi se komanda *"catkin_make"* ili *"catkin build"*.

U ROS2, nema više *catkin*. *Ament* je novi sustav izgradnje, a povrh toga dobivate *colcon* alat naredbenog retka. Za prevođenje se koristi naredba *"colcon build"* u svom radnom prostoru ROS2.

Alati naredbenog retka (command line)

Većina alata naredbenog retka slična je između ROS1 i ROS2. Nazivi alata i neke opcije su različiti, ali inače nema velike razlike kada se koristite. Na primjer da bi izlistali sve teme u ROS1 bi koristili komandu *"rostopic list"*, a u ROS2 *"ros2 topic list"*; *"rosservice"* u ROS2 je *"ros2 service"*; *"rosrun"* postaje *"ros2 run"*, itd. Zaključak u ROS2 se prvo napiše *"ros2"* potom razmak, pa ime alata.

Cpp i Python paketi

U ROS1 se stvori paket, a zatim se dodaje bilo koja Cpp/Python datoteka u taj paket.
ROS2 čini razliku između Cpp i Python paketa. Kada kreirate paket iz naredbenog retka, mora se navesti jedna vrsta izgradnje paketa:

- ament_cmake
- ament_python

Ovisno o tom argumentu, arhitektura paketa neće biti ista.

Za Cpp paket u ROS2, stvari su prilično slične kao i u ROS1. I dalje se stvori CMakeLists.txt datoteka. Moraju se samo prilagoditi cmake upute za korištenje *ament*-a, a ne *catkin*-a.

Za Python paket stvari su drugačije: prilikom kreiranja paketa dobivaju se neke nove datoteke, kao što su setup.py i setup.cfg. Setup.py zamjenjuje CMakeLists.txt. Python skripte se mogu izravno pokretati, ali ako se one žele pokrenuti uz pomoć ROS2 alata naredbenog retka ili preko datoteke za pokretanje, prvo ih je potrebno instalirati (s *"colcon build"*).

Ako je potreba također je moguće stvoriti jedan paket za Python i Cpp, no to zahtjeva malo više podešavanja. Općenito je postavljanje paketa u ROS2 složenije nego u ROS1, ali je također potpunije i organiziranije.

Podrška za OS

ROS1 je uglavnom fokusiran samo na Ubuntu.
ROS2 Zahvaljujući njegovoj novoj arhitekturi, može ga se instalirati i koristiti na Ubuntu, MacOS, Windows, te drugim OS-ima. To će učiniti ROS2 pristupačnijim i ugradljivijim u mnoge aplikacije. Na primjer, možemo imati mobilnog robota s Raspberry Pi i Ubuntu-om i drugo računalo koje koristi Windows s alatom za 3D simulaciju i upravljački čvor za kameru koja skenira scenu. Sve navedene informacije oko usporedbe ROS1 i ROS2 mogu se pročitati ovdje [14].

4 INTEL REALSENSE KAMERA

Intel-ova RealSense tehnologija je asortiman tehnologija za dubinu i praćenje osmišljen kako bi strojevima i uređajima omogućili percepciju dubine. Intel RealSense tehnologija se može koristi u autonomnim dronovima, robotima, Augmented Reality (AR), Virtual Reality (VR), pametnim kućnim uređajima, itd. RealSense proizvod se sastoji od procesora za vid, modula za dubinu i praćenje, i dubinskih kamera. Podržan je sa SDK-om otvorenog koda za više platformi(cross platform), pojednostavljujući kamere za podršku za programere softvera trećih strana. U nastavku će biti navedene neke tehnologije koje nudi RealSense za računanje dubine, njihove razlike i primjene.

SR300 linija(Kodirano svjetlo)

Kodirano svjetlo je potklasa tehnika strukturiranih svijetla. Oni se oslanjaju na projiciranje svjetlosti (obično infracrvene svjetlosti) iz neke vrste emitera na scenu. Projicirana svjetlost je uzorkovana, vizualno ili tijekom vremena, ili neka kombinacija tog dvoje. Budući da je projicirani uzorak poznat, način na koji senzor u kameri vidi uzorak u sceni pruža informacije o dubini. Na primjer, ako je uzorak niz pruga projiciranih na loptu, pruge bi se deformirale i savijale oko površine lopte na specifičan način.

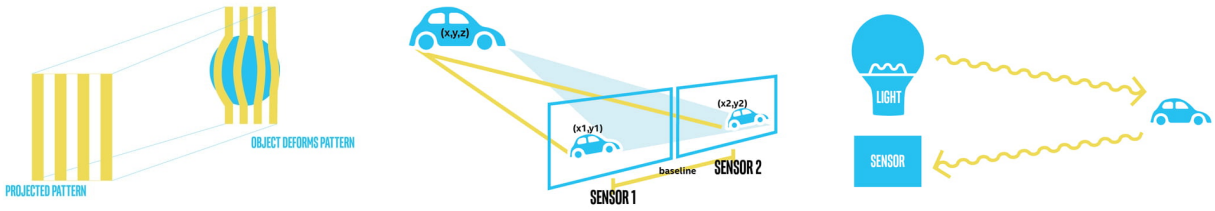
Intel RealSense SR300 linija uređaja su kamere s kodiranim svjetlom. Budući da se ova tehnologija oslanja na točno viđenje projiciranog uzorka svjetla, kamere s kodiranim i strukturiranim svjetlom najbolje rade u zatvorenom prostoru na relativno kratkim udaljenostima (ovisno o snazi svjetlosti koju emitira kamera). Drugi problem s ovakvim sustavima je taj što su osjetljivi na drugu buku u okolišu od drugih kamera ili uređaja koji emitiraju infracrveno svijetlo. Idealne upotrebe za kamere s kodiranim svjetlom su stvari kao što su prepoznavanje gesta ili segmentacija pozadine (također poznata kao virtualni zeleni zaslon).[15]

D400 linija(Stereo dubina)

Stereo kamera je kamera koja ima dvije ili više leća sa zasebnim senzorom slike za svaku leću. To omogućuje kameri da simulira ljudski dvo-okularni vid i time joj daje mogućnost snimanja trodimenzionalnih slika. Kako je razmak između leća poznat i fiksan, dubina se računa metodom triangulacije.

Stereo dubinske kamere također često projiciraju infracrveno svjetlo na scenu kako bi poboljšale točnost podataka, ali za razliku od kodiranih ili strukturiranih svjetlosnih kamera, stereo kamere mogu koristiti bilo koje svjetlo za mjerenje dubine. Za stereo kameru, sav infracrveni šum je dobar šum.

Budući da stereo kamere koriste sve vizualne značajke za mjerenje dubine, dobro će raditi u većini uvjeta osvjetljenja, uključujući i na otvorenom. Dodatak infracrvenog projektora znači da u uvjetima slabog osvjetljenja kamera i dalje može uočiti detalje dubine. Kamere serije Intel RealSense D400 su stereo dubinske kamere. Druga prednost ove vrste dubinske kamere je da nema ograničenja koliko ih možete koristiti u određenom prostoru - kamere ne ometaju jedna drugu na isti način kao što bi to činila kamera za kodirano svjetlo ili kamera koja koristi vrijeme leta za računanje dubine(LiDAR).[15]



Slika 11: Ilustracije različitih metoda računanja dubine: Strukturirano svijetlo(lijevo), stereo dubina(u sredini), vrijeme leta - LiDAR (desno)

L500 linija(vrijeme leta - LiDAR)

Svaka vrsta dubinske kamere oslanja se na poznate informacije kako bi ekstrapolirala dubinu. Na primjer, u stereo, udaljenost između senzora je poznata. Kod kodiranog svjetla i strukturiranog svjetla, obrazac svjetlosti je poznat. U slučaju vremena leta, brzina svjetlosti je poznata varijabla koja se koristi za izračunavanje dubine. LiDAR senzori vrsta su kamere za vrijeme leta koja koristi lasersko svjetlo za izračunavanje dubine. Sve vrste uređaja za vrijeme leta emitiraju neku vrstu svjetlosti, prevlače je po sceni, a zatim mjere koliko je vremena potrebno tom svjetlu da se vrati do senzora na kameri. Ovisno o snazi i valnoj duljini svjetlosti, senzori vremena leta mogu mjeriti dubinu na značajnim udaljenostima – na primjer, koriste se za kartiranje terena iz helikoptera.

Primarni nedostatak kamera za vrijeme letenja je to što one mogu biti osjetljive na druge kamere u istom prostoru, a mogu i slabije funkcionirati u vanjskim uvjetima. Svaka situacija u kojoj svjetlost koja udari u senzor možda nije svjetlost emitirana iz kamere, već je mogla doći iz nekog drugog izvora poput sunca ili druge kamere, te može pogoršati kvalitetu dubinske slike.[15]

Slika 12: Prikaz različitih Intel RealSense kamera: SR305 (lijevo), D455 (u sredini), L515 (desno)

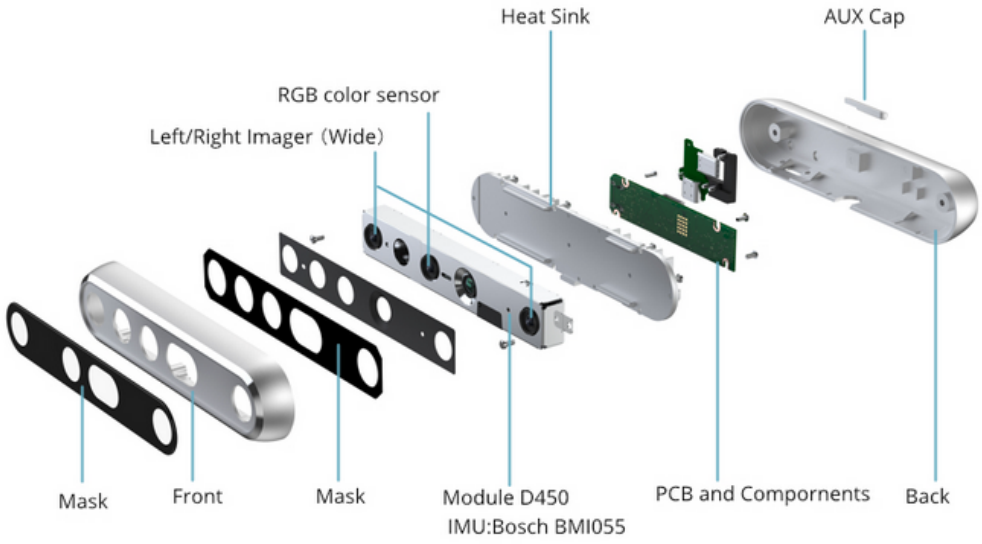
Tehnologija koja se odabire za ovaj projekt je tehnologija koja računa dubinu stereo kamerom. Iz prethodne analize da se zaključiti kako je ta tehnologija najpovoljnija za vanjsko okruženje jer najmanje pati od vanjskog osvjetljenja. Intel RealSense nudi više kamera u D400 seriji kao što su D405, D415, D435/D435i, D455. Najnovija od njih je D455, nju sam i odabrao za projekt, kako je ona bila dostupna na fakultetu. U nastavku slijedi kratak opis i usporedba s prethodnicima.

Intel RealSense D455 kamera

Intel RealSense D455 kamera je četvrta kamera po redu iz Intel-ove serije D400 kamera. To je dubinska stereo kamera koja uz standardnu izlaznu sliku u boji(RGB) može dati informaciju o dubini svakog piksela u slici. Spaja se preko USB-C porta na računalo. Ima ugrađen Bosch BMI055[9] IMU. U odnosu na svoje prethodnike D435i/D435 produljuje udaljenost(razmak) između senzora dubine na 95mm čime se pogreška dubine poboljšava i iznosi manje od 2% na 4 m.

Tablica 1: Usporedba specifikacija Intel RealSense D435/D435i i D455

Spec	D435/D435i	D455
Use Environment	Indoor/Outdoor	Indoor/Outdoor
Depth Technology	Stereoscopic	Stereoscopic
Baseline	50mm	95mm
Depth Sensor Technology	Global Shutter	Global Shutter
Depth Resolution	Up to 1280 × 720	Up to 1280 × 720
Depth Accuracy	<2% at 2m	<2% at 4m
Depth Frame Rate	Up to 90 fps	Up to 90 fps
RGB Sensor Technology	Rolling Shutter	Global Shutter
RGB Sensor Resolution	2 MP	1MP
Left/Right Imagers type	Wide	Wide
IR Projector	Wide	Wide
RGB Frame Rate and Resolution	1920 × 1080 at 30 fps	1280 × 800 at 30 fps
Inertial Measurement Unit	No/Yes	Yes
Min. Depth Distance (at Max. Reso.)	∼28 cm	∼52 cm
Ideal Range	0.3 m to 3 m	0.6m to 6m



Slika 13: Intel RealSense D455

[9]https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi055/

5 ISTRAŽENA RJEŠENJA ZA KARTIRANJE U STVARNOM VREMENU OTVORENOG KODA

Postoji veliki izbor SLAM pristupa otvorenog koda koji su dostupni putem ROS-a. U ovom poglavlju osvrnut ćemo se na najpopularnije te istaknuti njihove karakteristike. Listu popularnijih pristupa rješavanju SLAM problema može se vidjeti u tablici Table 2.

Tablica 2: Popis poznatijih SLAM rješenja otvorenog koda

Naziv rješenja	Senzor*	URL
Hector SLAM	L	http://wiki.ros.org/hector_slam
GMapping	L	http://wiki.ros.org/gmapping
Cartographer	L	https://google-cartographer-ros.readthedocs.io
SLAM toolbox	L	https://github.com/SteveMacenski/slam_toolbox
MonoSLAM	M	https://github.com/hanmekim/SceneLib2
PTAM	M	https://github.com/Oxford-PTAM/PTAM-GPL
ORB-SLAM	M/S/R	https://webdiis.unizar.es/~raulmur/orbslam/
DTAM	M	https://github.com/anuranbaka/OpenDTAM
LSD-SLAM	M	https://github.com/tum-vision/lsd_slam
SVO-Pro	M/S	https://github.com/uzh-rpg/rpg_svo_pro_open
DVO-SLAM	R	https://github.com/tum-vision/dvo_slam
VINS-Mono	M+IMU	https://github.com/HKUST-Aerial-Robotics/VINS-Mono
VINS-Fusion	M/S+IMU	https://github.com/HKUST-Aerial-Robotics/VINS-Fusion
RGBD-SLAM-V2	R	https://github.com/felixendres/rgbdslam_v2
RTABMAP	S/R/L	https://github.com/introlab/rtabmap
Elastic Fusion**	R	https://github.com/mp3guy/ElasticFusion

* M - Monocular; S - Stereo; R - RGBD; L - Lidar
**Elastic Fusion nije dostupan u ROS-u

Kao što se može vidjeti u tablici Table 2 ima popriličan broj rješenja, a ovo su samo neka od poznatijih. Također se može primijetiti da nisu sva rješenja SLAM-a vezana uz korištenje kamere. **Ovaj rad će obratiti pažnju isključivo na ona koja koriste kamere i VO.**

MonoSLAM

Profesor A. J. Davison predložio je MonoSLAM 2007. kao prvi monokularni vizualni SLAM sustav u stvarnom vremenu. MonoSLAM koristi prošireni Kalmanov filtar(EKF) kao pozadinu za praćenje vrlo rijetkih značajki na frontendu. Budući da EKF zauzima istaknutu dominantnu poziciju u ranom SLAM-u, MonoSLAM se također temelji na EKF-u, koristeći trenutno stanje kamere i sve orijentirane točke kao predodžbu pozicije, te ažurira srednju vrijednost i kovarijancu pozicije. Ovaj pristup je bio prekretnica u svoje vrijeme, jer većina prethodnih VSLAM sustava nije mogla raditi online. S moderne točke gledišta, MonoSLAM ima nedostatke kao što su mali scenariji, ograničen broj orijentira i rijetke značajke koje je vrlo lako izgubiti. Njegov rad na održavanju također je zaustavljen i zamijenjen naprednijim teorijama i programskim alatima. [10]

PTAM

Kleinov tim je 2007. godine predložio PTAM (Parallel Tracking and Mapping) [97], koji je također važan događaj u razvoju vizualnog SLAM-a. Važan značaj PTAM-a leži u sljedeće dvije točke:

- PTAM je predložio i realizirao paralelizaciju procesa praćenja i kartiranja. Postaje jasno da dio za praćenje treba odgovoriti na slikovne podatke u stvarnom vremenu, a optimizacija se ne mora izračunati u stvarnom vremenu. Pozadinska optimizacija može se obaviti polako u pozadini, a zatim se po potrebi može izvesti sinkronizacija procesa. Ovo je prvi put da se koncept *frontend* i *backend* dijela razlikuje u vizualnom SLAM-u, što je dovelo do strukture mnogih kasnijih vizualnih SLAM sustava (većina SLAM-ova koje sada vidimo podijeljena je na *frontend* i *backend*).

- PTAM je prvo rješenje u stvarnom vremenu koje koristi nelinearnu optimizaciju umjesto tradicionalnih filtara u pozadini. Uvodi mehanizam ključnih kadrova: ne moramo pažljivo obraditi svaku sliku, već nizati nekoliko ključnih slika i optimizirati njezinu putanju i mapu. Rani SLAM uglavnom je koristio EKF filtere ili njihove varijante. Nakon PTAM-a, vizualna SLAM istraživanja postupno su se okrenula backendu u kojem je dominirala nelinearna optimizacija (bundle adjustment).

PTAM je također softver proširene stvarnosti koji pokazuje obećavajuće AR efekte. Prema pozi kamere koju procjenjuje PTAM, možemo postaviti virtualni objekt na virtualnu ravninu, koja izgleda kao da je u stvarnoj sceni. Kao i mnogi prethodni radovi, postoje očiti nedostaci: scena je mala, praćenje je lako izgubiti itd. Oni su poboljšani u novijim sustavima.[10] Neki od novijih izvedbi PTAM-a su S-PTAM (stereo PTAM)[10] i MCPTAM(Multi camera PTAM)[11].

ORB-SLAM

ORB-SLAM je vrlo poznat među nasljednicima PTAM-a. Predložen je 2015. godine i jedan je od najcjelovitijih i najjednostavnijih za korištenje sustava u modernim SLAM sustavima (ako ne i najcjelovitiji i najjednostavniji za korištenje).[10] ORB-SLAM je vrhunac SLAM-a otvorenog koda koji se temelji na mainstream značajkama. U usporedbi s prethodnim radovima, ORB-SLAM ima sljedeće očite prednosti:

- Podržava različite postavke senzora: monokularni, stereo i RGB-D. Za većinu senzora gotovo ga je moguće testirati na ORB-SLAM-u. Ima dobru svestranost.

- Cijeli sustav se izračunava s ORB značajkama, uključujući ORB rječnik za vizualnu odometriju i detekciju petlje. ORB značajke pružaju kompromis između brzine računanja i kvaliteti usporedbe u odnosu na druge deskriptore. Pružaju robusnost na rotacije i skaliranje.

- ORB-ova detekcija petlje je njegov vrhunac. Izvrstan algoritam detekcije zatvaranja petlje osigurava da ORB-SLAM učinkovito sprječava nagomilane pogreške i da se može brzo vratiti nakon gubitka. Mnogi postojeći SLAM sustavi nisu savršeni u točki relokalizacije. Iz tog razloga, ORB-SLAM treba učitati veliku ORB datoteku rječnika.

- ORB-SLAM inovativno koristi strukturu tri dretve u SLAM-u: dretva praćenja - za praćenje značajki u stvarnom vremenu, optimizacijska dretva za lokalnu prilagodbu poza i 3D značajki (*local bundle adjustment*)(Graf ko-vidljivosti, uobičajeno poznat kao mali graf) i dretva optimizacije grafa globalne pozicije(*global pose graph optimization*)(esencijalni graf, poznat kao veliki graf). Dretva za praćenje je odgovorna za izdvajanje točaka ORB značajki za svaku novu sliku, uspoređivanje s najnovijim ključnim kadrom, izračunavanje lokacije značajki i grubu procjenu položaja kamere. Mali graf

[10]https://github.com/lrse/sptam
[11]https://github.com/aharmat/mcptam

rješava problem prilagodbe lokalnog paketa(zadnjih par okvira), uključujući točke značajki u lokalnom prostoru i pozu kamere. Ova je dretva odgovorna za pročišćavanje poza kamere i prostornih lokacija značajki. Dretva za praćenje i lokalno kartiranje stvaraju dobru vizualnu odometriju. Treća dretva, dretva velikog grafa, izvodi detekciju petlje na globalnoj karti i ključnim kadrovima kako bi se eliminirale nakupljene pogreške. Slijedeć dvo-dretvenu strukturu PTAM-a, tro-dretveni dizajn ORBSLAM-a postigao je izvrsne efekte praćenja i mapiranja, osiguravajući globalnu konzistentnost putanje i karte. Ova trostruka struktura također će biti prihvaćena i usvojena od strane kasnijih istraživača.

- Gore spomenute prednosti čine da ORB-SLAM doseže najsuvremeniji vizualni SLAM sustav otvorenog koda. Mnogi istraživački radovi koriste ORB-SLAM kao standard ili nastavak razvoja. Njegov je kod poznat po tome što je jasan i lak za čitanje s komentiranim kodom, prijateljski nastrojen prema kasnijim istraživačima.

Danas postoje ORB-SLAM, ORB-SLAM2 i ORB-SLAM3, kao i mnoge izvedbe bazirane na ORB SLAM-u. ORB-SLAM3 je najnovija i najpotpunija izvedba, te je prva SLAM biblioteka u stvarnom vremenu koja može izvesti vizualni, vizualno-inercijalni i višekartni SLAM s monokularnim, stereo i RGB-D kamerama.[16] U vremenu pisanja rada jedino ORB-SLAM2 ima omotač(wrapper) za ROS2.

Naravno, i ORB-SLAM ima nekih nedostataka. Budući da cijeli SLAM sustav koristi značajke za izračun, moramo otkriti ORB značajke za svaku sliku, što je vrlo dugotrajno. ORB-SLAM-ova tro-dretvena struktura također donosi veće opterećenje za CPU, što ga čini sposobnim da radi u stvarnom vremenu samo na CPU-u modernijeg računala. Nije lako transformirati se u ugradbene uređaje. Drugo, ORB-SLAM karta se sastoji od rijetkih značajki. Rijetka karta točaka značajki može zadovoljiti samo naše potrebe za lokalizacijom, ali ne može zadovoljiti navigaciju, izbjegavanje prepreka, interakciju ili druge zahtjeve. Međutim, ako koristimo ORB-SLAM samo za rješavanje problema lokalizacije, čini se da nije jeftina metoda samo za lokalizaciju . Nasuprot tome, neki drugi programi pružaju lakšu lokalizaciju, omogućujući nam pokretanje SLAM-a na procesorima niske razine. [10]

DTAM

DTAM(Dense Tracking and Mapping)(2011) pristup bio je jedan od prvih izravnih vizualnih SLAM implementacija u stvarnom vremenu, ali se u velikoj mjeri oslanjao na GPU da bi se izveo. Uvelike pojednostavljeno, DTAM počinje uzimanjem više stereo temeljnih linija za svaki piksel sve dok se ne dobije prvi ključni kadar i stvori početna karta dubine sa stereo mjerenjima. Koristeći ovu početnu kartu, kretanje kamere između okvira prati se uspoređivanjem slike s prikazom modela generiranog iz karte. Sa svakim uzastopnim okvirom slike, informacije o dubini se procjenjuju za svaki piksel i optimiziraju minimiziranjem ukupne energije dubine. Rezultat je model s informacijama o dubini za svaki piksel, kao i procjenom položaja kamere. Budući da prati svaki piksel, DTAM proizvodi mnogo gušću kartu dubine, čini se da je mnogo robusniji u okruženjima bez značajki i prikladniji je za rješavanje različitih fokusa i zamućenja pokreta. [7] Danas ovaj projekt više nije aktivan, ali ideja direktnog VSLAM-a se nastavila dalje razvijati.

LSD-SLAM

LSD-SLAM(Large-Scale Direct Monocular SLAM) je direktna monokularna SLAM tehnika: umjesto korištenja ključnih točaka (značajki), izravno djeluje na intenzitet slike i za praćenje

i za kartiranje. Kamera se prati korištenjem izravnog poravnanja slike, dok se geometrija procjenjuje u obliku polugustih (*semi-dense*) karata dubine, dobivenih filtriranjem kroz mnoge stereo usporedbe po pikselima.[17] LSD-SLAM ostvaruje polugustu rekonstrukciju na CPU-u, što se rijetko viđa u prethodnim shemama. Budući da LSD-SLAM koristi metodu izravnog praćenja, ima i prednosti i nedostatke izravne metode. Na primjer, LSD-SLAM je vrlo osjetljiv na intrinzične parametre kamere i lako se gubi kada se kamera brzo pomiče. Također, u dijelu otkrivanja petlje, LSD-SLAM se još uvijek oslanja na metodu značajnih točaka za detekciju petlje i nije u potpunosti eliminirao izračun značajki.[10] LSD-SLAM radi čak i na pametnom telefonu, gdje se može koristiti za AR. Procijenjene poluguste karte dubine obojene su i dopunjene procijenjenom podnom površinom(ground plane), što zatim omogućuje implementaciju osnovne fizičke interakcije s okolinom.[17]

SVO-Pro

SVO(Semi direct Visual Odometry) koristi poludirektnu paradigmu za procjenu 6DOF gibanja sustava kamere iz intenziteta piksela (izravno) i značajki (bez potrebe za dugotrajnim izdvajanjem značajki i postupcima uparivanja), dok postiže bolju točnost izravnim korištenjem intenziteta piksela. SVO je svestran i učinkovit. Prvo, radi na različitim vrstama kamera, od uobičajenih projektivnih kamera do katadioptričkih. Također podržava stereo i sustav s više kamera. Stoga se može prilagoditi za različite scenarije. Drugo, SVO zahtijeva vrlo malo računskih resursa u usporedbi s većinom postojećih algoritama. Može doseći do 400 sličica u sekundi na i7 procesoru (dok uzima manje od 2 jezgre!) i do 100 fps na procesoru pametnog telefona (npr. Odroid XU4).[18]

SVO Pro proširuje SVO. SVO sam po sebi je nastao kao VO. Dodatnim radom i proširenjem ove odometrije stvoren je SVO Pro, koji implementira *global bundle adjustment* i zatvaranje petlje, te se klasificira kao VSLAM pristup. SVO Pro nudi sljedeće funkcionalnosti:

- Vizualna odometrija: najnovija verzija SVO-a koja podržava perspektivne i riblje oko/katadioptričke kamere u monokularnom ili stereo postavu. Također uključuje aktivnu kontrolu izloženosti.
- Vizualno-inercijalna odometrija: SVO *frontend* + vizualno-inercijalni *backend* optimizacije kliznog prozora[12].
- Vizualno-inercijalni SLAM: SVO *frontend* + *backend* optimizacije vizualno-inercijalnog kliznog prozora + karta optimizirana *global bundle adjustment*-om. Globalna karta se ažurira u stvarnom vremenu, zahvaljujući iSAM2[13], i koristi se za lokalizaciju brzinom kadrova.
- Vizualno-inercijalni SLAM sa zatvaranjem petlje: Zatvaranje petlje, preko DBoW2[14], integrirano je u *global bundle adjustment*. *Pose graph optimization* također je uključena kao lagana zamjena za *global bundle adjustment*.

[12]Optimizacija kliznog prozora je neka vrsta pristupa *bundle adjustment* problema. Ove tehnike uklanjaju starija stanja (značajke i/ili poze kamere) iz aktivno procijenjenog vektora stanja i provode iterativno minimiziranje kako bi se proizvele procjene za najnovija stanja.[19]

[13]iSAM2 - Incremental Smoothing and Mapping - `https://www.cs.cmu.edu/~kaess/pub/Kaess12ijrr.pdf`

[14]DBoW2 - unaprijeđena vrsta *bag-of-words* pristupa - `https://github.com/dorian3d/DBoW2`

DVO-SLAM

DVO(Dense Visual Odometry) pruža implementaciju vizualne procjene odometrije iz RGB-D slika za ROS. Za razliku od algoritama temeljenih na značajkama, pristup koristi sve piksele dviju uzastopnih RGB-D slika za procjenu kretanja kamere. Implementacija se izvodi u stvarnom vremenu na novijem CPU-u.[20]

DVO-SLAM pruža implementaciju gustog vizualnog SLAM sustava za RGB-D kamere. SLAM sustav se temelji na DVO. Svaki novi ključni kadar se umeće u graf poze. Dodatno traži zatvaranja petlje za starije ključne kadrove. Ova zatvaranja petlje pružaju dodatna ograničenja za graf poza. Graf se postupno optimizira pomoću g2o alata. Izlaz SLAM sustava su metrički dosljedne poze za sve okvire.[20]

VINS

VINS-Mono(Visual-Inertial System) je SLAM alat za monokularne vizualno-inercijalne sustave u stvarnom vremenu. Koristi formulaciju kliznog prozora temeljenu na optimizaciji za pružanje vizualno-inercijalne odometrije visoke točnosti. Sadrži učinkovitu predintegraciju IMU-a. VINS-Mono je prvenstveno dizajniran za procjenu stanja i povratnu kontrolu autonomnih dronova, ali je također sposoban pružiti točnu lokalizaciju za AR aplikacije. Ovaj kod radi na Linuxu i potpuno je integriran s ROS-om. Za iOS mobilnu implementaciju stvoren je i VINS-Mobile. Više informacija na github stranici, link je u tablici Table 2.

VINS-Fusion je proširenje VINS-Mono, koje podržava više tipova vizualno-inercijalnih senzora (mono kamera + IMU, stereo kamera + IMU, čak i samo stereo kamera). Značajke VINS-Fusion-a:

- podrška za više senzora (stereo kamere / mono kamera+IMU / stereo kamere+IMU)
- online prostorna kalibracija (transformacija između kamere i IMU-a)
- online vremenska kalibracija (vremenski pomak između kamere i IMU-a)
- zatvaranje vizualne petlje

RTABMAP

RTABMAP (Real-Time Appearance-Based Mapping) knjižnica je otvorenog koda koja implementira otkrivanje zatvaranja petlje s pristupom upravljanja memorijom, ograničavajući veličinu karte tako da se detekcije zatvaranja petlje uvijek obrađuju u fiksnom vremenskom ograničenju, čime se zadovoljava online zahtjevi za dugoročnim i velikim kartiranjem okoliša. Pokrenut 2009. i objavljen kao knjižnica otvorenog koda 2013., RTABMAP je od tada proširen na potpuni SLAM pristup baziran na optimizaciji grafa koji se koristi u različitim postavkama i aplikacijama. Kao rezultat toga, RTABMAP se razvio u višeplatformsku samostalnu C++ biblioteku i ROS paket.[21]

Neke od značajki i mogućnosti koje pruža RTABMAP su:

- Korištenje više vrsta odometrije - neovisan je o korištenom pristupu odometriji, što znači da se može koristiti vizualnom odometrijom, lidarskom odometrijom ili čak samo odometrijom kotača. To znači da se RTABMAP može koristiti za implementaciju

vizualnog SLAM pristupa, lidarskog SLAM pristupa ili mješavine oba, što omogućuje usporedbu različitih konfiguracija senzora na stvarnom robotu.[21]

- Vlastiti čvorovi za odometriju - RTABMAP implementira svoje čvorove za odometriju, tako pruža vizualnu odometriju za RGBD kamere i za stereo kamere, i ICP odometriju za LiDAR. Razlika između rgbd čvora i stereo čvora je što je stereo čvor proširen kako bi procijenio dubinu u početku. Vizualna odometrija se i za stereo kameru i za RGBD kameru se može koristiti u dvije strategije: praćenjem značajki ili direktnom metodom. Metoda praćenja značajki implementira više vrsta značajki deskriptora koji su na raspolaganju. Zadano je postavljeno korištenje GFTT(Shi Tomasi - Good Features To Track) i BRIEF deskriptor. Estimacija položaja kamere se računa pomoću PnP algoritma s RANSAC-om. Implementiran je i *local bundle adjustment*.

- Druge VO otvorenog koda - Odometrija u RTABMAP-u neovisna je o procesu mapiranja, drugi pristupi vizualnoj odometriji integrirani su u RTABMAP radi praktičnosti i jednostavnosti usporedbe između njih. Odabrani pristupi su otvorenog koda ili pružaju aplikacijsko programsko sučelje (API) i mogu se koristiti samo kao odometrija. Potpuni vizualni SLAM pristupi u kojima je teško odvojiti odometriju od procesa mapiranja ne mogu se integrirati jer se RTAB-Map brine za proces mapiranja. Sedam pristupa integrirano je u RTAB-Map: FOVIS [Huang et al., 2011.], Viso2 [Geiger et al., 2011.], DVO [Kerl et al., 2013.], OKVIS [Leutenegger et al., 2015.], ORB-SLAM2 [Mur-Artal i Tard os, 2017.], MSCKF [Sun et al., 2018.] i Google Project Tango.[21] Ovi su svi pristupi dostupni u punoj verziji RTABMAP-a, koji je trenutno dostupan samo u ROS1. Razlog tome je što sve prethodno navedene odometrije nisu prebačene u ROS2 te stoga nisu još integrirane u ROS2 verziju RTABMAP-a za vrijeme pisanja ovog rada.

- Sinkronizacija tema(topic-a) - RTABMAP pruža čvorove koji rade pretprocesiranje podataka dobivene iz RGBD kamere ili stereo kamere. Ovo pred procesiranje se koristi ako recimo RGBD kamera nema sinkronizaciju (RGB slika i depth slika nisu sinkronizirane). Ovaj čvor služi kako bi se informacije iz kamere sinkronizirale kako bi daljnja sinkronizacija s drugim senzorima bila lakša.

- Upravljanje memorijom - Kako je RTABMAP SLAM pristup s optimizacijom grafa u pozadini, upravljanje memorijom služi kako bi ograničio veličinu grafa tako da se dugoročni online SLAM može postići u velikim okruženjima. Bez upravljanja memorijom, kako graf raste, vrijeme obrade za module poput zatvaranja petlje, optimizacije grafa i globalnog sastavljanja karte može na kraju premašiti ograničenja u stvarnom vremenu. RTABMAP-ova memorija je podijeljena na radnu memoriju(*working memory* - WM) i dugoročnu memoriju(*long term memory* - LTM). Radna memorija sprema čvorove grafa i ograničenja(odometrijska, zatvaranje petlje). Informacije iz radne memorije se koriste za detekciju petlji, za optimizaciju, za kreiranje karte. Kada radna memorija dostigne određeni kapacitet, na osnovu neke strategije prebacuje dio čvorova u dugotrajnu memoriju. Čvorovi iz dugotrajne memorije se ne koriste u izračunima. Kada se dogodi zatvaranje petlje s lokacijom u WM-u, susjedni čvorovi ove lokacije mogu se vratiti iz LTM-a u WM za više zatvaranja petlje.

- Optimizacija grafa - Kada se otkrije zatvaranje petlje ili detekcija blizine ili se neki čvorovi dohvate ili prenesu zbog upravljanja memorijom, primjenjuje se pristup optimizacije grafa kako bi se greške na karti svele na najmanju moguću mjeru. RTABMAP integrira tri pristupa optimizacije grafa: TORO [Grisetti et al., 2010], g2o [Kummerle et al., 2011] and GTSAM [Dellaert, 2012].[21]

- kartiranje - RTABMAP može kreirati i 2D karte popunjenosti i 3D guste karte.

- GUI - RTABMAP ima prilagođeno grafičko sučelje za jednostavnije korištenje sustava. Također RTABMAP šalje teme u ROS sustav što omogućuje korištenje informacija dobivenih RTABMAP-om i u drugim ROS aplikacijama(npr. Rviz).

Tablica 3: Značajke VSLAM rješenja

NAZIV	ROS	VO pristup	backend*	gustoća karte
MonoSLAM	1	usporedba predložaka	EKF	rijetka
LSD-SLAM	1	direktna metoda	GO	polu-gusta
DTAM	1	direktna metoda	GO	gusta
SVO-Pro	1	hibridna metoda	GO	rijetka
DVO-SLAM	1	direktna metoda	GO	gusta
VINS-Mono	1	izvlačenje značajki	GO	rijetka
VINS-Fusion	1	izvlačenje značajki	GO	rijetka
S-PTAM	1	izvlačenje značajki	GO	rijetka
ORB-SLAM2	1/(2)	izvlačenje značajki	GO	rijetka
ORB-SLAM3	1	izvlačenje značajki	GO	rijetka
RGBD-SLAM-V2	1	razni pristupi	GO	gusta
RTAB-MAP	1/2	razni pristupi	GO	gusta

*backend: EKF - Extended Kalman FIlter; GO - graph optimization

6 PREDLOŽENA KONFIGURACIJA SUSTAVA ZA KARTIRANJE POLJOPRIVREDNOG ZEMLJIŠTA

U prethodnim poglavljima opisan je VSLAM problem, te su dani neki od pristupe rješavanja istog. Zatim je opisana Intel RealSense D455 dubinska kamera s aktivnim stereo-om. Ova kamera je odabrana za izradu ovog rada, kako je bila u tom vremenu dostupna na fakultetu. Zatim su u petom poglavlju opisani neki od VSLAM rješenja otvorenog koda koji su dostupni u ROS okruženju. ROS okruženje je odabrano jer se koristilo na našem fakultetu kroz kolegije, a također ima dobru podršku za Intel RealSense kamere. Usporedba nekih od značajki VSLAM rješenja mogu se vidjeti u tablicama u prethodnom poglavlju Table 2-3. Svaki pristup ima svoje prednosti i mane, te ovisno koji senzor koristimo, koliku računsku snagu koristimo i koji zadatak treba obaviti, odlučujemo se za pojedino rješenje.

Zadatak rada je bila istražiti postojeća rješenja za kartiranje u stvarnom vremenu uz uporabu Intel RealSense D455 (RGBD) kamere u ROS okruženju. U dogovoru s profesorom, težnja je bila na novijim rješenjima, po mogućnosti da se mogu izvoditi na novijim verzijama ROS-a (ROS2). Iz tog razloga se kao operacijski sustav odabrao Linux Ubuntu verzija 20.04. *Focal Fossa* i ROS2 verzija *Foxy Fitzroy*. Od istraženih rješenja, ona koja koriste RGBD kameru su DVO-SLAM, ORB-SLAM2, ORB-SLAM3, RGBD-SLAMv2, RTABMAP. Od nabrojanih pristupa pronašao sam da samo ORB-SLAM2 i RTABMAP imaju omotač za ROS2.

Moja odluka za rješenjem je bila RTABMAP iz razloga jer je do dana pisanja ovog rada održavan, ima mogućnost korištenja u ROS1 i ROS2(nema sve funkcionalnosti kao u ROS1), pruža grafičko sučelje za lakše korištenje, jako je dobro dokumentiran s mnoštvom primjera i za razliku od ORB-SLAM2 omogućuje korištenje više vrsta odometrije, kao i generacija guste 3D karte prostora. Također ima primjera korištenja s Intel RealSense serijom kamera. Kako RTABMAP rješenje zadovoljava sve kriterije rada, njega sam odabrao za testiranje.

Ovime završava faza istraživanja i odabira rješenja. U nastavku rada slijedi predlaganje konfiguracije te testiranje odabranog rješenja.

6.1 Instalacija programske podrške

Sva programska podrška je podržana na Ubuntu verziji 20.04., te se može instalirati na sljedećim poveznicama:

- ROS 2 Foxy Fitzroy
  (`https://docs.ros.org/en/foxy/Installation.html`)
- RealSense SDK 2.0
  (`https://github.com/IntelRealSense/libRealSense/blob/master/doc/distribution_linux.md`)
- Napraviti mapu za ros2 okruženje, u njoj mapu src/ i sljedeće pakete tu preuzeti (meni izgleda ovako /ros2_ws/src)
- RealSesne SDK wrapper za ROS2
  (`https://github.com/IntelRealSense/RealSense-ros/tree/ros2`)
- RTABMAP for ROS 2 Foxy
  (`https://github.com/introlab/rtabmap_ros/tree/foxy-devel`)

- imu_tools (dok sam radio projekt nije postojala verzija za ROS2 Foxy, jedina verzija za ROS2 je bila za Eloquent, i meni je super radilo. Samo se klonira kod u /ros2_ws/src i build-a. NAPOMENA: ova je knjižnica obavezna ako se želi koristiti IMU data.) (`https://github.com/ccny-ros-pkg/imu_tools/tree/eloquent`)

- robot_lokalization (močna knjižnica za fuziju senzora) (`https://github.com/cra-ros-pkg/robot_localization/tree/foxy-devel`)

Kada je sve instalirano, sada se treba pozicionirati u ros2 workspace mapu (ros2_ws u mom slučaju) i pokrenuti komandu:

```
$ colcon build
```

Nakon build-a *source*-at bash:

```
$ . install/setup.bash
```

Sada je sve spremno, u nastavku ćemo se fokusirati na određene launch datoteke i upoznati se s određenim parametrima koji su nam bitni. Dva najvažnija su:

- `WORKSPACEPATH/src/RealSense-ros/RealSense2_camera/launch/rs_launch.py` (za pokretanje Intel RealSense kamere)

- `WORKSPACEPATH/src/rtabmap_ros/launch/ros2/rtabmap.launch.py` (Za pokretanje RtabMap-a)

6.2 Priprema i konfiguracija Intel Realsense D455 kamere

Podešavanje parametara kamere uvelike utječe na izlazne podatke koja nam ona daje. Detaljan pregled parametara i njihovo podešavanje moguće je pronaći u RealSense alatu, koji je prethodno spomenut, RealSense Viewer-u.

Parametri koje treba prilagoditi su oni u launch datoteci koja pokreće kameru. Parametri se mogu ili ručno mijenjati u datoteci ili prilikom pokretanja komande u nastavku komande. Komanda s prilagođenim parametrima izgleda ovako:

```
$ ros2 launch RealSense2_camera rs_launch.py \
        enable_color:=true \
        color_width:=-1 \
        color_height:=-1 \
        color_fps:=60. \
        enable_depth:=true \
        depth_width:=-1 \
        depth_height:=-1 \
        depth_fps:=90. \
        enable_gyro:=true \
        enable_accel:=true \
        unite_imu_method:=copy \
        gyro_fps:=400. \
        accel_fps:=250. \
        enable_infra1:=false \
        enable_infra2:=false \
        infra_width:=848 \
        infra_height:=480 \
        infra_fps:=60. \
```

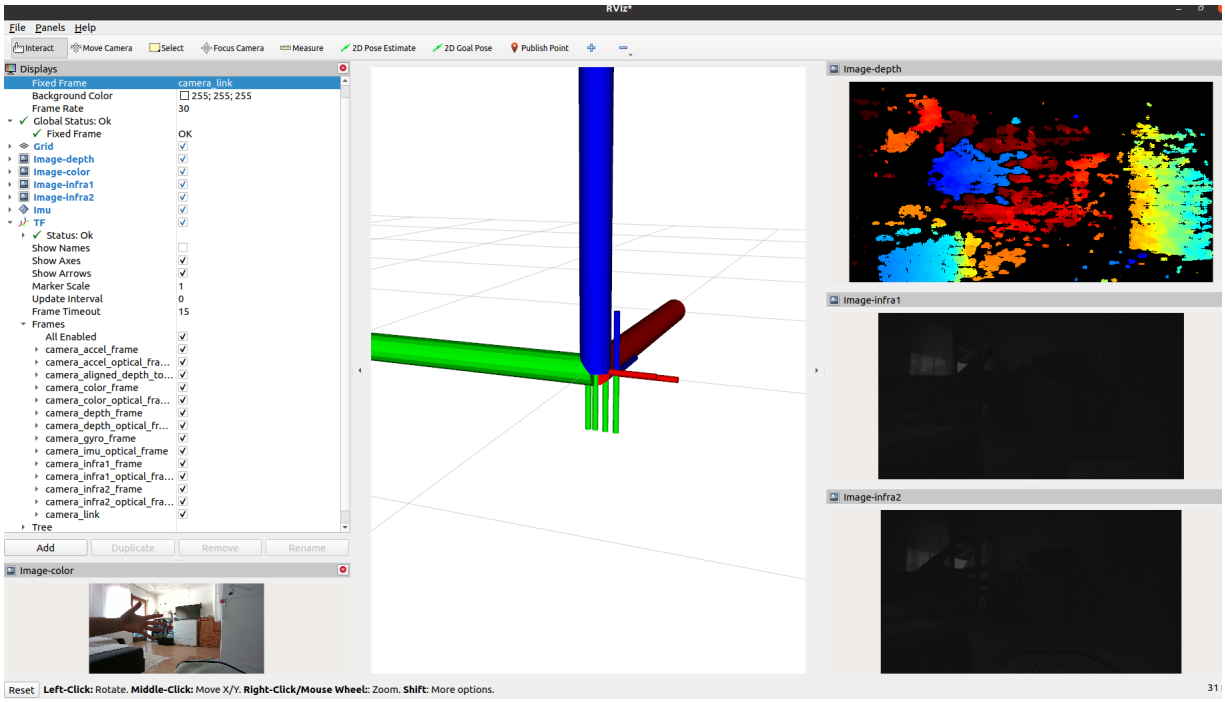
```
enable_sync:=false \
align_depth:=true \
enable_pointcloud:=false \
initial_reset:=true
```

Listing 1: Komanda za pokretanje kamere

Na ovaj nam je način omogućeno pokretanje svake leće kamere, njene rezolucije i fps(frames per second - slika u sekundi), kao i nekih unutarnjih funkcionalnosti kamere općenito. Parametre kao što su *width*, *height*, *fps* mogu se postaviti na neku od vrijednosti koje kamera podržava. Vrijednost -1 označava najveću moguću vrijednost. Listu mogućih vrijednosti mogu se pogledati s komandom u terminalu:

```
$ rs-enumerate-devices
```

Intel RealSense D455 kamera ima ugrađen IMU, način kako dobit podatke iz IMU-a opisano je u podcjelini 6.3.



Slika 14: ROS rviz sucelje sa sa prikazanom RGB (dole lijevo), dubinskom (gore desno) i infra crvenim slikama (dole desno) kamere, te položajem kamere (sredina)

Na slici Slika 14 mogu se vidjeti prikaz RGB senzora kamere (dole lijevi kut), dubinsku sliku (gore desni kut), i dvije infracrvene kamere(desno u sredini i desno dole). Iz priloženog se da zaključiti kako su ustvari skroz lijevi i skroz desni senzor na Intel RealSense D455 kameri ustvari infracrveni senzori. Infra1 senzor je lijevi senzor, dok je infra2 desni senzor. Na slici Slika 14 ako se pogleda na sličice koje daju infracrvene kamere može se vidjeti da se "više" ruke vidi na infra1 slici nego na infra2 slici.
Kako bi se postigla ovakva dubinska slika u boji(desno gore na slici Slika 14) potrebno je pri pokretanju kamere postaviti parametar $\{filters := colorizer\}$.
NAPOMENA: Ovaj parametar "*filters*" utječe na stream dubinske slike i mijenja ga. To je dobro za vizualizaciju, ali nije pogodno za korištenje kao dubinske slike u RTABMAP-u. Ne koristiti filter za RTABMAP!
IMU je fiksiran. Ovo je prikaz IMU-a kamere, kako on nema orijentaciju postavljen je u početnu poziciju.

6.3 Konfiguracija inercijalne mjerne jedinice

Kako bismo koristili IMU koji nam je na raspolaganju kod Intel RealSense D455 kamere u ROS-u, potrebno je pri pokretanju launch datoteke za kameru postaviti određene parametre. Kao što je prethodno navedeno to su parametri:

1. *enable_gyro* i *enable_accel*
2. *gyro_fps* i *accel_fps*
3. *unite_imu_method*

Žiroskop i akcelerometar su dva senzora koja rade svaki na svojoj frekvenciji. Moguće frekvencije za žiroskop su $\{200, 400\}$, dok su moguće frekvencije za akcelerometar $\{63, 250\}$. Kako ova dva senzora rade zasebno, tako i njihova mjerenja dolaze zasebno, svaki senzor u svome topic-u(temi)[15]. Mi bi htjeli da dobijemo mjerenja i žiroskopa i akcelerometra spojena u jednoj temi (IMU topic). Postavljanjem parametra *unite_imu_method* dobiva se upravo to. Umjesto dvije zasebne teme za svaki senzor, dobije se jedna koja spaja mjerenja senzora. *Unite_imu_method* možemo definirati na jednu od dvije moguće metode:

- *copy*
  (Svako žiroskopska poruka je priložena uz zadnju akcelerometrijsku - kopirana)
- *linear_interpolation*
  (Svaka žiroskopska poruka je priložena uz akcelerometrijsku, koja je interpolirana na vremensku oznaku žiroskopa.)

Frekvencija spojene poruke (Žiroskopske i akcelerometrijske) iznosi približno frekvenciji na kojoj radi žiroskop. Neovisno koju smo metodu za *unite_imu_method* odabrali, frekvencija poruka će biti vezana za frekvenciju na kojoj radi žiroskop. Spojena poruka sada sadrži informaciju o linearnom ubrzanju i o angularnoj brzini. No i dalje ovoj poruci nedostaje informacija o orijentaciji. Jedan od rješenja je korištenje alata *imu_tools*.

IMU tools

Imu_tools je ROS paket/knjižnica koja uzima informacije o žiroskopu i akcelometru i računa orijentaciju uređaja. Imu_tools sadrži 2 filtera na raspolaganju:

- Complementary filter
- Madgwick filter

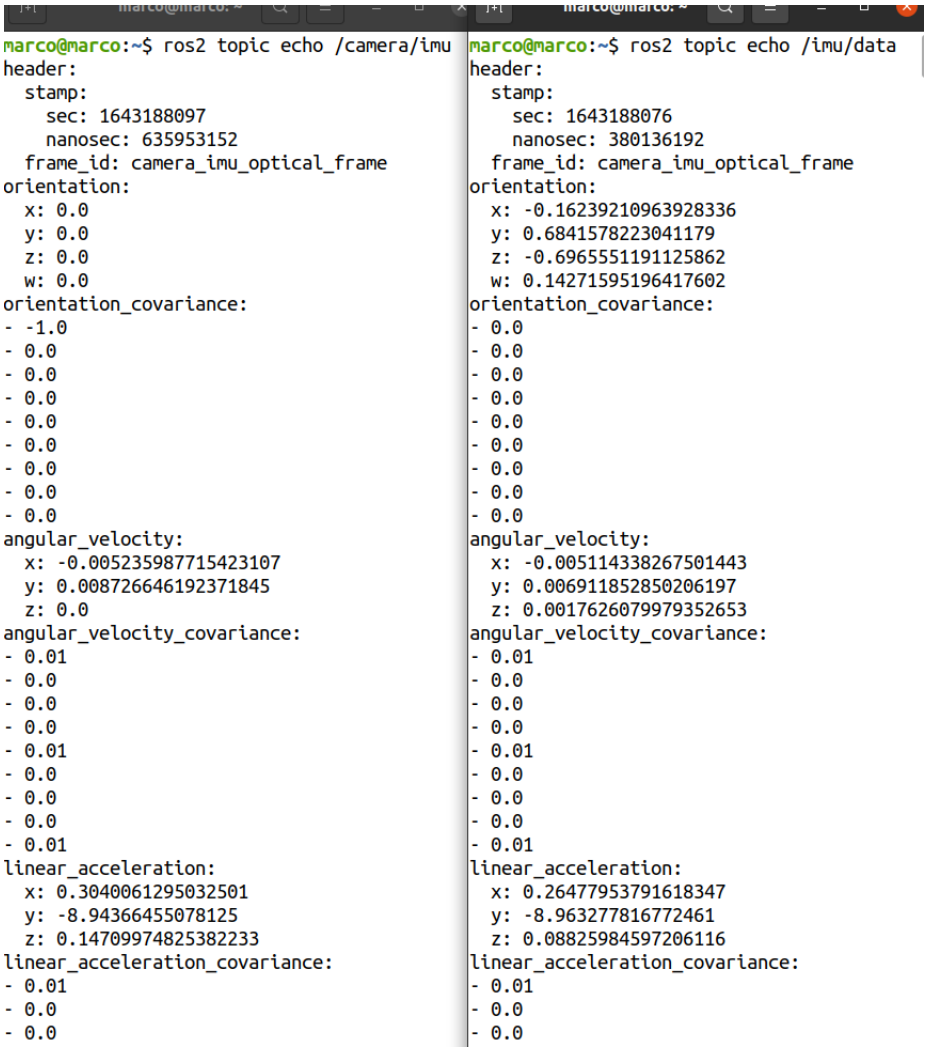
```
$ ros2 run imu_complementary_filter complementary_filter_node \
    --ros-args --remap /imu/data_raw:=/camera/imu

$ ros2 run imu_filter_madgwick imu_filter_madgwick_node \
    --ros-args --remap /imu/data_raw:=/camera/imu \
    --param use_mag:=false
```

Listing 2: Komande za pokretanje IMU filtera

[15] https://github.com/IntelRealSense/RealSense-ros/tree/ros2

Kamera kada se pokrene s uključenim IMU-om i *unite_imu_method* ona šalje poruku na topic pod nazivom **/camera/imu**. *Imu_tools* filteri se preplaćuju na topic pod nazivom **/imu/data_raw**. To je zadano od knjižnice. Sve što je potrebno je promijeniti topic na kojeg se filter pretplaćuje. Komande koje pokreću IMU filtere i postavljaju topic na kojeg se filteri pretplaćuju su prikazane u Listing 2. Izlaz iz filtera nam vrača istu vrstu poruke *sensor_msgs/msg/Imu*, ali s izračunatom orijentacijom, pogledaj Slika 15.



Slika 15: Usporedba IMU poruke bez filtra(lijevo) i sa filtrom(desno)

Da se primijetiti u Slika 15 kako topic **/camera/imu** koji dolazi iz kamere, nema izračunatu orijentaciju. Nadalje filter se pretplaćuje na topic **/camera/imu** izračunava orijentaciju, kombinira ju s porukom koju je pročitao iz **/camera/imu** i šalje u novi topic pod nazivom **/imu/data**. Ovako potpuna IMU poruka spremna je za korištenje.

6.4 Intel RealSense kamera i RTABMAP

Ideja ovog projekta bila pronaći i istestirati jedno od postojećih metoda za trodimenzionalno kartiranje te ga primijeniti u vanjskoj okolini. Kao što je do sad spomenuto isprobat će se kartiranje RTABMAP rješenjem. Do sada smo uspjeli pokrenuti Intel RealSense D455 kameru u ROS2 okruženju. Uspjeli smo pokrenuti kamerin IMU, i pokrenuli filter za računanje orijentacije istog. Sljedeći korak je povezati kameru sa RTABMAP-om. Prvo pitanje je što se točno mora povezati? Kako bi RTABMAP algoritam mapirao potrebno mu je dati informaciju o vizualnoj odometriji ili odometriju nekog drugog senzora kao sto je enkoderi kotača ili LiDAR. Kako smo za ovaj projekt odlučili koristiti kameru, jedno od mogućih pristupa problemu je korištenje rgbd odometrije ili stereo odometrije i tu informaciju proslijediti RTABMAP-u za kartiranje. RTABMAP je specijaliziran za kartiranje kamerama te ima svoje rješenje za dobivanje odometrije iz vizualnih podataka. RTABMAP je implementirao nekoliko metoda za računanje odometrije iz slike i LiDAR-a, a to su:

(a) RGB-D odometrija

(b) Stereo odometrija

(c) ICP odometrija

a) RGB-D odometrija pronalazi kretanje kamere iz dvije uzastopne RGB-D slike. Traži značajke u slici te prati njihov pomak u sljedećoj slici, tako prati kako se kamera kreće. RGB-D senzor se koristi kod ove metode dobivanja odometrije. Kako je Intel RealSense D455 kamera RGB-D senzor, ovo je osnovna metoda na koju čemo obratiti pažnju.

b) Stereo odometrija koristi kalibrirani set kamera za kartiranje. Ideja je metodom triangulacije pomoću više kamera izračunati dubinu prostora. Prethodno navedeni RGB-D senzori računanje dubine obavljaju direktno u senzoru.

c) Iterative Closest Point (ICP) odometrija je odometrija dobivena ICP algoritmom. ICP algoritam pokušava minimizirati razliku između dvaju Pointclouda.

Kada odlučimo koju metodu dobivanja odometrije želimo koristiti, potrebno je povezati topic-e kamere i algoritma za dobivanje odometrije, te dobivenu odometriju dalje povezati sa RTABMAP algoritmom za kreiranje 3D karte. U nastavku ćemo govoriti kako povezati topic-e kamere s algoritmom za dobivanje odometrije iz slikovnih ulaznih podataka. Te kako se dobivena odometrija povezuje sa RTABMAP algoritmom za kartiranje. Nadalje ćemo spomenuti neke od parametara RTABMAP-a, koji se mogu podešavati po potrebi.

6.4.1 Pokretanje RTABMAP čvorova

Jednom kada pokrenemo kameru s komandom prikazanom u Listing 1, dobivamo na raspolaganje topic-e koja nam pruža kamera. Za provjeru topic-a koristi se komanda:

```
$ ros2 topic list
```

Najbitniji od njih, oni koji će se koristiti od strane RTABMAP-a su:

```
/camera/aligned_depth_to_color/camera_info
/camera/aligned_depth_to_color/image_raw
/camera/color/camera_info
/camera/color/image_raw
/camera/imu
```

Iako RTABMAP dolazi s prethodno napisanom launch datotekom za Intelovu seriju kamera D400, samostalno pokretanje te launch datoteke nije mi dalo željeno rješenje. Parametar koji je bio izostavljen je bio parametar queue_size, njegovim povećanjem na vrijednost veću od 100, RTABMAP počinje raditi, barem u mom slučaju. Sljedeća komanda pokreće RTABMAP koji povezuje topic-e naše kamere:

```
$ ros2 launch rtabmap_ros rtabmap.launch.py \
    args:="-d" \
    frame_id:=camera_link \
    rgb_topic:=/camera/color/image_raw \
    depth_topic:=/camera/aligned_depth_to_color/image_raw \
    camera_info_topic:=/camera/color/camera_info \
    approx_sync:=false \
    queue_size:=200
```

Listing 3: Komanda za pokretanje RTABMAP-a

Komanda prikazna u Listing 3 pokreće 3 čvora(node-a):

- RGB-D Odometry node
  Čvor koji prima kao ulaznu informaciju stream RGB-D slika, te na osnovu njih računa odometriju, kretanje kamere u prostoru.

- RTABMAP node
  Čvor koji prima kao ulaznu informaciju također stream RGB-D slika i odometriju, i na osnovu tih informacija ispravlja globalnu pozu kamere i gradi mapu prostora.

- RTABMAPViz node
  Čvor koji pokreče grafičko sučelje RTABMAP-a. U tom sučelju se može pratiti generiranje 3D , preuzimanje , post-processing, podešavanje parametara, ponovno pokretanje snimljene sesije i mnoge druge stvari koje pomažu pri radu s RTABMAP-om. Nije nužno pokretanje ovog čvora, sva je vizualizacija moguća i u rvizu.

RTABMAP osim samo informacije o slici, može primat i druge vrste senzora za poboljšanje mapiranja kao što su GPS, laserscan, IMU. Iako i bez pružanja dodatnih senzora, RTABMAP može mapirati samo i s do sada postavljenim postavkama. Ovo je najosnovnija primjena RTABMAP-a s Intel RealSense D455 kamerom. Na ovaj se način još ne koristi IMU, njegovo korištenje opisat će se u nastavku.

6.4.2 Dodatni senzori

Ako se želi povezati i IMU od kamere sa RTABMAP-om, potrebno je:

- pokretanje kamere s komandom u Listing 1
- pokretanjem jednog od IMU filtera jednom od komandi u Listing 2
- pokretanje RTABMAP node-a komandom u Listing 3
  NAPOMENA: potrebno je dodati pri pokretanju RTABMAP-a sljedeću liniju u komandu za pokretanje: { *wait_imu_to_init:=true* }

RTABMAP je zadan tako da se pretplaćuje na IMU topic pod imenom **/imu/data**, a kako filter šalje IMU podatke baš na taj topic tu se nema što mijenjati. Također je bitno napomenuti pošto se koristi IMU od kamere a sve transformacije između koordinatnog sustava kamere (*camera_link* (*base_link*)) i koordinatnog sustava IMU-a, nije potrebno niti raditi transformacije manualno, jer se pri pokretanju kamere sve transformacije objavljuju s kamerom.

Kako RTABMAP koristi IMU podatke kad kartira, na koji način radi fuziju između RGB-D odometrije i IMU podataka, nisam uspio puno pronaći. Izgleda da se IMU podatci pri zadanom kartiranju koriste samo prilikom inicijalizacije kako bi se kartiranje počelo sa saznanjem o smjeru gravitacijske sile [16]. Čak se spominje kako se vrši *labavo* spajanje IMU i VO [17]. Postoji mogućnost da se koriste drugačije strategije prilikom računanja RGB-D odometrije, po dokumentaciji se može postaviti strategija da uključuje podatke dobivene iz IMU-a (*Odom/Strategy=6*) [18], kao što je i navedeno u RTABMAP poglavlju 5 ovog rada. Daljnjim istraživanjem sam doznao kako se RTABMAP i ostali pristupi moraju build-at *"from source"* kako bi bile omogućene sve moguće strategije računanja vizualne odometrije s RTABMAP-om [19]. To znači da u vremenu pisanja rada ROS2 verizija RTABMAP-a nema sve one podrške za razne strategije odometrije koje su i opisane u podcjelini 5. I dalje je moguće povezati odometriju kamere i IMU manualno uz pomoć alata za fuziju senzora koji se zove *robot_localization*. Ako se žele povezati dodatni senzori, rtabmap još uz IMU, može primiti i laserscan topic kao i GPS topic, neće biti tema ovog projekta. Način za napraviti fuziju između više senzora je spomenuti *robot_localization* paket. U nastavku više i o tome.

6.5 Konfiguracija RTABMAP čvorova

Dvije dostupne strategije za računanje vizualne odometrije koje RTABMAP nudi iako nije build-an *"from source"* su:

- 0 - F2M (Frame to Map (zadano))
  - svaki se novi prozor pokušava upariti sa globalnom mapom
- 1 - F2F (Frame to Frame)
  - svaki se novi prozor pokušava upariti s prethodnim prozorom

[16] https://github.com/introlab/rtabmap_ros/issues/733

[17] https://answers.ros.org/question/396271/rtabmap_ros-generates-a-distorted-map-when-using-the-filtered-odometry-by-robot_localization-ekf-node/

[18] https://github.com/introlab/rtabmap/issues/371

[19] https://github-wiki-see.page/m/Kapernikov/tech-session-visual-odometry/wiki/Tuning-and-Experimenting-with-Different-Algorithms-and-Parameters

Kako bi se mijenjalo između različitih strategija računanja vizualne odometrije, potrebno je promijeniti parametar pod nazivom *Odom/Strategy* u RGB-D odometry node-u. Kako ovako preuzet RTABMAP ima samo 2 strategije na raspolaganju i odnosi se na dvije strategije koje na različit način prilažu nove prozore. Ova vrijednost može biti samo 0 (F2M) ili 1 (F2F).

Zatim u odnosu na procesiranje slike RTABMAP nudi 2 pristupa vizualnoj odometriji kao što su:

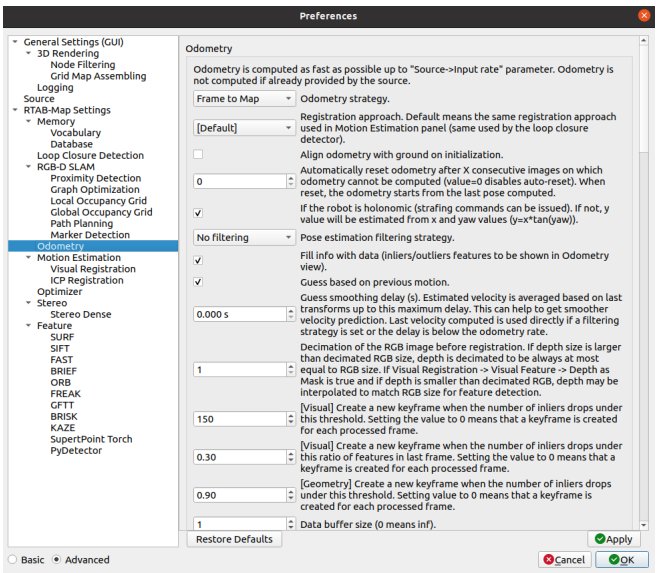
- 0 - Feature matching (zadano)
- 1 - Optical flow

Mijenjanje između pristupa moguće je promjenom parametra *Vis/CorType*. Ova vrijednost može biti 0 (FM) ili 1 (OF).

Kad je riječ o vrsti značajki, RTABMAP implementira više mogućnosti koje su na raspolaganju. Promjena vrste značajki koje se koriste postiže se promjenom parametra *Vis/FeatureType*. Kad je RTABMAP build-an "*from source*" ovih je vrsta još i više. Što se tiče trenutne verzije ROS2 RTABMAP nudi sljedeće vrste značajki:

- 2 - ORB
- 7 - BRISK
- 8 - GFTT+ORB
- 9 - KAZE
- 10 - ORB OKTREE

Mnoštvo parametara je moguće podesiti u RTABMAP-u, lista parametara se može vidjeti preko ROS alata "*rqt*", upisivanjem komadi *ros2 run rtabmap_ros rtabmap –params* ili *ros2 run rtabmap_ros rgbd_odometry –params*. Također logično posloženi parametri se mogu provjeriti i podesiti iz GUI aplikacije rtabmapviz. Prikaz prozora za podešavanje parametara u rtabmapviz-u može se vidjeti na slici Slika 16.



Slika 16: Prozor rtabmapviz-a za podešavanje parametara

6.6 Konfiguracija čvora za estimaciju položaja

Robot_localization je ROS paket za fuziju senzora. Sastoji se od kolekcije čvorova za procjenu pozicije. Svaki čvor implementira ne-linearnog procjenitelja pozicije za robote koji se kreču u 3D prostoru. *Robot_localization* se sastoji od dva čvora za procjenu pozicije:

- *ekf_localization_node* (Extended Kalman Filter)
- *ukf_localization_node* (Unscented Kalman Filter)

te jedan čvor za koji služi kao podrška za GPS podatke:

- *navsat_transform_node*

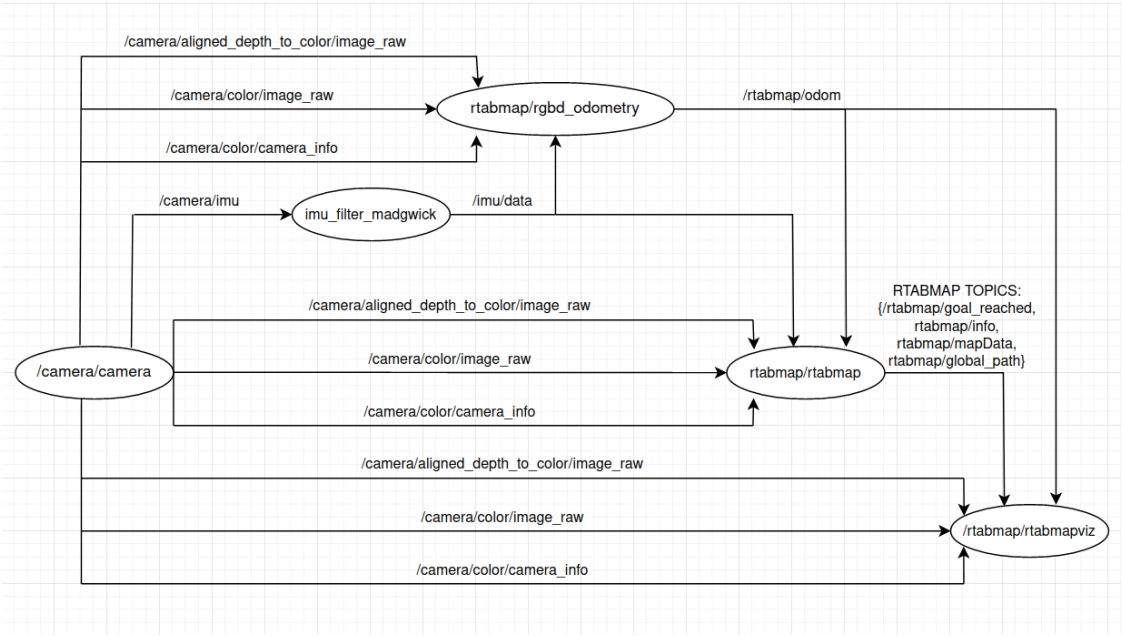
Oba čvora za procjenu pozicije pružaju podršku za proizvoljan broj senzora, čak podržavaju više senzora iste vrste(više IMU-a istovremeno). Topic-e koje podržavaju su:

- Odometry (nav_msgs/Odometry)
- IMU (sensor_msgs/Imu)
- Pose (geometry_msgs/PoseWithCovarianceStamped
- Twist (geometry_msgs/TwistWithCovarianceStamped)

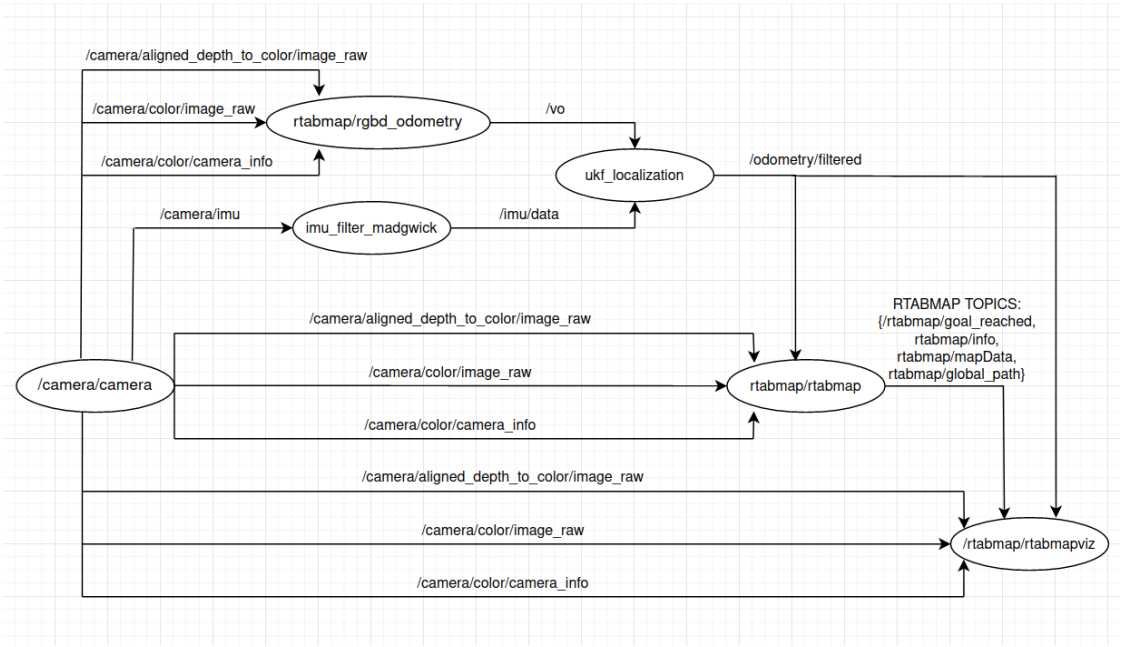
Robot_localization nam omogućuje da filtriramo koje podatke od kojeg senzora posebno želimo uključiti u estimaciju pozicije. Što nam daje veću kontrolu i tako možemo prilagoditi rješenje našem specifičnom problemu. Uz to, povratna informacija koju dobivamo u obliku odometrije, je kontinuirana. Estimator procjenjuje stanje od kad prvi put dobije ulazni podatak, a nakon toga ako i dođe do kašnjenja podataka iz senzora, estimator procjenjuje stanje na osnovu unutarnjeg modela kretanja.

U našem slučaju, senzore koje mi želimo spojiti(napraviti fuziju senzora) su vizualna odometrija dobivena sa *rgbd_odometry* od RTABMAP-a, te IMU od kamere. Dakle moramo pokrenut kameru s njenim IMU-om i IMU filter za računanje orijentacije IMU-a. Zatim moramo pokrenuti *rtabmap/rgbd_odometry node* i dat mu vizualne podatke kamere. On nam vrača odometriju iz dobivenih podataka. Tu odometriju kao, kao i IMU ne šaljemo u *rtabmap/rtabmap node* još, već šaljemo jednom od čvorova za estimaciju pozicije iz *robot_ localization* paketa. On uz pomoć Kalmanovog filtra radi estimaciju pozicije na osnovu dobivenih podataka od senzora. Čvor za estimaciju pozicije zatim vrača kontinuiranu procijenjenu odometriju. Tu odometriju dalje prosljeđujemo u *rtabmap/rtabmap node*, kao i vizualne podatke od kamere da izvrši kartiranje. Kartitanje se može vizualno pogledati u *rtabmapviz*-u ili *rviz*-u.

Na sljedeće dvije slike može se vidjeti prikaz ROS node-ova kada se koristi sam RTAB-MAP (bez paketa *robot_localization*) na Slika 17, te grafički prikaz čvorova kada se koristi *robot_ localization* paket na Slika 18



Slika 17: Prikaz ROS čvorova samo s RTABMAP-om, bez robot_localization-a



Slika 18: Prikaz ROS čvorova sa robot_localization-om

Ideja kako povezati *robot_localization* s kamerom i RTABMAP-om je opisana u prethodnoj cjelini. U ovoj će biti opisani konfiguracijski parametri čvora za estimaciju pozicije. Svejedno je jeli riječ o EKF ili UKF node-u, oba imaju identičan način konfiguracije. Konfiguracijom parametara omogućeno je prilagoditi čvor za estimaciju pozicije našem specifičnom problemu. Većina parametara se odnosi na kontrolu ulaznih podataka, priprema podataka prije prolaza kroz filter. Kako je naš problem 3D prirode, prvi parametar kojeg je potrebno postaviti u čvoru za estimaciju pozicije je parametar:

```
"two_d_mode": False,
```

Nadalje potrebno je definirati okvire(frames, koordinatne sustave).

```
"odom_frame": 'odom',
"base_link_frame": 'camera_link',
"map_frame": 'odom',
```

Ovi parametri definiraju "način rada" za *robot_localization*. REP-105 specificira tri glavna koordinatna okvira: map, odom i base_link:

- Okvir *base_link* je koordinatni okvir koji je pričvršćen za robota.
- Položaj robota u okviru *odom* će se vremenom mijenjati, ali je kratkoročno točan i trebao bi biti kontinuiran.
- Okvir *map*, kao i okvir *odom*, je koordinatni okvir fiksiran u svijetu, i iako sadrži globalno najtočniju procjenu položaja vašeg robota, podložan je diskretnim skokovima.

Definiranje senzora u čvoru za estimaciju pozicije radi se tako da se za svaki senzor definira parametar ovisno o vrsti poruke koju taj senzor daje (odomN, imuN, poseN, twistN, (N=0,1,2,3,...)). Ako želimo koristiti fuziju 2 senzora odometrije onda moramo definirati parametre odom0 i odom1. Senzori se definiraju sekvencijalno, bez preskakanja. Vrijednost koju novodefinirani parametar za senzor prima je naziv teme tog senzora. U našem slučaju mi imamo jedan senzor za odometriju (Vizualna odometrija) i jedan senzor za IMU. Definiramo sljedeće parametre:

```
"odom0": '/vo' #rgbd_odometry output
"imu0": '/imu/data'
```

Za svaku od gore definiranih poruka senzora, trebalo bi se odrediti koje varijable tih poruka trebaju biti spojene u konačnu procjenu pozicije. To se vrši preko definiranja parametara (odomN_config, imuN_config, ... (N=0,1,2,3,...)). X_config parametar je polje od 15 boolean varijabli, te nam je omogućeno manualno određivanje koje vrijednosti pojedinog senzora želimo uključiti u estimaciju pozicije. Ovako izgleda i koje vrijednosti po redu predstavlja polje X_config:

$$X_config : [X, Y, Z, roll, pitch, yaw, \dot{X}, \dot{Y}, \dot{Z}, \dot{roll}, \dot{pitch}, \dot{yaw}, \ddot{X}, \ddot{Y}, \ddot{Z}]$$

Iznad navedeni parametri su osnovni parametri za definiranje čvora filtera za estimaciju položaja spajanjem senzora. Uz to postoje dodatni parametri kojim se dodatno prilagođava filter specifičnom problemu i sustavu kojeg koristimo. U ovom radu se neće ulaziti u detalje ovog pristupa. U nastavku slijedi testiranje RTABMAP-a te komentiranje rezultata.

7 TESTIRANJE I REZULTATI

Do sada je opisano kako se postavi Intel RealSense D455 kamera u ROS2 okruženju sa RTABMAP-om. U ovom će se poglavlju prikazati rezultati testiranja. Testiralo se samostalno rješenje RTABMAP-a u zatvorenom prostoru kao i u vanjskom prostoru. Zatvoreni prostor u kojem se testiralo bilo je u stanu u kojem živim, radi jednostavnosti testiranja. Testiranje vani izvodilo se na više lokacija. Prva lokacija je oko parkinga za bicikle u gradu u kojem živim. Druga lokacija je oko kompleksa objekata u istom gradu.

Rezolucija kamere i broj slika u sekundi

Intel RealSense D455 omogućuje prikaz slike u različitim rezolucijama, pogledaj potpoglavlje 6.2. Kada bi pogledali mogućnosti koje su dostupne vidjeli bismo da pri maksimalnoj rezoluciji za RGB sliku (1280x800) moguće je postići 30 slika u sekundi, kao i za dubinsku (1280x720). Smanjenjem rezolucije RGB slike (848x480) moguće je postići 60 slika u sekundi dok za dubinsku iste rezolucije moguće je postići 90. Daljnjim smanjivanjem rezolucije ne povećava se značajno ili bitno količina slika u sekundi. Bolja rezolucija slike dovodi do mogućnosti pronalaska više značajki u slici što pomaže za bolje lokaliziranje, ali zahtjeva i više računanja. Za manju rezoluciju vrijedi obratno. Količina slika u sekundi isto igra veliku ulogu, jer se bolje prati tranzicija između 2 položaja kamere. Ja sam se odlučio za rezoluciju (848x480 i 60fps), daje dobar kompromis između rezolucije i slika po sekundi.

RTABMAP-ova vizualna odometrija

Za početak bih prokomentirao RTABMAP bez korištenja IMU-a i fokusirao se na korištenje RTABMAP-a uz korištenje VO koje pruža sa RGBD kamerom kao što je D455 koju koristimo. Vizualna odometrija koju pruža RTABMAP za RGBD kamere, te korištenje tako dobivene unutar RTABMAP čvora za optimizaciju i kartiranje, daje dobre rezultate. Omogućuje se kretanje kamere sa 6DoF, što omogućuje kartiranje u 3D svijetu. RTABMAP ima pristup detekcije i zatvaranja petlje kojeg koristi za ispravljanje nastale greške kroz vrijeme. Uz to kartiranje koje radi je gusto, te se može dobiti gusta karta prostora.

Kako vizualna odometrija pati od scenarija s malo značajki može doći do gubitka odometrije. RTABMAP ima i za takve stvari rješenje. Naime kada dođe do gubitka odometrije moguće je odometriju resetirati, što stavlja do sad kreiranu kartu u pozadinu i krene kartiranje iz početka, kada dođe do zatvaranja petlje između karte u pozadini i nove karte, RTABMAP spaja karte i lokalizira se te nastavlja s ponovno pokrenutom odometrijom. Odometrija se može resetirati na više načina:

- Iz rtabmapviz GUI aplikacije (Detection/Reset odometry)
- Pozivanjem servisa kroz komandnu liniju

```
$ ros2 service call /rtabmap/reset_odom std_srvs/srv/Empty
```

- Postavljanje RTABMAP parametra za automatski reset odometrije kada se izgubi (parametar *Odom/ResetCountdown*=1 u čvoru *rtabmap_ros/rgbd_odometry*)

NAPOMENA: kada dođe do gubitka odometrije, savjet je postaviti kameru da gleda u scenu u kojoj ima značajki, negdje gdje će se moći pronaći kada se odometrija resetira, te ju zatim resetirati. Ako smo se izgubili na jednoličnom prizoru, resetiranjem odometrije nalazimo se na istom u kojem ne možemo i dalje pratiti kretanje kamere, niti se može prepoznat petlja, što dovodi do novog gubitka odometrije.

Fuzija senzora

Vizualna odometrija je dobar izvor odometrije (bolja od odometrije kotača [9]), no ima svoje mane kao što je gubitak odometrije u područjima s malo vizualnih značajki(monotona scena, scena jednoličnog uzorka...). Kako bi se borilo protiv takvog problema jedno od rješenja bi moglo biti koristiti kombinaciju više senzora, tako da kad jedan ima poteškoća da ga drugi potpomaže. Kao što je navedeno do sada u potpotpoglavlju 6.4.2, RTABMAP ima mogućnost primanja IMU podataka i njegovo labavo povezivanje s vizualnom odometrijom. Drugi pristup je pokušati povezati podatke dobivene od kamere i IMU-a preko paketa *robot_localization*.

Što se tiče prvog pristupa, povezivanja IMU-a sa RTABMAP-om se vrši kroz RTABMAP-ovo sučelje postavljanjem parametra *wait_imu_to_init = true*, pod pretpostavkom da topic na kojeg dolazi poruka od IMU-a je naziva *imu/data*. Ovaj pristup se ne razlikuje puno od pristupa bez IMU-a. Neke sitne promijene u kretnji se mogu vidjeti, ali robusnost na jednoličnim površinama i dalje predstavlja problem. Kako IMU podatci ne pogoršavaju kartiranje RTABMAP-om, koristio sam ovaj pristup umjesto same vizualne odometrije bez IMU-a.

Što se tiče drugog pristupa, povezivanja IMU-a preko paketa *robot_localization*, naišao sam na razne probleme. Za početak samo povezivanje, kao što je navedeno u potpoglavlju 6.6 potrebno je prilagoditi teme(topic) i podesiti određene parametre:

- u čvoru *rtabmap/rgbd_odometry* remapirat topic "odom" u "/vo"
- u čvoru *rtabmap/rtabmap* remapirat topic "odom" u "/odometry/filtered"
- u čvoru *rtabmap/rgbd_odometry* promijeniti parametar *publish_tf* = **false**
  - Ne želimo tf od rgbd odometrije kad odometriju za rtabmap dobivamo od filtera za fuziju senzora (EKF/UKF čvor *robot_localization* paketa)
- u čvoru *rtabmap/rtabmap* promijeniti parametar *approx_sync* = **true**
  - Problem sa sinkronizacijom, kamera i njeni topic-i ne objavljuju se kada i odometrija filtera, moramo dopustiti ne egzaktnu sinkronizaciju, u protivnom ne dobivam poruke u RTABMAP-u. Ovo stvara još jedan problem, a taj je što neegzaktna sinkronizacija dovodi do nepoklapanja odometrije s podatcima iz kamere. Nepoklapanja dovode do distorzija karte do neprepoznatljivosti. Uz to IMU dosta brzo počinje raditi svoje greške, što također utječe na kombiniranu odometriju. Ja nisam uspio podesiti parametre *robot_lokalization* čvora za estimaciju i RTABMAP-a kako bi popravio ovu situaciju. Na internetu sam naletio na mnoge koji imaju isti problem, neki koji kažu da su riješili problem [20] [21] i one koji nisu [22] [23]. Isprobao sam njihove konfiguracije i nisam došao do željenog rješenja. Čak i kada bih riješio problem sa sinkronizacijom i pomakom, tu bi i dalje ležao problem gubitka odometrije. Naime čvor za estimaciju pozicije prima vizualnu odometriju, koja se može izgubiti, što znači da RTABMAP "ne zna" kako spojiti trenutnu sliku s ostatkom karte, ako nam čvor za estimaciju daje neku odometriju koja nije vizualno definirana onda može doći do velikih nepodudaranja s kartom. Kako se gubitak odometrije bez pristupa sa *robot_lokalization*-om da

[20] https://answers.ros.org/question/320379/improving-odometry-from-rtab-mapping/

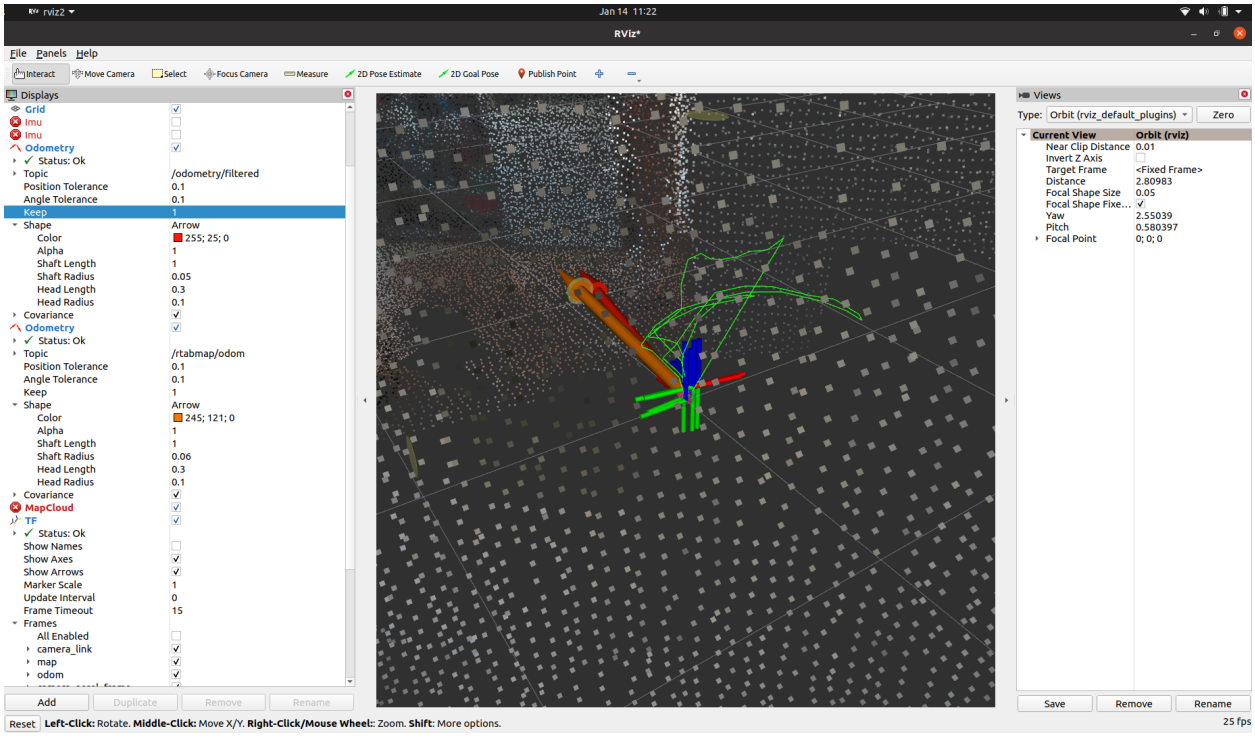
[21] https://answers.ros.org/question/377784/robot_localization-visual-odom-imu-gps/

[22] https://answers.ros.org/question/396271/rtabmap_ros-generates-a-distorted-map-when-using-the-filtered-odometry-by-robot_localization-ekf-node/

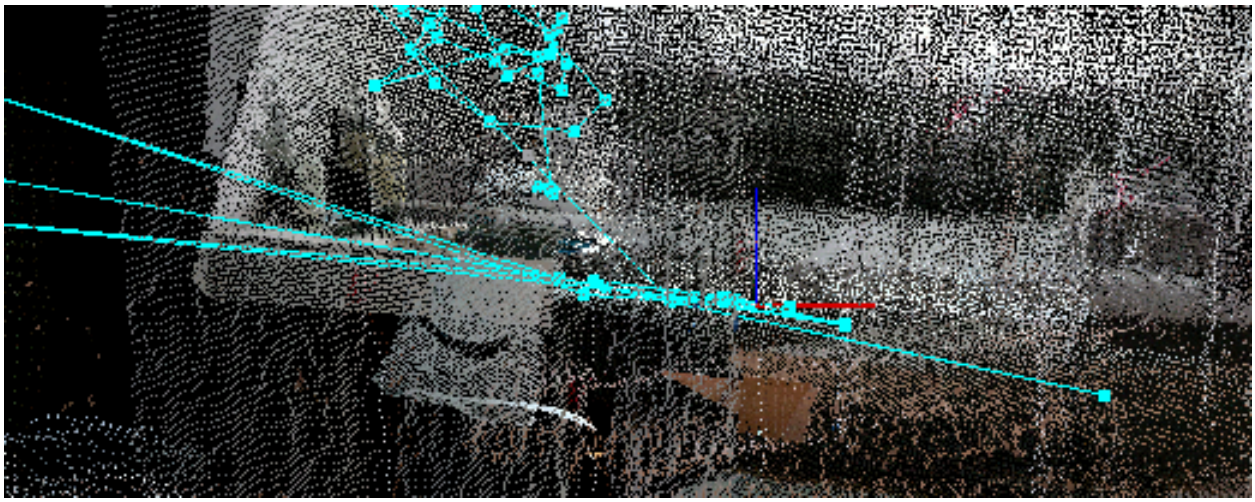
[23] https://answers.ros.org/question/221586/how-to-fuse-imu-visual-odometry-using-robot_localization/?answer=396935#post-id-396935

lako povratiti resetiranjem odometrije, i kartiranje je precizno zbog samo vizualnih ograničenja, taj je pristup, u mom slučaju, gdje želimo dobiti preciznu kartu jedini bio primjenjiv.

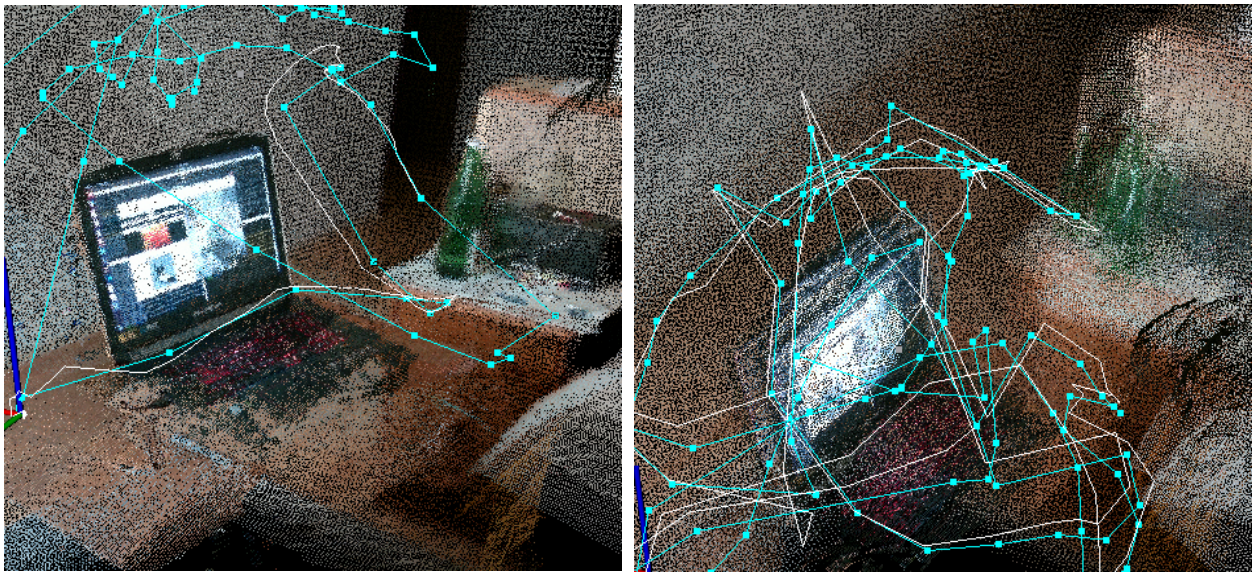
U nastavku slijedi par slikovitih primjera problema na koje sam naletio s korištenjem *robot_localization* paketa:



Slika 19: Vizualizacija vizualne odometrije dobivene rgbd_odometrijom (narančasta strelica) i odometrije dobivene iz čvora za estimaciju (crvena strelica): mali pomak u odometriji



Slika 20: Skok u odometriji zbog naglijeg pokreta kamere (IMU + VO u ovom primjeru rade veliku grešku)



Slika 21: Usporedba kartiranja: samostalno korištenje VO od RTABMAP-a(lijevo), VO sa paketom *robot_localization*(desno). Vidi se kako mali pomak u odometriji radi potpuno zamučenje 3D karte

Testiranje samostalnog rješenja

Iz prethodnih slika se može vidjeti da u mom primjeru *robot_localization* sa RTABMAP-om nije dao rješenje za precizno kartiranje. U nastavku slijede primjeri samostalnog korištenja RTABMAP-a i Intel RealSense D455 kamere. Kamera je spojena na laptop s procesorom Intel Core i7, osma generacija, i držana je u ruci prilikom kartiranja. Rezultati su dobiveni na 3 lokacije:

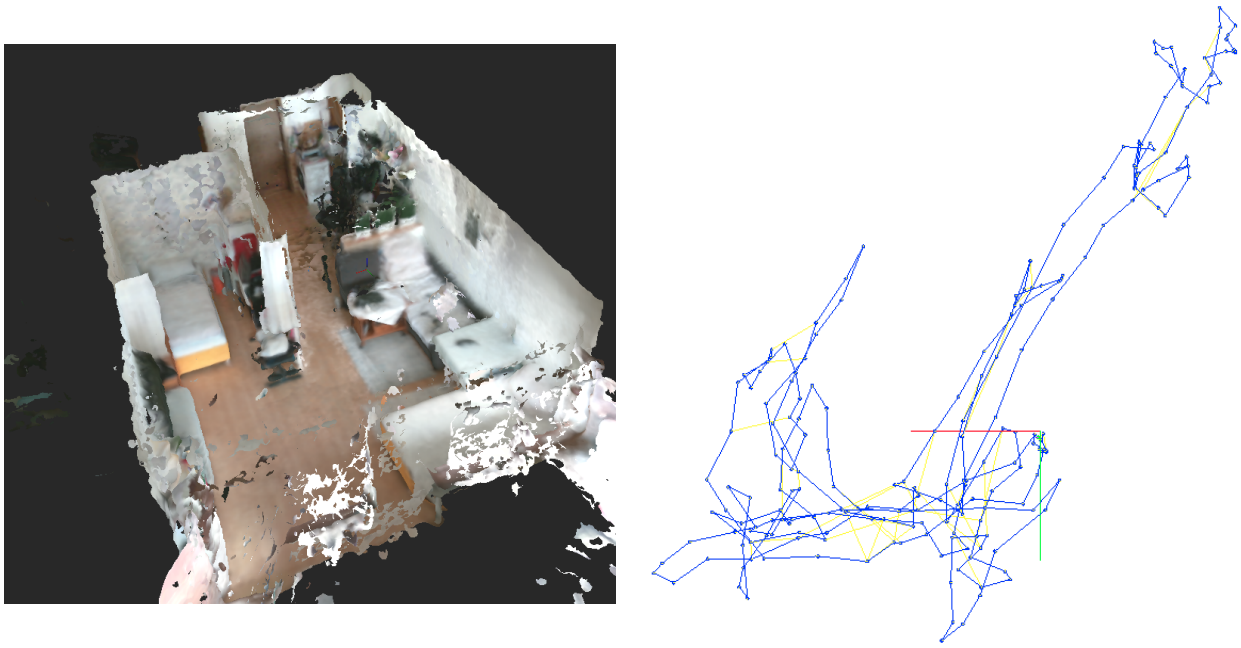
- Lokacija A: zatvoreni prostor, u stanu (cca. 35 metara kvadratnih)
- Lokacija B: vani, oko parkinga za bicikle (opsega približno 30mx5m)(kraća)
- Lokacija C: vani, oko kompleksa kuća (opsega približno 50mx20)(duža)
- Lokacija D: vani, na livadi

Lokacija A je lokacija u stanu u zatvorenom prostoru. S kamerom se kartiralo tako da se dobije predodžba o detaljima u prostoru. Dobiven je gusti 3D model prostorije. Prikaz se može vidjeti na slici Slika 22. Na slici se vidi prikaz 3D modela nakon procesiranja u RTABMAP-u. RTABMAP ima funkcionalnost za dodatnim optimiziranjem grafa nakon završetka kartiranja, čime se mogu postići bolji rezultati od onih dobivenih u stvarnom vremenu. Broj čvorova[24] u grafu je 231.

Lokacija B je lokacija na otvorenome. Kartiranje je izvođeno kruženjem oko parkinga za bicikle. Parking za bicikle se sastoji od 2 odvojene cjeline jedan za drugim. Napravljen je jedan krug oko cijelog parkinga te još jedan polukrug kroz prolaz između dvije cjeline. Dobiven 3D model se može vidjeti na slici Slika 24. Broj čvorova u grafu je 167.

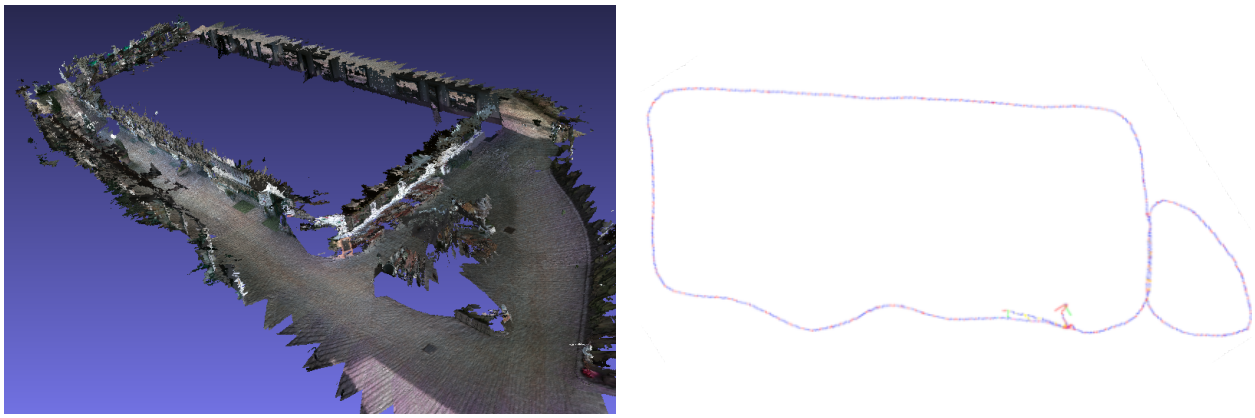
[24]u VSLAM-u se za svaku novu pozu kamere kreira novi čvor, ovaj broj može dati predodžbu o veličini kartiranja

Lokacija C je također lokacija na otvorenome. Kartiranje je izvođeno oko kompleksa objekata. Napravljena su 2 kruga, jedan veći, jedan manji. Velika je petlja zatvorena povratkom na početak, te je karta ispravljena. Za razliku od prethodna 2 ovo je mapiranje duže trajalo te se izvodilo na malo većem prostoru. RTABMAP je počeo kasniti s procesiranjem slika, pred kraj kartiranja, no sa zakašnjenjem završio kartiranjem. Broj čvorova u grafu kod ovog kartiranja je iznosilo 710. Rezultati se mogu vidjeti na slici Slika 23.

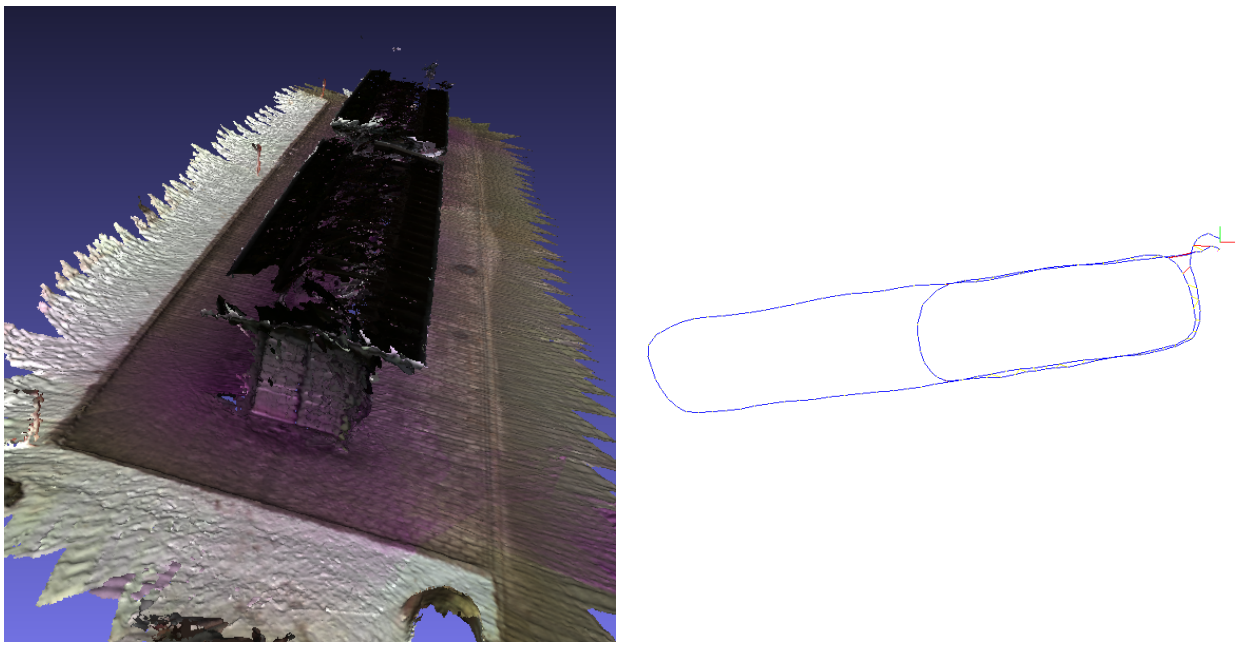


Slika 22: Lokacija A: 3D gusta karta(lijevo), putanja(desno)

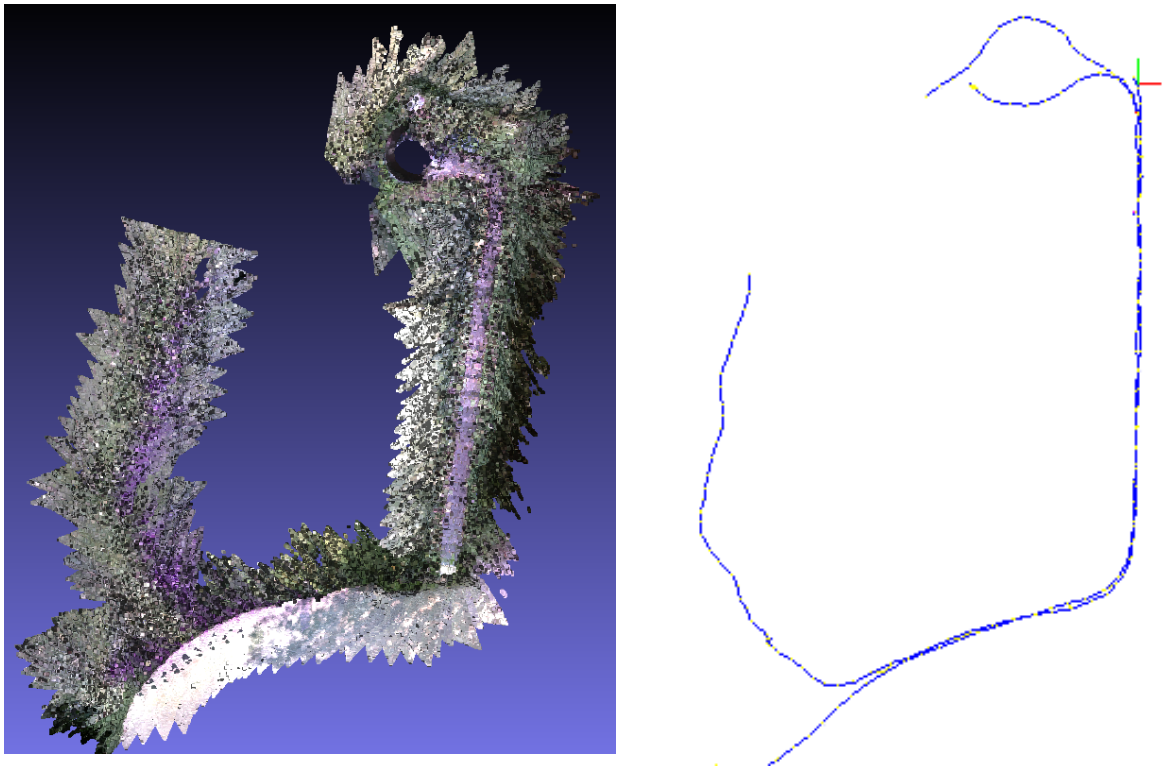
Lokacija D je lokacija koja se izvodila na livadi. Livada kao i poljoprivredni tereni imaju karakteristiku da su tereni jednolikog uzorka. Tereni s jednolikim uzorkom predstavljaju problem vizualnoj odometriji, ne može se u slici prepoznati u kojem se smjeru kamera pomaknula. Rezultat kartiranja može se vidjeti na slici Slika 25. Može se primijetiti kako karta nije spojena, vizualna odometrija se "izgubila" nekoliko puta. Kada se vizualna odometrija izgubi i postavljen je parametar za automatsko resetiranje odometrije(vidi potpoglavlje RTABMAP-ova vizualna odometrija), RTABMAP stavi mapu u pozadinu i započne novu kartu s novom odometrijom. Ako se vratimo na mjesto gdje se može prepoznati petlja onda se trenutna(nova karta) spaja s kartom u pozadini s kojom je prepoznata petlja.



Slika 23: Lokacija C: 3D gusta karta(lijevo), putanja(desno)



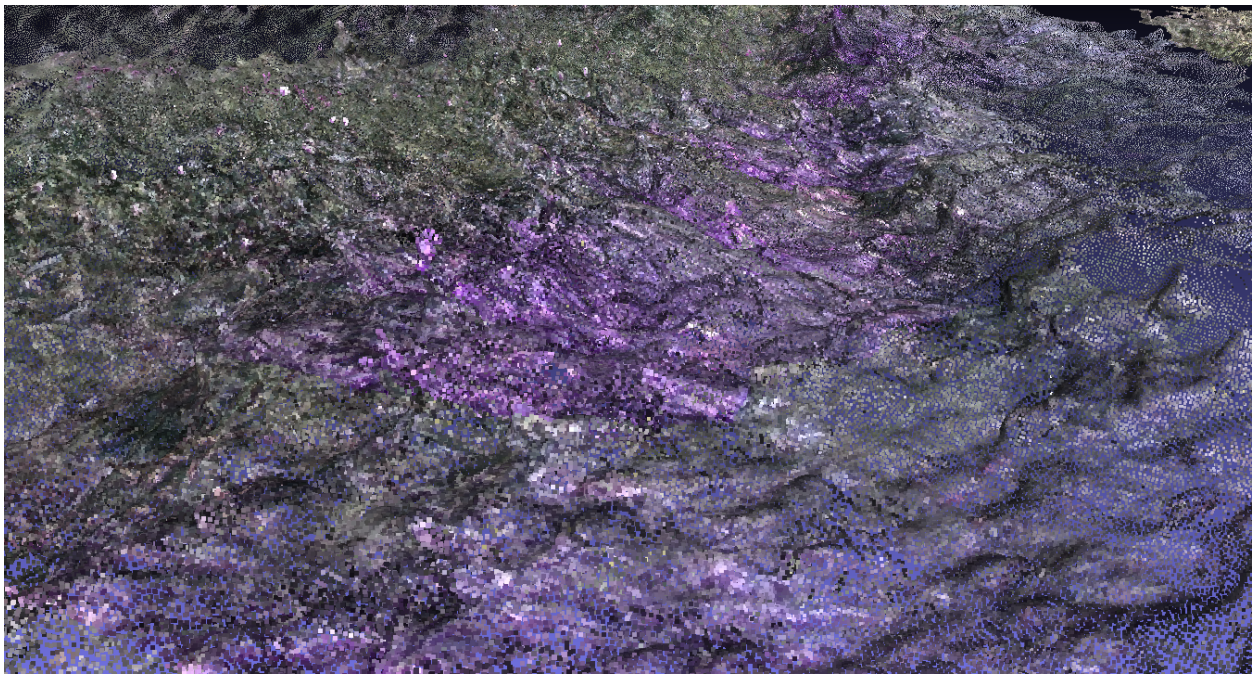
Slika 24: Lokacija B: 3D gusta karta(lijevo), putanja(desno)



Slika 25: Lokacija D: 3D gusta karta(lijevo), putanja(desno)

Na rezultatima lokacije D, može se primijetiti kako kartiranje oko livade nije bilo problematično, no kada se prešlo na kartiranje livade tu se odometrija počinje gubiti. Može se primijetiti kako prelaskom iz sjene na suncem osvijetljene površine, nije imalo veliki utjecaj na kartiranje, no povratkom na istu lokaciju nakon nekog vremena(kada se sunce pomakne), sjena se pomakne, što može dovesti do krivih zatvaranja petlje i iskrivljenja karte. Također neravan teren, može dovesti do podrhtavanja kamere, što također dovodi do gubitka odometrije na poljoprivrednim terenima.

Kao rezultat korištenja RTABMAP-a dobila se gusta 3D karta (eng. dense pointcloud). No koliko je ona "precizna"? Ispod na slikama Slika 26 i 27 može se vidjeti uvećani prikaz livadnog dijela 3D karte, te RGB slika livade.



Slika 26: Uvećani prikaz dijela 3D karte: livadni dio

Slika 27: RGB slika dobivena Intel RealSense D455 kamerom tokom snimanja na livadi

Može se zaključiti kako se na ovom prikazu ne mogu prepoznati biljke koje se nalaze na livadi, niti njihova visina. 3D pointcloud jednostavno nije dobra reprezentacija za detaljne scene s malim i tankim objektima, za koje je teško izračunati udaljenost od kamere precizno. Također ako se pogleda donji desni kut u slici Slika 27 može se primijetiti kako zbog sunčevog

svijetla Intel RealSesne D455 kamera pati od ljubičastih resica (eng. Purple fringing [25]). Iako na livadi biljke nisu ljubičaste boje, na 3D modelu na slici Slika 26 se može vidjeti kako je model poprimio ljubičastu boju.

U ovom poglavlju pokazani su rezultati dobiveni korištenjem Intel RealSense D455 kamere sa RTABMAP-om u ROS2 okruženju. Može se vidjeti koje su mogućnosti ovog VSLAM rješenja. Moguće je dobiti model zatvorene prostorije, model vanjskih objekata i trajektorije u vanjskom okruženju koje ima dovoljno vizualnih značajki. Kvaliteta istih ovisi o veličini modela, broju značajki, rezoluciji kamere, itd.

Ovakav pristup nije primjenjiv u poljoprivrednim aplikacijama, ako se od nas traži kartiranje. Navedeno je već nekoliko razloga a to jest: različito osvjetljenje terena, neravan teren(pristup iz zraka bi bio primjenjiviji), manjak različitih vizualnih značajki... Uz to poljoprivredni strojevi odrađuju posao u dužem vremenskom periodu(više od 5 minuta koje sam ja maksimalno koristio u kartiranju). Sve i da se odometrija ne izgubi na sceni jednolikog uzorka, i dalje bi imali problema s procesiranjem velike karte u stvarnom vremenu. Da se primijetiti da prikupljanje informacija o karti se može obaviti prije te se na osnovu tih informacija napravi plan za obavljanje zadatka. Proces kartiranja bi trebao biti izvanmrežni problem, tako da možemo koristiti više računskih resursa, boljih algoritama koji imaju fokus na točnosti 3D rekonstrukcije(a ne na izvođenje u stvarnom vremenu), slike većih rezolucija...

[25]U fotografiji (posebno digitalnoj fotografiji), ljubičaste resice izraz je za nefokusiranu ljubičastu ili magenta "duh" sliku na fotografiji. Ova optička aberacija općenito je najvidljivija kao bojanje i posvjetljivanje tamnih rubova u blizini svijetlih područja širokog spektra osvjetljenja, kao što je dnevno svjetlo ili razne vrste svjetiljki s plinskim pražnjenjem. (IZVOR: wikipedija)

8 ZAKLJUČAK

Istražena su postojeća rješenja za trodimenzionalno kartiranje vanjskih prostora u stvarnom vremenu u ROS(Robot Operating System) okruženju upotrebom Intel RealSense D455 kamere. Predloženo je i testirano RTABMAP VSLAM rješenje koje je prilagođeno za izradu gustih trodimenzionalnih karata. Predložene su dvije konfiguracije sustava, te opis povezivanja svih komponenti. Konfiguracija sustava s Kalmanovim filtrom se pokazala neprimjenjiva. Pokušavaju se povezati vizualna odometrija i integrirani IMU u Intel RealSense D455 kameri. Kako vizualna odometrija kao i IMU pate od akumulacije greške kroz vrijeme, fuzijom ova dva izvora odometrije dolazi do bržeg gubitka odometrije nego kada se koristi vizualna odometrija samostalno. Stoga je predloženo rješenje samostalno korištenje RTABMAP-a. Rezultati su dobivene trodimenzionalne karte snimljenih u unutarnjim(zatvorenim) i vanjskim prostorima. Rezultati su pokazali da je testiranje u zatvorenom prostoru bilo uspješno, dobiven je jasan i gusti trodimenzionalni model prostorije. Testiranje koje se izvodilo u gradu (na parkiralištu i oko zgrade) također je bilo uspješno, dobiveni su trodimenzionalni modeli prostora. Testiranje koje je bilo izvođeno na livadi nije dalo zadovoljavajuće rješenje, nije dobivena konzistentna karta prostora, već samo djelomična karta. Vizualna odometrija zahtjeva različite vizualne značajke u slici kako bi uspjela pratiti položaj kamere iz slike u sliku. Kada dođe do gubljenja vizualne odometrije, prekida se kartiranje (kamera se izgubila). Do gubljenja vizualne odometrije može doći zbog više razloga bilo to nagli pokret kamere ili je to manjak različitih vizualnih značajki u slici. Trzanje je prebrz pokret kamere da se izračuna pravovremeno njen pomak, te dolazi do gubitka vizualne odometrije. Ako u uzastopnim slikama nema dovoljno različitih vizualnih značajki za estimaciju pokreta kamere, vizualna odometrija se gubi. Livadno područje pati zbog manje količine različitih vizualnih značajki za estimaciju pomaka kamere, kako je trava jednoličnog uzorka sve su značajke "slične" te je teško procijeniti koja značajka odgovara kojoj iz slike u sliku. Također neravan teren na livadi može dovesti do podrhtavanja kamere, koja također mogu dovest do gubitka vizualne odometrije. Kvaliteta karte na mjestima s malo značajki je loša, teško se može prepoznati što je snimano. Rad je pokazao da je Intel RealSense D455 kamera primjenjiva i u zatvorenim prostorima, kao i u vanjskim(otvorenim) prostorima, manji utjecaj sunca može tolerirati jer koristi stereo podudaranja za estimaciju dubine. RTABMAP rješenje je primjenjivo na mjestima gdje je dovoljno vizualnih značajki za estimaciju pokreta kamere. Također ako dođe do gubitka vizualne odometrije RTABMAP ima alate za prepoznavanje posječenih mjesta, te mogućnost ponovnog pronalaženja u karti.

Literatura

[1] M. Perer-Ruiz and S. Upadhyaya. Gnss in precision agricultural operations. in new approach of indoor and outdoor localization systems. IntechOpen, 2012. London, UK.

[2] Doaa Mahmoud, Mohammed Abdel-Megeed Mohammed Salem, Hassan Ramadan, and Mohamed Roushdy. 3d graph-based vision-slam registration and optimization. *International Journal of Circuits, Systems and Signal Processing*, 8:123–, 06 2014.

[3] Summary on the kalman filter friends: Kf, ekf, ukf, eif, seif. s Interneta, `http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam08-kf-wrapup.pdf`. Cyrill Stachniss, University of Freiburg.

[4] Norhidayah Yatim and Norlida Buniyamin. Particle filter in simultaneous localization and mapping (slam) using differential drive mobile robot. *Jurnal Teknologi*, 77, 12 2015.

[5] Hauke Malte Strasdat, J. M. M. Montiel, and Andrew J. Davison. Visual slam: Why filter? *Image Vis. Comput.*, 30:65–77, 2012.

[6] Slam course - 21 - short summary (2013/14; cyrill stachniss). s Interneta, `https://www.youtube.com/watch?v=op0byxn7IzI&list=PLgnQpQtFTOGQrZ4O5QzbIHgl3b1JHimN_&index=22`. 10.2.2014.

[7] Direct visual slam. s Interneta, `https://www.kudan.io/direct-visual-slam/`. Kudan, 16.09.2020.

[8] Mohammad H.; Saripan M. Iqbal; Ismail Napsiah Bt. Aqel, Mohammad O. A.; Marhaban. Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, 2016.

[9] Alireza; Hoseinnezhad Reza Yousif, Khalid; Bab-Hadiashar. An overview to visual odometry and visual slam: Applications to mobile robotics. *Intelligent Industrial Systems*, pages 289–311, 2015.

[10] X. Gao and T. Zhang. *Introduction to Visual SLAM: From Theory to Practice*. Springer Singapore, 2021.

[11] Orb slam 2 : an open-source slam system for monocular, stereo and rgb-d cameras. s Interneta, `https://pdfs.semanticscholar.org/7d77/312b3ee80f11ce4b890d4c017b04e95aa53c.pdf`. Raul ur-Artal and Juan D. Tardos.

[12] Davide Scaramuzza. Vision algorithms for mobile robotics. s Interneta, `https://rpg.ifi.uzh.ch/docs/teaching/2020/10_multiple_view_geometry_4.pdf`.

[13] What is ros? s Interneta, `http://wiki.ros.org/ROS/Introduction`. last edited 8.8.2018.

[14] Ros1 vs ros2, practical overview for ros developers. s Interneta, `https://roboticsbackend.com/ros1-vs-ros2-practical-overview/`. The Robotics Back-End.

[15] Beginner's guide to depth (updated). s Interneta, `https://www.intelrealsense.com/beginners-guide-to-depth/`. 15.7.2019.

[16] Orb-slam3. s Interneta, https://github.com/UZ-SLAMLab/ORB_SLAM3.

[17] Lsd-slam: Large-scale direct monocular slam. s Interneta, https://vision.in.tum.de/research/vslam/lsdslam.

[18] Svo pro: Semi-direct visual-inertial odometry and slam for monocular, stereo, and wide angle cameras. s Interneta, https://rpg.ifi.uzh.ch/svo_pro.html.

[19] Tue-Cuong Dong-Si and Anastasios I. Mourikis. Consistency analysis for sliding-window visual odometry. In *2012 IEEE International Conference on Robotics and Automation*, pages 5202–5209, 2012.

[20] Dense visual slam. s Interneta, https://vision.in.tum.de/data/software/dvo.

[21] Mathieu Labbé and François Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation: LabbÉ and michaud. *Journal of Field Robotics*, 36, 10 2018.

Popis slika

1 Shema optimizacijskog pristupa: *frontend* i *backend* 5
2 Detekcija ključnih točaka: jednolični prostor, rub, kut 6
3 Metoda DoG: Shematski prikaz oduzimanja slika različite zamučenosti(lijevo). Rezultat oduzimanja(desno) 7
4 FAST detektor kuteva 7
5 SIFT: gradijenti 8
6 Skice algoritama za estimaciju položaja kamere: epipolarna geometrija(lijevo), PnP(sredina), ICP(desno) 11
7 RANSAC: Cijeli skup podudaranaja(lijevo), skup u kojem su izbačeni outlieri(desno) 12
8 Razlika između ROS1 i ROS2 arhitekture 16
9 Čvor sa životnim ciklusom u ROS2 17
10 ROS Master, server za parametre - usporedba ROS1 i ROS2 19
11 Ilustracije različitih metoda računanja dubine: Strukturirano svijetlo(lijevo), stereo dubina(u sredini), vrijeme leta - LiDAR (desno) 22
12 Prikaz različitih Intel RealSense kamera: SR305 (lijevo), D455 (u sredini), L515 (desno) 22
13 Intel RealSense D455 23
14 ROS rviz sucelje sa sa prikazanom RGB (dole lijevo), dubinskom (gore desno) i infra crvenim slikama (dole desno) kamere, te položajem kamere (sredina) . 33
15 Usporedba IMU poruke bez filtra(lijevo) i sa filtrom(desno) 35
16 Prozor rtabmapviz-a za podešavanje parametara 39
17 Prikaz ROS čvorova samo s RTABMAP-om, bez robot_localization-a 41
18 Prikaz ROS čvorova sa robot_localization-om 41
19 Vizualizacija vizualne odometrije dobivene rgbd_odometrijom (narančasta strelica) i odometrije dobivene iz čvora za estimaciju (crvena strelica): mali pomak u odometriji 45
20 Skok u odometriji zbog naglijeg pokreta kamere (IMU + VO u ovom primjeru rade veliku grešku) 45
21 Usporedba kartiranja: samostalno korištenje VO od RTABMAP-a(lijevo), VO sa paketom *robot_localization*(desno). Vidi se kako mali pomak u odometriji radi potpuno zamučenje 3D karte 46
22 Lokacija A: 3D gusta karta(lijevo), putanja(desno) 47
23 Lokacija C: 3D gusta karta(lijevo), putanja(desno) 47
24 Lokacija B: 3D gusta karta(lijevo), putanja(desno) 48
25 Lokacija D: 3D gusta karta(lijevo), putanja(desno) 48
26 Uvećani prikaz dijela 3D karte: livadni dio 49
27 RGB slika dobivena Intel RealSense D455 kamerom tokom snimanja na livadi 49

Popis tablica

1 Usporedba specifikacija Intel RealSense D435/D435i i D455 23
2 Popis poznatijih SLAM rješenja otvorenog koda 24
3 Značajke VSLAM rješenja 30

Listings

1 Komanda za pokretanje kamere 32
2 Komande za pokretanje IMU filtera 34
3 Komanda za pokretanje RTABMAP-a 37

9 SAŽETAK I KLJUČNE RIJEČI

Hrvatski

U ovom radu su istražena postojeća rješenja za trodimenzionalno kartiranje vanjskih prostora u stvarnom vremenu u ROS okruženju upotrebom Intel RealSense D455 kamere. Rad obuhvaća uvid u korake općenitog VSLAM(Vizualna Simultana Lokalizacija i Kartiranje) rješenja, te daje uvid u postojeća VSLAM rješenja dostupna u ROS okruženju. Predloženo je i istestirano rješenje RTABMAP, te je napisana datoteka za pokretanje svih čvorova u ROS-u. Testiranje koje se izvodilo u zatvorenom i otvorenom(vanjskom) području je potkrijepljeno dobivenim 3D modelima.

Ključne riječi: *ROS, RealSense, VSLAM, RTABMAP, trodimenzionalno kartiranje*

English

This paper investigates existing solutions for three-dimensional real-time outdoor mapping in a ROS environment using an Intel RealSense D455 camera. The paper includes an insight into the steps of a general VSLAM (Visual Simultaneous Localization and Mapping) solution, and provides an insight into existing VSLAM solutions available in the ROS environment. An RTABMAP solution was proposed and tested, and a launch file was written to run all nodes in ROS. Testing performed in closed and open (outdoor) area was supported by the obtained 3D models.

Keywords: *ROS, RealSense, VSLAM, RTABMAP, 3D mapping*

10 Dodatak A

Uz ovaj diplomski rad prilažem datoteku za pokretanje ROS2 čvorova (eng. launch file) pod nazivom *rtabmap_all.launch.py*. Ova datoteka za pokretanje je napisana kako bi mi olakšala testiranje različitih konfiguracija rješenja s RTABMAP-om. Pokretanje kamere se obavlja posebno s komandom Listing 1 u ovom radu. U nastavku kratak opis korištenja priložene datoteke za pokretanje.

```
$ ros2 launch rtabmap_ros rtabmap_all.launch.py \
        rtabmap_args:="--delete_db_on_start" \
        wait_imu_to_init:=true \
        rtabmapviz:=true                                    (0.1)
```

Komanda iznad pokreće 4 čvora (RGBD odometriju, RTABMAP, RTABMAPViz i IMU filter čvor(Madgwick)) sa zadanim parametrima:

- Odom/Strategy = "0" (0 - F2M, 1 - F2F)
  RTAB-Map implementira dva standardna pristupa odometriji [Scaramuzza i Fraundorfer, 2011.] pod nazivom Frame-To-Map (F2M) i Frame-To-Frame (F2F). Glavna razlika između ovih pristupa je u tome što F2F registrira novi okvir na zadnjem ključnom kadru, a F2M registrira novi okvir na lokalnoj karti značajki stvorenih iz prošlih ključnih kadrova.

- Vis/CorType = "0" (0 - feature matching, 1 - optical flow)
  - U dokumentaciji se spominje da ako se koristi strategija F2M onda je Cor_type postavljen na feature matching, dok ako je strategija postavljena na F2F onda je Cor_type optical flow. Više o parametrima RTABMAP-a može se pročitati tu [21]

- Vis/FeatureType = "8" (vidi poglavlje 6.5)

- Odom/ResetCountdown = "1" (vidi potpoglavlje "RTABMAP-ova vizualna odometrija" u cjelini 7)

- imu_filter = "madg" (madg - Madgwick filter, comp - Complementary filter)
  - Ako parametar wait_imu_to_init nije postavljen na true onda RTABMAP ne uzima informaciju iz IMU-a.

S gore prikazanom konfiguracijom je obavljeno testiranje i dobiveni prikazani rezultati.

Za pokretanje robot_lokalization paketa potrebno je samo promijeniti parametar robot_localization. Komanda sada izgleda ovako:

```
$ ros2 launch rtabmap_ros rtabmap_all.launch.py \
        rtabmap_args:="--delete_db_on_start" \
        wait_imu_to_init:=true \
        rtabmapviz:=true \
        robot_localization:=true                            (0.2)
```

Isprobao sam razne konfiguracije parametara s robot localization paketom ali bezuspješno. Korištenje samostalne odometrije RTABMAP-a je bio bolji pristup.

Za pokretanje kartiranja u Rviz-u potrebno je na komandu 0.1 dodati sljedeću liniju:

```
rviz:=true
```

Još jedna dodatna stvar koja je omogućena s RTABMAP-om je detekcija prepreka (eng. obstacles detection). Mali primjer korištenja omogućen je s priloženom datotekom za pokretanje. Komanda izgleda ovako:

```
$ ros2 launch rtabmap_ros rtabmap_all.launch.py \
        rtabmap_args:="--delete_db_on_start" \
        localization:=true \
        rviz:=true \
        obs:=true                                    (0.3)
```

Opis navedene komande 0.3:

- rtabmap_args:="–delete_db_on_start"
  Briše bazu podataka na lokaciji "database_path" parametra. Ako želimo započeti od početka.

- localization:=true
  - RTABMAP može raditi na 2 načina(eng. mode): kartiranje s lokalizacijom, ili samo lokalizacija(bez kartiranja). Postavljanjem parametra localization na true ne izvodi se kartiranje, odnosno odbacuju se stari kadrovi. Ovo se može koristiti na prethodno učitanoj karti. Ako karta nije prethodno učitana (parametar rtabmap_args:="–delete_db_on_start" je postavljen) onda imamo nešto što podsjeća na VO.

- rviz:=true
  - Sva vizualizacija prepreka omogućena je u rviz-u.

- obs:=true
  - Pokreće RTABMAP-ov čvor za detekciju prepreka.

NAPOMENA: Svi parametri RTABMAP-a se mogu podešavati i preko parametra "rtabmap_args" sa zadanim datotekama za pokretanje koje se dobiju instalacijom RTABMAP-a u ROS2 okruženju. Meni je ovako napisana datoteka za pokretanje poslužila jer mi je omogućila pokretanje više čvorova istovremeno (RGBD odometrija, RTABMAP, RTABMAPViz, rviz, robot_localization, IMU filter, obstacles detection čvor...). Nisu navedeni svi parametri koji se mogu modificirati, ali na ove sam ja obratio pažnju.