Lokalizacija i robotsko rukovanje otpadom iz mora

Jakovljević, Jan

Master's thesis / Diplomski rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:624919

Rights / Prava: Attribution 4.0 International/Imenovanje 4.0 međunarodna

Download date / Datum preuzimanja: **2024-05-18**

Repository / Repozitorij:

Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:
Doc. dr. sc. Marko Švaco, mag. ing. mech.

Student:
Jan Jakovljević

Komentor:
dr. sc. Ante Bakić, mag. ing. mech.

Zagreb, 2022. godina

Izjavljujem da sam ovaj radi izradio samostalno koristeći znanja stečena tijekom studija i navedene literature.

Zahvaljujem se mentoru doc. dr. sc. Marku Švaci i komentoru dr. sc. Anti Bakiću na dodijeljenoj temi, pruženoj pomoći i usmjeravanju prilikom izrade rada.

Zahvaljujem se prijatelju i kolegi Danielu Biharu mag. phys. geophys na savjetima i ugodnoj radnoj atmosferi prilikom izrade ovog rada.

Veliko hvala obitelji i zaručnici Karli na pruženoj potpori tijekom cijelog studija.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske radove studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa: 602-14/22-6/1	
Ur. broj: 15-1703-22-	

DIPLOMSKI ZADATAK

Student: **JAN JAKOVLJEVIĆ** Mat. br.: 0035206493

Naslov rada na hrvatskom jeziku: **Lokalizacija i robotsko rukovanje otpadom iz mora**

Naslov rada na engleskom jeziku: **Localization and robotic manipulation with marine litter**

Opis zadatka:

S obzirom na porast zagađenja svjetskih oceana, javlja se potreba za autonomnim čišćenjem velikih nenaseljenih dijelova obale na koje more izbacuje sve veće količine otpada. Kao jedno od ekološki prihvatljivih rješenja nameće se autonomna mobilna platforma s robotskom rukom za sakupljanje otpada. Danas je razvoj upravljačke podrške za autonomne robote moguće ostvariti na različitim softverskim platformama od kojih je robotski operativni sustav (eng. Robot Operating System – ROS) standard koji je opće prihvaćen u akademskoj zajednici, a sve se više se koristi i u industriji. ROS pruža okolinu za razvoj modularne upravljačke programske podrške, komunikacijsku infrastrukturu i otvorenu biblioteku algoritama.
U ovom radu potrebno je razviti programsku podršku za lokalizaciju otpada te rukovanje otpadom pomoću robotske ruke.
U sklopu ovog rada, na postojećem prototipnom robotskom sustavu za sakupljanje otpada potrebno je:
- Izvršiti kalibraciju kamere i robota,
- Ispitati i validirati točnost spajanja oblaka točaka snimanjem scene iz više položaja robota (eng. „point cloud registration and stitching"),
- U upravljački sustav robota integrirati prethodno naučen algoritam YOLO v5 za detekciju određenih klasa otpada na 2D slici,
- Predložiti i ispitati metodu segmentacije objekta (otpada) od podloge,
- Razviti algoritam 3D lokalizacije otpada u koordinatnom sustavu robotske ruke,
- Istražiti planere za manipulaciju dijelovima (eng. „grasp planning") unutar ROS okruženja te implementirati jedan takav planer,
- Implementirati postojeće algoritme planiranja trajektorija robotske ruke dostupne u ROS-u,
- Predložiti strategije sakupljanja otpada, implementirati i validirati najmanje jednu strategiju.

Rad cjelokupnog sustava potrebno je eksperimentalno validirati u ROS okruženju, a prema mogućnostima i na postojećoj opremi.
U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:	Rok predaje rada:	Predviđeni datum obrane:
5. svibnja 2022.	7. srpnja 2022.	18. srpnja do 22. srpnja 2022.

Zadatak zadao: doc. dr. sc. Marko Švaco

Komentor: dr. sc. Ante Bakić

Predsjednica Povjerenstva: prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA II
POPIS TABLICA V
POPIS OZNAKA I KRATICA VI
SAŽETAK VII
ABSTRACT VIII
1. UVOD 1
1.1. Autonomni robotski sustavi 2
1.2. Izazovi i pristup zadatku 3
2. EKSPERIMENTALNI POSTAV SUSTAVA ZA MANIPULACIJU OTPADOM 6
2.1. Kolaborativna robotska ruka s hvataljkom 7
2.2. Jetson Xavier NX 9
2.3. Stereovizijski sustav 10
2.4. Robotski operacijski sustav 11
2.5. Neuronska mreža YOLO v5 13
2.6. Komunikacija između dijelova eksperimentalnog postava sustava 14
2.7. Open3D programska knjižnica 15
3. IMPLEMENTACIJA ROS-A U EKSPERIMENTALNI POSTAV SUSTAVA 16
3.1. MoveIt 16
3.2. Sučelje za upravljanje robotskom rukom i vizualizacija podataka 19
3.3. Kalibracija 3D vizijskog sustava 21
3.4. Detekcija i prostorna lokalizacija objekata 25
3.5. Segmentacija objekata u prostoru i njihovo preklapanje s primitivima 28
3.6. Algoritmi za izuzimanje i manipulaciju objektima 36
4. EVALUACIJA EKSPERIMENTALNOG POSTAVA SUSTAVA 44
5. ZAKLJUČAK 47
PRILOZI 50

POPIS SLIKA

Slika 1.1 Prikaz zagađene obale [1] 1
Slika 1.2 a)Roomba [2], b)Boston Dynamics Atlas [3] 2
Slika 1.3 a)Mobilna platforma s robotskom rukom Panda [4], b) Mobilna platforma s robotskom rukom UR5 [5] 3
Slika 1.4 Franka Emika Panda [6] 4
Slika 2.1 Eksperimentalni postav sustava za manipulaciju otpadom 6
Slika 2.2 a) xArm5, b) xArm6, c) xArm7 [7] 7
Slika 2.3 Radni opseg xArm6 robotske ruke [8] 8
Slika 2.4 GUI aplikacija za robotsku ruku [8] 8
Slika 2.5 xArm Gripper [10] 9
Slika 2.6 Jetson Xavier NX [11] 10
Slika 2.7 Intel RealSense D455 [14] 11
Slika 2.8 Razvojna okolina ROS-a [15] 12
Slika 2.9 Detekcija objekata YOLO neuronskom mrežom [18] 13
Slika 2.10 Shematski prikaz toka informacija unutar sustava 14
Slika 2.11 a) Učitani oblak točaka u Open3D [21], b) Obrađeni oblak točaka unutar Open3D-a[21] 15
Slika 3.1 Prikaz arhitekture move_group node-a [22] 17
Slika 3.2 Implementacija MoveIt-a s robotoskom rukom 17
Slika 3.3 a) Učitavanje URDF datoteke u MoveIt setup Assistent, b) Izrada Matrice kolizija različitih dijelova unutar robota, c) Povezivanje prvog zgloba robota sa svijetom oko sebe odnosno fiksiranje u prosotru, d) Definiranje različitih grupa robota i softvera za rješavanje kinematike i dinamike robota zajedno sa svim potrebnim parametrima, e) Definiranje grupa za planiranje kretnji, f) Spremanje predefinirani pozicija robotske ruke, g) Definiranje hvatljke, h) Definiranje vizijskog sustava i podešavanje svih potrebnih parametara, i) Definiranje kontrolera koji će se koristiti za upravljanje električnim motorima na zglobovima robotske ruke, j) generiranje svih potrebnih datoteka za korištenje MoveIt-a 19
Slika 3.4 RViz vizualizacija MotionPlanning plugin-a 20

Slika 3.5 Prikaz dobivenih podataka iz OctoMapa u RViz-u..... 21
Slika 3.6 a) Eye in Hand kalibracija, b) Hand to Eye kalibracija [23] 22
Slika 3.7 Prikaz MoveIt kalibracijske arhitekture [23]..... 22
Slika 3.8 Prikaz homogenih transformacija Eye in Hand kalibracije [23] 23
Slika 3.9 a) Kalibracijski alat unutar RViz-a, b) Prikaz robotske ruke prilikom kalibracije ... 24
Slika 3.10 Prikaz kalibracijskog postupka Eye in Hand kalibracije [23] 24
Slika 3.11 Detekcija objekata na slici pomoću YOLO mreže [25]..... 25
Slika 3.12 Prikaz lokalizacije objekta u prostoru pomoću *gb_visual_detection_3d* ROS paketa[26]..... 27
Slika 3.13 a) Prikaz oblaka točaka iz gornje perspektive, b) Prikaz oblaka točaka iz prednje perspektive 28
Slika 3.14 a) Oblak točaka prije Pass Through filtera, b) oblak točaka poslije filtera 29
Slika 3.15 Oblak točaka poslije Voxel filtera..... 30
Slika 3.16 a)-c) Prikaz nepodudaranja jednog oblaka točaka na drugi..... 31
Slika 3.17 a) Prikaz spajanja dva oblaka točaka poslije ICP registracije, b) Prikaz spajanja tri oblaka točaka 32
Slika 3.18 a) Segmentirana ravnina pomoću RANSAC algoritma, b) Prestali objekti u oblaku točaka 32
Slika 3.19 a) Gruba izolacija objekta u prostoru pomoću YOLO mreže prikazana u RViz-u, b) Izolirani objekt u prostoru pomoću YOLO mreže od ostatka spojenog oblaka točaka 33
Slika 3.20 a) Preklapanje geometrijskog primitiva za podlogu, b) Preklapanje geometrijskog primitiva za izolirani objekt..... 34
Slika 3.21 Prikaz dodanih geometrijskih primitva u ROS okolinu kao objekti za manipulaciju 35
Slika 3.22 Dijagram toka prostorne lokalizacije objekta..... 36
Slika 3.23 a) Dolazak u predefiniranu točku prihvata objekta, b) Pomicanje objekta na točku odlaganja i puštanje objekta..... 37
Slika 3.24 a) Početna pozicija robotske ruke prije pozivanja metode *pick*, b) Pozivanje metode *pick* te gdje je definiran vektora prilaza objektu robotske ruke po osi x, c)

Odmicanje od pozicije prihvata objekta po definiranom vektoru odmaka po osi z zajedno s objektom 39
Slika 3.25 Prikaz predefiniranih putanja prilaska objektu zajedno s pozicijama hvataljki 41
Slika 3.26 a)-c) hvatanje objekta iz vertikalnog prilaska bez prepreke, d)-f) hvatanje objekta s preprekom u vertikalnom prilasku i nemogućnosti prilaska s prednje strane objekta 42
Slika 3.27 Dijagram toka za manipulaciju objektima u prostoru 43

POPIS TABLICA

Tablica 1 Karakteristike Jetson Xavier NX [12]..... 10

POPIS OZNAKA I KRATICA

Kratica	Značenje
YOLO	You Only Look Once
ROS	robotski operacijski sustav (eng. *Robot Operating System*)
GUI	Graphical User Interface
SOM	System on a module
API	Application Programming Interface
PCL	Point Cloud Library
URDF	Unified Robotics Description Format
CAN	Controller Area Network
RViz	ROS Visualisation
OMPL	The Open Motion Planning Library
ICP	Iterativ closest point
RANSAC	Random sample consensus

SAŽETAK

S obzirom na porast zagađenja svjetskih oceana i mora širom planete zemlje i velike fluktuacije otpada u oceanima i morima dolazi do izbacivanja otpada na obale. Izbačeni otpad na obalama potrebno je sakupljati, što je težak i zahtjevan posao. Kao jedna od ekološki prihvatljivih rješenja nameće se autonomna mobilna platforma s robotskom rukom za sakupljanje otpada. U ovom radu dan je naglasak na rukovanje otpadom pomoću robotske ruke. Robotska ruka, odnosno sustav za rukovanje otpadom razvijen u ovom radu kasnije se može implementirati na različite mobilne platforme. U razvoju sustava za manipulaciju otpadom koriste se robotska ruka xArm6 Južno Korejske tvrtke UFACTORY. Za detekciju i grubu prostornu lokalizaciju otpada upotrebljava se konvolucijska neuronska mreža YOLO v5 koja potrebne vizijske 2D i 3D podatke dobiva iz stereovizijskog sustava Intel RealSense D455. Cjelokupna upravljačka podrška sustava razvijena je u robotskom operativnom sustavu (eng. *Robot Operating System - ROS*) pomoću programskih jezika Python i C++. Dobivene podatke iz stereovizijskog sustava, osobito oblake točaka (eng. *point clouds*) potrebno je obraditi. Obrada 3D podataka u ovom radu napravljena je pomoću programske knjižnice Open3D. Softver i sva akvizicija podataka iz okoline zajedno s obradom i planiranjem svih kretnji robota napravljena je na računalu. Odabrano računalo na modulu (eng. *system-on-module*) je Jetson Xavier NX tvrtke NVIDIA. Na početku rada opisuje se eksperimentalni postav sustava, svaka njegova komponenta te se daje dublji uvid u problematiku rada. Poslije detaljnijeg opisa problematike i sustava opisuje se način integracije različitih algoritama u sustav kako bi se ostvarila detekcija, lokalizacija, segmentacija i klasifikacija otpada u prostoru te preklapanje klastera s geometrijskim primitivom. Geometrijski primitiv se kasnije dodaje u ROS MoveIt scenu gdje se izvršava manipulacija otpadom. Na kraju rada napravljena je analiza dobivenih rezultata i evaluacija eksperimentalnog postava sustava, nakon čega se iznosi odgovarajući zaključak.

Ključne riječi : Robotska ruka, MoveIt, ROS, 3D vizijski sustav, open3D, YOLO, detekcija objekata, sterevizijski sustav

ABSTRACT

Due to the increase in pollution across the planet and the large fluctuations of waste in the oceans and seas, waste is dumped onto shores. Garbage accumulated on beaches must be collected, which is a complicated and painstaking job. An autonomous mobile platform with a robotic arm for waste collection is being introduced as one of the environmentally friendly solutions. In this paper, the emphasis is on waste manipulation using a robotic arm. The xArm6 robotic arm from the South Korean company UFACTORY is used to create the waste manipulation system. For the detection and rough spatial localization of waste, the YOLO v5 convolutional neural network is used, which obtains the necessary 2D and 3D visual data from the Intel RealSense D455 stereo vision system. The full management support of the system was developed in the robot operating system (Robot Operating System - ROS) using Python and C++ programming languages. The obtained data from the stereovision system, especially the point clouds, must be processed. The processing of 3D data in this paper was done using the open3D program library. The software and all the data acquisition from the environment and the processing and planning of all system movements are made on the NVIDIA microprocessor. The chosen microprocessor is the Jetson Xavier NX. At the beginning of the work, the experimental set-up of the system and each of its components is described, and a deeper insight into the problems of the work is given. After a more detailed description of the problem and the system, the method of integrating different algorithms into the system is described to achieve the detection, localization, segmentation and clustering of waste in space and overlapping clusters with a geometric primitive. The geometry primitive is later added to the ROS MoveIt scene where it is manipulated. In the end, the task is rounded off with the analysis of the obtained results and the evaluation of the experimental setup of the system, after which the corresponding conclusion is obtained.

Keywords: Robotic arm, MoveIt, ROS, 3D vision system, open3D, YOLO, object detection, stereo vision system

1. UVOD

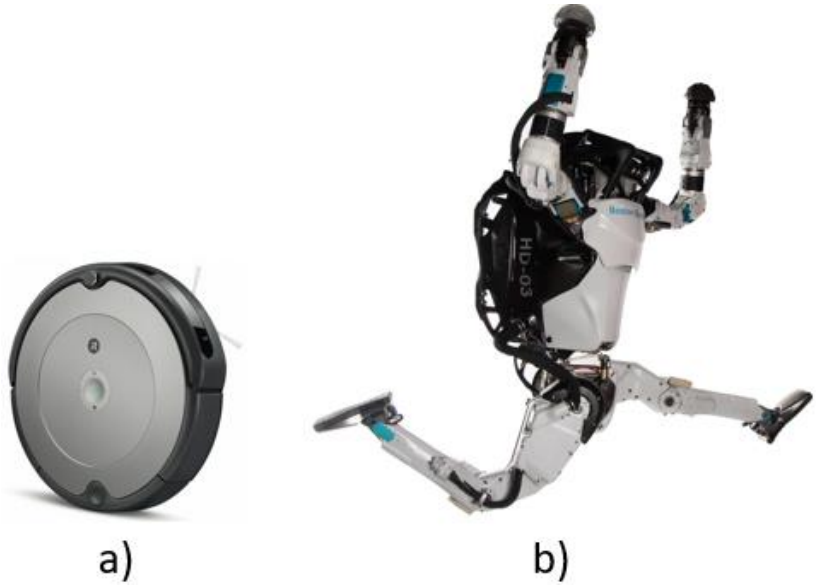
S obzirom na porast zagađenja svjetskih oceana i mora širom planete zemlje i velike fluktuacije otpada u oceanima i morima dolazi do izbacivanje otpada na obale. Posljednjih 40 godina nastale su mnogobrojne udruga i inicijativa koje se trude sakupiti otpad koji oceani i mora nanesu na obale. Posao sakupljanja otpada je vrlo mukotrpan i iscrpljujući proces koji je potrebno učestalo ponavljati kako bi se obale održale čistima. Na slici ispod prikazana je otpad koji je more/ocean izbacilo na kopno.

Slika 1.1 Prikaz zagađene obale [1]

Zbog izrazito velike količine otpada koji se nalaze u morima i oceanima većina država nema riješeno pitanje kontinuiranog sakupljanja smeća s obale te održavanjem istih čistima. Upravo iz tih razloga su i nastale gore navedene udruge i inicijative koje se trude doprinijeti očuvanju obala. Veliki dijelovi obala su nenaseljeni i teško pristupačni ljudima s kopnene strane čime se nameće potreba za autonomnim čišćenjem obala sa što manje ljudske intervencije. Zbog gore opisanih problema učestalog čišćenja obala od otpada i rapidnim razvojem tehnologije u zadnjih 20 godina nameće se kao ekološki prihvatljivo rješenje koristiti autonomne robotske sustave u ovakvom zadatku.

1.1. Autonomni robotski sustavi

Autonomni robotski sustavi su inteligentni mehanički sustavi sposobni izvoditi zadatke u svijetu oko njih samostalno, bez izričite ljudske kontrole. U današnje vrijeme može se primijetiti sve veća prisutnost autonomnih robotskih sustava u svakodnevnom životu. Posljednjih godina otvoren je veći broj restorana u svijetu u kojima roboti poslužuju ljude. Današnje moderne tvornice u potpunosti su automatizirane i autonomne te potreba za ljudima u njima je skoro ne postojeća. Primjena u današnjem svijetu autonomnih robotskih sustava vrlo je raznolika od Roombe, autonomnog robota čistača do humanoidnog robota Atlasa od tvrtke Boston Dynamics. Na slici ispod prikazan su jedan i drugi robot.

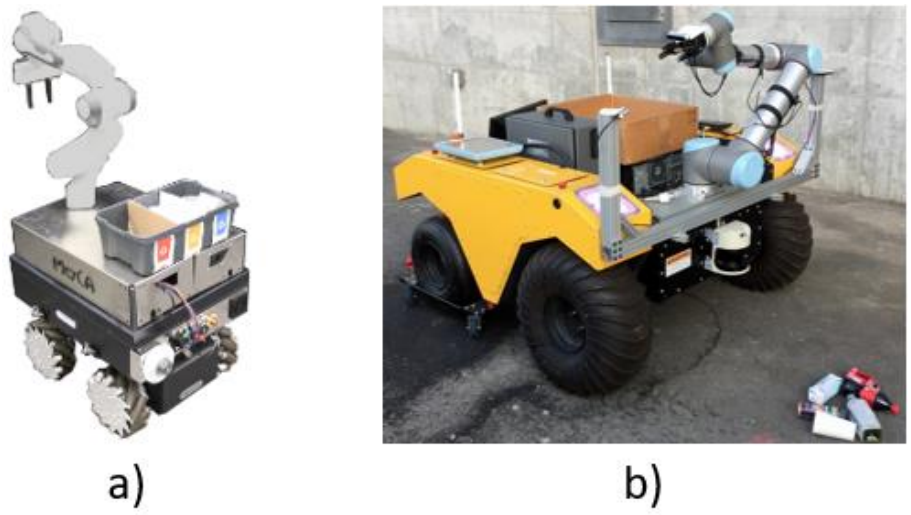


Slika 1.2 a)Roomba [2], b)Boston Dynamics Atlas [3]

Upravo zbog strelovitog razvoja autonomnih robotskih sustava i njihovoj sve jednostavnoj implementacijije u ovom radu razvijen prototip sustava za rješavanje aktualnog problema sakupljanja otpada na obalama mora i oceana. Razvoj takvog sustava je vrlo kompleksan zadatak koji je potrebno podijeliti u manje smislene cjeline. Zbog problematike zadatka nameću se neke osnovne funkcionalnosti koje autonomni robotsku sustav mora sadržavati u sebi.

1.2. Izazovi i pristup zadatku

Prije svega, autonomni robotski sustav mora biti mobilan odnosno mora se moći kretati u prostoru i mora imati mogućnost manipulacije otpadom. Navedeni zahtjevi nameću određena ograničenja prilikom konstruiranja ovakvog sustava. Sustav mora biti relativno lagan kako bi nesmetano mogao ići po plaži i sakupljati otpad bez da dođe do zapinjanja u pijesku ili šljunku. Nadalje, mora biti vrlo fleksibilan prilikom manipulacije otpadom kako bi mogao doći do svih dijelova plaže i pokupiti iste te mora imati prostor unutar ili na sebi za odlaganje sakupljenog otpada. Do sada neki od razvijenih autonomnih sustava u laboratorijima na ovu problematiku prikazani su na slikama 1.3.



Slika 1.3 a)Mobilna platforma s robotskom rukom Panda [4], b) Mobilna platforma s robotskom rukom UR5 [5]

Može se primijetiti da se oba mobilna autonomna robotska sustava sastoje od dvije cjeline. Prva cjelina je mobilna platforma a druga cjelina je robotska ruka na mobilnoj platformi čime zadovoljavaju sve potrebne zahtjeve za samostalno sakupljanje otpada. Zbog kompleksnosti samog autonomnog sustava, gore navedenog u ovom diplomskom radu, će se razraditi i evaluirati autonomni sustav za rukovanje otpadom uz pomoć robotske ruke. Prilikom odabira robotske ruke u obzir se mora uzeti njezina masa i masa pripadajuće upravljačke jedinice. Industrijske robotske ruke su velikih masa te su njihove upravljačke jedinice

neprikladne za ugradnju na mobilne platforme zbog svojih dimenzija. Iz tog se razloga trebaju se razmotriti alternativne robotske ruke kako bi se kasnije mogle implementirati na autonomnu mobilnu platformu. Robotska ruka treba imati minimalni doseg od 0,5 m i nosivost od minimalno 1 kg. Robotska ruka treba unutar sebe sadržavati senzore sile ili momenta kako bi mogla raditi u okolini ljudi. Senzori će osiguravati ako slučajno dođe do kolizije s čovjekom kako bi robotska ruka stala na vrijeme i spriječila povrede čovjeka. Na slici ispod prikazana je Franka Emika Panda, kao jedna od mogućih robotskih ruku koja zadovoljava sve navedene kriterije.

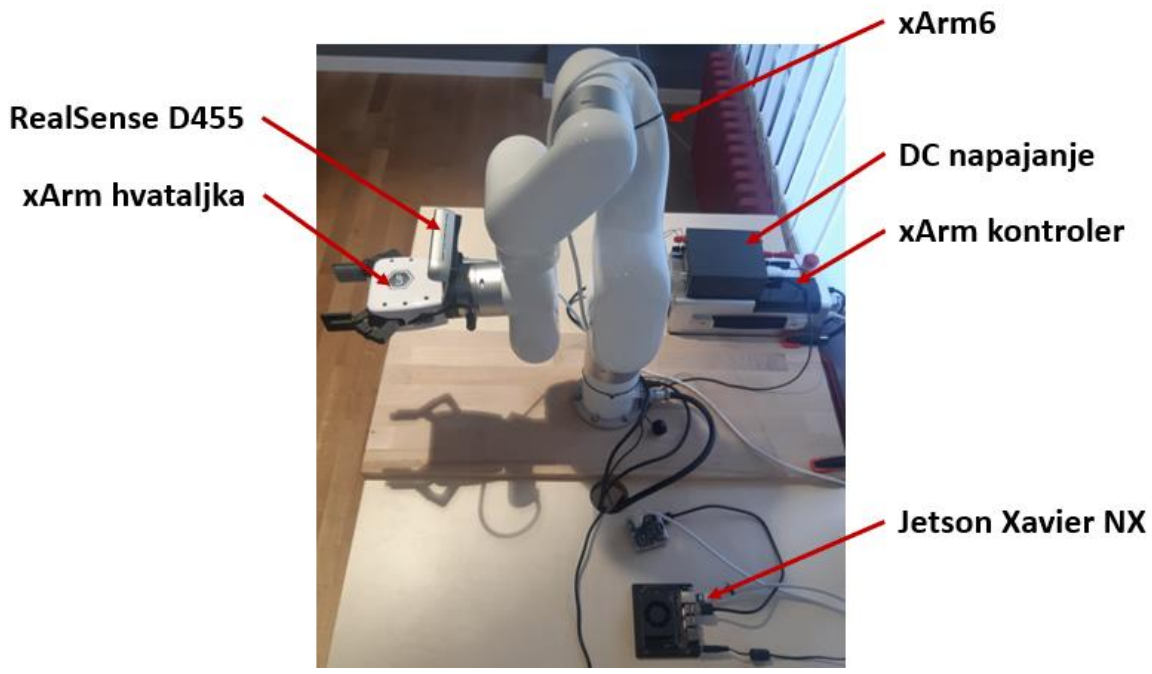
Slika 1.4 Franka Emika Panda [6]

Razvoj autonomnog robotskog sustava za manipulaciju otpadom moguće je razviti na različitim softverskim platformama od kojih Robotski operacijski sustav (eng. *Robot Operating System* – ROS) predstavlja standard koji je opće prihvaćen u akademskoj zajednici, a sve se više koristiti i u industriji. ROS je operativni sustav koji za razliku od standardnih operativnih sustava se pokreće preko Linux operativnog sistema što mu daje veliku fleksibilnost u odabiru hardvera na kojima će biti pokrenut. On pruža okolinu za razvoj modularne upravljačke programske podrške, komunikacijsku infrastrukturu i otvorenu biblioteku algoritama.

Sustav mora u sebi sadržavati računalo koje radi na Linux operacijskom sustavu na kojem će se pokretati ROS. Unutar sustava implementiran je 3D vizijski sustav iz kojeg se dobivaju potrebni podatci o okolini. Iz dobivenih podataka radi se klasifikacija smeća u okolini oko sebe pomoću neuronske mreže. Klasificirano smeće potrebno je lokalizirati u prostoru. Poslije segmentacije smeća u okolini i lokalizaciji u prostoru potrebno je isplanirati način rukovanja i izuzimanja smeća. Izuzimanje smeća sastoji se od dva dijela. Prvi dio se odnosi na planiranje putanje do smeća te izvršavanje potrebne inverzne kinematike robotske ruke kako bi pozicionirali ruku u traženu poziciju te nakon toga određivanje prilazne putanje prema objektu, izuzimanju objekta i izlazne putanje od objekta. Nakon što se objekt nalazi u hvataljci robotske ruke, potrebno je uzeti u obzir u daljnjoj inverznoj kinematici njegovu dimenziju kako ne bi došao u koliziju s rukom. Na kraju, objekt je potrebno odložiti na predviđeno mjesto.

2. EKSPERIMENTALNI POSTAV SUSTAVA ZA MANIPULACIJU OTPADOM

Prilikom planiranja i definiranja eksperimentalnog postava u obzir su uzeti svi navedeni zahtjevi iz prvog poglavlja. Sustav prije svega mora biti fleksibilan, odnosno, mora imati mogućnost jednostavne implementacije na bilo koju mobilnu platformu te mogućnost u prilagođavanju okoline oko sebe. Također, sustav mora biti upotrebljiv u okolini oko ljudi zbog njegove moguće suradnje s ljudima prilikom izvršavanja zadatka ili izvršavanja zadataka oko ljudi. Zbog gore navedenih razloga izabrana je kolaborativna robotska ruka xArm6 i hvataljka xArm Gripper od tvrtke UFACTORY(opisano u potpoglavlju 2.1.). U svrhu prikupljanja podataka iz okoline odabrana je 3D kamera Intel RealSense D455(opisano u potpoglavlju 2.3.). Odabrano računalo na kojem će se raditi operativni sustav Linux, na kojem će se također pokretati ROS, je Jetson Xavier NX (opisano u potpoglavlju 2.2). Za segmentaciju i klasifikaciju otpada iz dobivenih podataka iz kamere koristit se YOLO v5 konvolucijska neuronska mreža koja je razvijena izvan ovog diplomskog rada. Na slici ispod prikazani su svi dijelovi eksperimentalnog postava sustava za rukovanje otpadom.

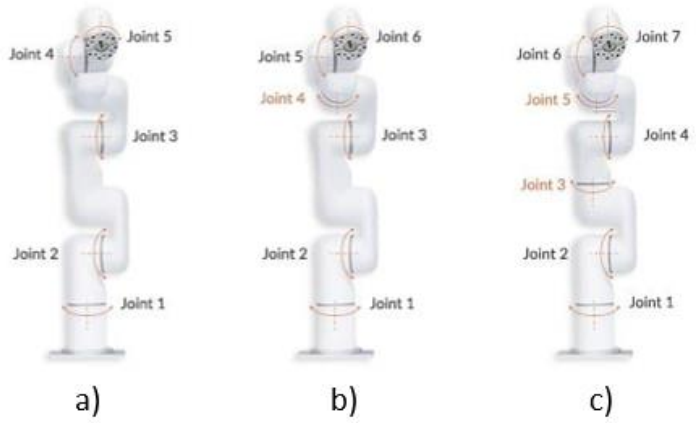


Slika 2.1 Eksperimentalni postav sustava za manipulaciju otpadom

2.1. Kolaborativna robotska ruka s hvataljkom

Priroda ovog zadatka zahtjeva veliku fleksibilnost sustava, malu masu, moguću suradnju s čovjekom, laganu integraciju u autonomne mobilne robote, relativno veliki opseg rada sustava i mogućnost manipulacijom objektima. Upravo iz ovih razloga odabrana je kolaborativna robotska ruka xArm6 tvrtke UFACTORY. Tvrtka UFACTORY nastala je 2014. godine te veliku većinu svojih sredstava potrebnih za razvoj robotskih ruku dobivaju preko stranice KICKSTARTER. Ova stranica služi kao platforma za pronalazak investitora kojima se prezentira ideja i plan razvoja projekta. Od svojeg osnutka napravili su već tri uspješna projekta od kojih je projekt uArm Swift najfinanciraniji robotski projekt na stranici ikada.

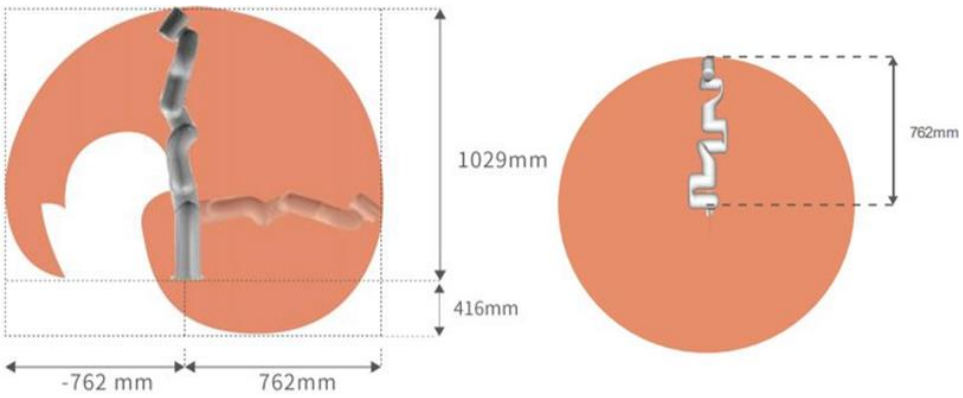
Nadalje, broj u nazivu imena ruke označava broj stupnjeva slobode gibanja ruke. Postoje još neke verzije s pet i sedam stupnjeva slobode. Na slici 2.2. prikazane su sve navedene ruke od tvrtke UFACTORY iz kategorije xArm.



Slika 2.2 a) xArm5, b) xArm6, c) xArm7 [7]

Ruka je napravljena od karbona što ju čini izuzetno laganom i krutom. Masa robotske ruke s upravljačkom jedinicom iznosi 13,8kg što je za usporedbu približno 50% lakše od Franka Emika Panda ruke čija masa iznosi 25kg.

Nosivost ruke je izuzetno velika u usporedbi s njenom masom te iznosi 5kg. Robotska ruka ima doseg od 762mm i vertikalni doseg od 1029mm. Na slici ispod prikazan je radni prostor robotske ruke.



Slika 2.3 Radni opseg xArm6 robotske ruke [8]

Robotska ruka može se programirati na različitim operacijskim sustavima od Windows-a do Linuxa-a . Za robotsku ruku postoji i GUI aplikacija preko koje se može upravljati i programirati u C++ ili Python-u. Na slici 2.4. prikazano je sučelje aplikacije.

Slika 2.4 GUI aplikacija za robotsku ruku [8]

Za robotsku ruku postoji podrška na GitHub-u [9] te za njeno upravljanje uz pomoć ROS-a su napravljeni inicijalni paketi unutar ROS-a. Postoji i URDF datoteka uz pomoć koje je opisana cijela robotska ruka zajedno sa svim međusobnim odnosima unutar ruke.

Nadalje, odabrana hvataljka (Slika 2.5) za robotsku ruku je od iste tvrtke UFACTORY. Ruka dolazi već s ugrađenom hvataljkom i potrebnim konektorima kako bi se mogla spojiti na xArm6. Hvataljka ima raspon širine hvatanja od 0 do 86mm, te masu od 0,8kg. Ona se može programirati po brzini i pozicija, a u zatvorenoj petlji konstantno vraća svoju trenutnu poziciju. Maksimalna struja koju hvataljka može doseći je 1,5A te pri toj struji može ostvariti silu stezanja od 30N. Komunikacija s hvataljkom izvršava se preko RS-485 protokola.

Slika 2.5 xArm Gripper [10]

2.2. Jetson Xavier NX

Odabrano računalo na modulu (eng. SOM) za upravljanje robotskom rukom je Jetson Xavier NX. Ovo računalo je odabrano zbog svoje kompaktnosti i specifikacija. Računalo zajedno sa pripadajućom razvojnom pločicom je dimenzija 90x105x35mm što ga čini kompaktnim u odnosu na njegove performanse. Također podržava CUDA API što je bitno za brzu obradu slika i efikasno iskorištavanja GPU-a na računalu. Na slici 2.6. prikazano je odabrano računalo.

Slika 2.6 Jetson Xavier NX [11]

CUDA je razvijen od strane Nvidia-e te je upravo zbog CUDA sama obrada slika puno brža što ovo računalo čini izvrsnim za implementaciju neuronskih mreža i strojnog učenja. Za razliku od Jetson Nano, namijenjeno je za industrijsku uporabu te su u tablici 1. prikazane njegove specifikacije.

Tablica 1 Karakteristike Jetson Xavier NX [12]

CPU	6-core NVIDIA Carmel ARM®v8,2 64-bit CPU 6MB L2 + 4MB L3
GPU	384-core NVIDIA Volta™ GPU s 48 Tensor Cores
GPU maksimalna frekvencija	1100 MHz
RAM memorija	8 GB 128-bit LPDDR4x @ 1866MHz, 59,7GB/s
Memorija	16 GB eMMC 5,1
CSI kamera	Do 6 kamera(36 moguće uz pomoć virtualnih portova)
Ethernet komunikacija	10/100/1000 BASE-T Ethernet
HDMI	DA
Napajanje	19V

2.3. Stereovizijski sustav

Zbog potrebe za detekcijom objekata u prostoru i njihovom lokalizacijom, važno je integrirati u eksperimentalni postav vizijski sustav koji je u mogućnosti pružiti 3D podatke. Iz tog razloga je odabran Intel RealSense stereo sustav. Intelova RealSense dubinska kamera D455

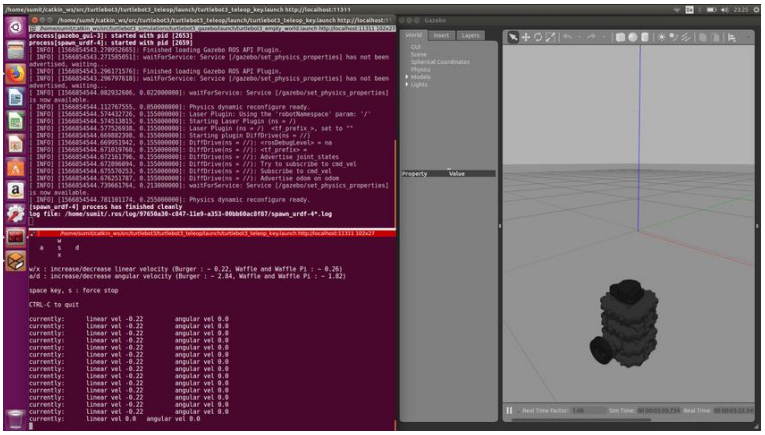
omogućava dobivanje 3D informacija iz scene, što omogućava razvijanje softvera za robotsku navigaciju u prostoru, prepoznavanje i lokalizaciju objekata u prostoru te mnogobrojne druge primjene. Kamera se napaja i šalje informacije uz pomoć USB-C-a, a u sebi sadrži i vizijsku procesorsku jedinicu koja obrađuje sve informacije dobivene podatke iz senzora. Kamera u sebi ima integriran i IMU senzor, te je njeno vidno polje 86 stupnjeva. Nadalje, dubinu računa pomoću stereo slike, odnosno, uz pomoć dvije kamere koju vizijski sustav ima na sebi. Isto tako, sustav ima i mogućnost stereo vida u infracrvenoj zoni i kameru za dobivanje slike u boji. Vizijski sustav ima maksimalni doseg od 20m i moguće ga je programirati u različitim programskim jezicima od kojih su najčešće korišteni C++, Python i C#. Vizijski sustav također ima razvijene pakete u ROS-u [13] što ga čini idealnim za implementaciju u eksperimentalni postav sustava. Stereokamera se montira na vrh robotske ruke kako bi bila u mogućnosti detaljnije snimati površinu ispod sebe te je držač za sustav i sama kamera uneseni u opis robota kako ne bi došlo do kolizije robotske ruke i vizijskog sustava. Opis robota napravljen je pomoću URDF datoteke (eng. *Unified Robotics Description Format*) u kojoj je opisana cijela robotska struktura, donosno njena masa, dimenzije dijelova, dopuštenih stupnjeva slobode gibanja ruke, kolizije svakog elementa ruke i svi pripadajući senzori na ruci. Na slici ispod prikazana je odabrani vizijski sustav.

Slika 2.7 Intel RealSense D455 [14]

2.4. Robotski operacijski sustav

Za izradu jednog robotskog sustava potreban je softver pomoću kojeg se hardver može upravljati te je u ovom diplomskog radu odabran robotski operacijski sustav. ROS je fleksibilno razvojno okruženje otvorenog koda (eng. *open source*) koje se upotrebljava za razvoj robotskih sustava. Nastao je na sveučilištu Stanford davne 2007. godine iz potrebe za komunikacijom

različitih laboratorija unutar fakulteta. ROS se pokreće iz Linux-a te nakon pokretanja ponašao se identično kao i bilo koji drugi operacijski sustav. Na slici 2.8. prikazana je razvojna okolina robotskog sustava u ROS-u pomoću virtualne simulacije u Gazebo-u.



Slika 2.8 Razvojna okolina ROS-a [15]

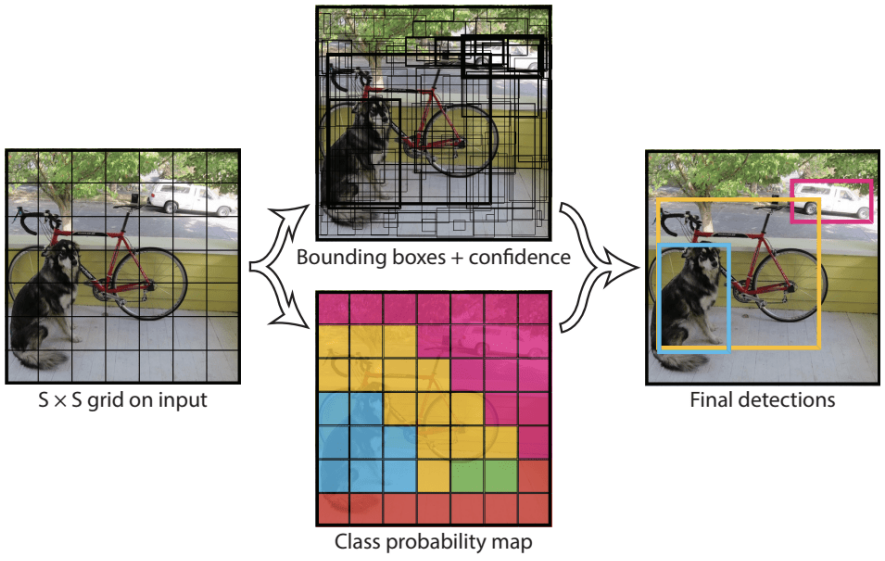
ROS je odabran zbog svoje dvije glavne značajke, a to su mogućnost korištenja velikih količina već unaprijed napisanih biblioteka raznolikih algoritama i izvrsna među komunikacija unutar operacijskog sustava različitih elementa robota. On pruža uslugu koju bi očekivali od bilo kojeg drugog operativnog sustava, uključujući i apstrakciju hardvera, mogućnost upravljanja uređajima na nižoj razini, implementaciju često korištenih funkcionalnosti u robotici (navigacija, odometrija, fuzija senzora, itd.), komunikaciju različitih dijelova sustava, njihovih procesa i upravljanje softverskim paketima. Također, sadrži alate i programske knjižnice za održavanje, simulaciju realnosti, vizualizaciju, izgradnju, pisanje i pokretanje kodova na više računala[16].

Glavni cilj ROS-a je podržati ponovnu upotrebu koda u istraživanju i razvoju robotike. Osnovna gradivna cjelina ROS-a je paket (eng. Package) koji služe kao jedna cjelina u koju se spremaju napisani algoritmi sa svim potrebnim informacijama za ponovnu upotrebu i instalaciju paketa. U ovom diplomskom radu koristit se ROS Melodic distribucija, ponajviše zato što je to zadnja ROS1 distribucija ROS-a podržana od strane Nvidia-e.

2.5. Neuronska mreža YOLO v5

Detekcija objekata na slici stavka je bez koje robotski sustav ne bi bio u mogućnosti spoznati s kojim objektima je potrebno manipulirati tj. ne bi mogao detektirati otpad koji je potrebno sakupiti. Bitna stavka sustava je mogućnost detekcije, klasifikacije i segmentacije objekata u stvarnom vremenu i iz bilo koje pozicije 3D vizijskog sustava. Zbog kompleksnosti problema, rješenje se traži u postojećim algoritmima i pristupima ovakvim problema koji daju pouzdane rezultate te se izabire korištenje neuronskih mreža. Za obradu dobivenih slika iz vizijskog sustava neuronskim mrežama, najefikasnije je korištenje onih koje se temelje na principu konvolucije. Konvolucija je matematička metoda za dobivanje bitnih značajki iz slike. Postoji puno konvolucijskih neuronskih mreža koje su često u potpunosti različitih struktura, samim time i princip po kojima predviđaju rezultate na još nepoznatim podatcima.

Odabrana konvolucijska neuronska mreža za ovaj zadatak je YOLO v5 [17]. YOLO (eng. You Only Look Once) neuronska mreža nastala relativno nedavno, 2020 godine, te radi segmentaciju slike na jednolike dimenzije te se nakon toga određuju područja na kojima bi se mogli nalaziti dijelovi definiranih objekata. Poslije određivanje područja na kojima bi objekt mogao biti, izvršava se obrada podataka na tim područjima i na kraju utvrđuje ukupno područja na kojima se objekt nalazio. Na slici ispod prikazan je princip detekcije objekata opisanom mrežom.

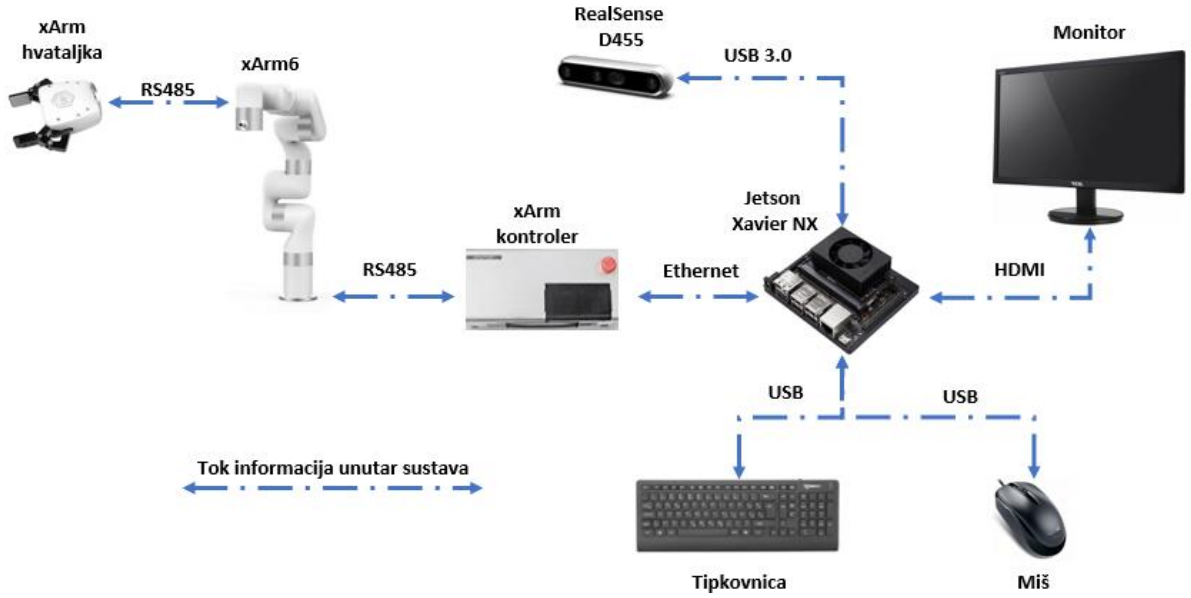


Slika 2.9 Detekcija objekata YOLO neuronskom mrežom [18]

YOLO konvolucijska neuronska mreža jednostavna je za trenirati, u mogućnosti je detektirati objekte u stvarnom vremenu te je već implementirana u ROS-u pomoću ROS paketa darknet_ros [19]. Napisani ROS paket za YOLO neuronsku mrežu koristi se Darknet open-source sučelje za neuronske mreže napisanom u C i CUDA-i što mu daje izrazito veliku brzinu detekcije objekata i niskoj potrošnji resursa računala [20].

2.6. Komunikacija između dijelova eksperimentalnog postava sustava

Sustav se sastoji od više različitih komponenti te ga je potrebno povezati u jednu cjelinu kako bi se omogućio protok informacija. Zato se prilikom odabira svih komponenti unutar sustava moralo uzeti u obzir pitanje " mogu li oni svi međusobno komunicirati?". Kao što je prije navedeno, velika prednost ROS-a je mogućnost prilagođavanja različitim hardverima i slanje unificiranih poruka kroz sustav. Također na eksperimentalnom postavu sustava spojeni su miš, monitor i tipkovnica kako bi se informacije mogle vizualizirati i zadavati u eksperimentalnoj fazi (slika 2.10)

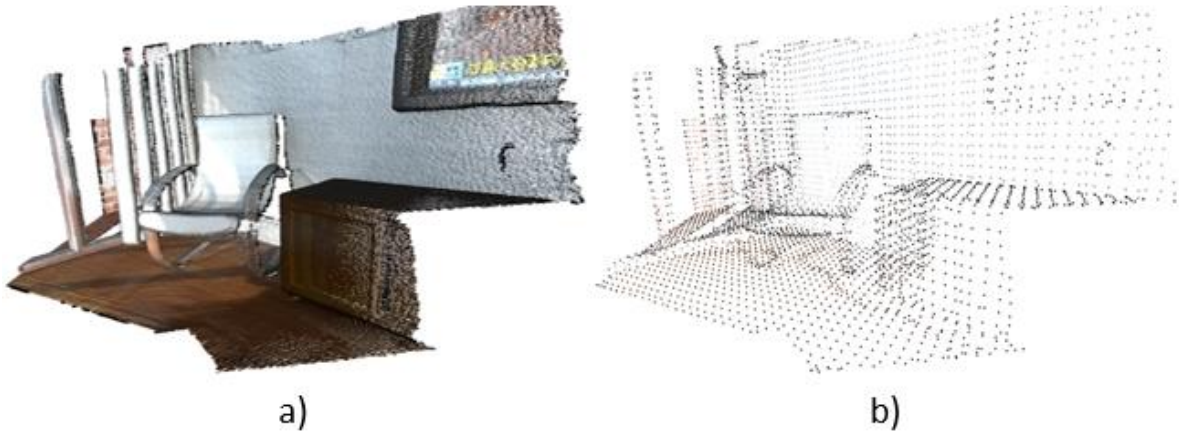


Slika 2.10 Shematski prikaz toka informacija unutar sustava

2.7. Open3D programska knjižnica

Eksperimentalni postav u sebi sadrži stereovizijski sustav čiji se podatci prikupljaju u obliku oblaka točaka odnosno 3D podataka. Za obradu 3D podataka postoje trenutno dvije velike programske knjižnice, PCL programska knjižnica i Open3D programska knjižnica. Obje knjižnice su otvorenog koda te su podržane od strane zajednice i koriste se u različitim projektima i istraživanjima. U ROS-u se više koristi PCL programska knjižnica, ali je za ovaj diplomski rad odlučeno korištenje Open3D programske knjižnice. Open3D u sebi sadrži manju količinu algoritama, ali je temeljitije dokumentiran te su svi njeni algoritmi podržani u C++ ili Pyhton programskom jeziku.

Zbog toga što je cijeli Open3D API podržan i opisan u Pythonu razvoj softvera za manipulaciju 3D podatcima je brži i jednostavniji. Algoritmi implementirani unutar Open3D knjižnice obuhvaćaju zapisivanje, čitanje, filtriranje, registraciju, vizualizaciju, segmentaciju te mnoge druge operacije nad oblacima točaka. Primjer jedne takve operacije nad oblakom točaka prikazan je na slici ispod.



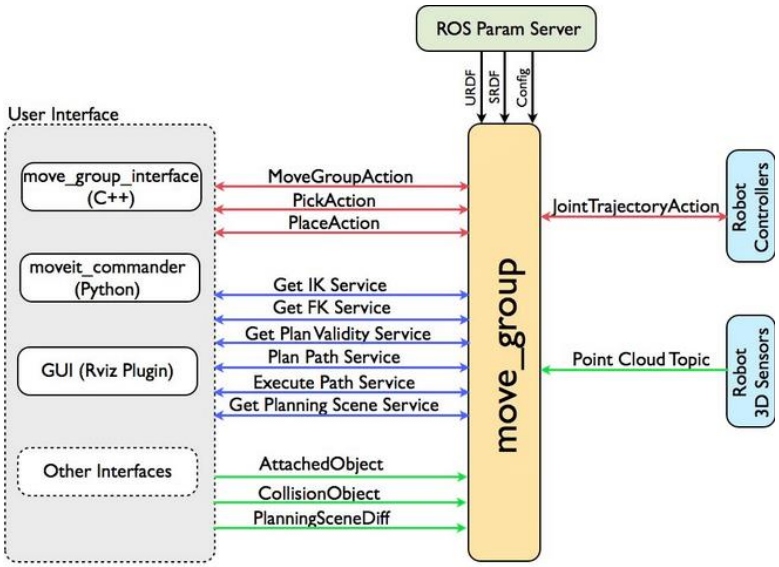
Slika 2.11 a) Učitani oblak točaka u Open3D [21], b) Obrađeni oblak točaka unutar Open3D-a[21]

3. IMPLEMENTACIJA ROS-A U EKSPERIMENTALNI POSTAV SUSTAVA

Upravljanje robotskom rukom, izračun njezine inverzne i direktne kinematike, izračun dinamike ruke i regulacije svakog električnog motora unutar svakog zgloba i paralelni rad više zglobova izazovan je zadatak. Upravo iz tih razloga teži se koristiti gotova i provjerena ROS rješenja s kojima se proces eksperimentiranja i razvijanja visoko složenih algoritama eksponencionalno ubrzava. Slogan ROS-a je "prestani ponovno izmišljati kotač" (eng. *stop reinventing the wheel).* Upravo ponovno izmišljanje kotača je glavni ubojica novih inovativnih projekata. U ovom poglavlju detaljno će se opisati proces implementacije ROS paketa za upravljanje robotskom rukom, detekcijom i lokalizacijom objekata.

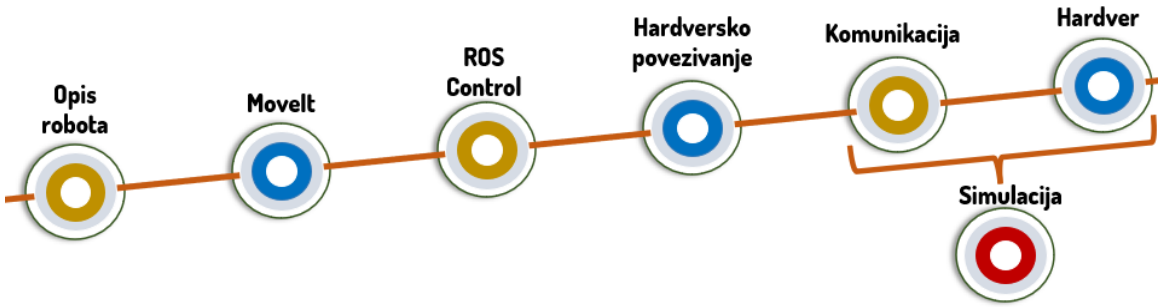
3.1. MoveIt

MoveIt je ROS meta-paket koji je rezultat zajedničkog napora velike internacionalne zajednice i više organizacija. MoveIt je standard u ROS zajednici za razvoj naprednih primjena i simulacija rada robotskih ruku, planiranje visoko sofisticiranih kretnji, analiziranje i ulaženje u interakciju s okolinom pomoću generatora hvatanja (eng. *grasp generation*). Pomoću paketa riješeni su izazovi poput izračunavanja inverzne i direktne kinematike robotskih rukom, 3D percepcije prostora uz pomoć OctoMapa, mogućnost detekcije kolizije uz pomoć oblaka točaka i definiranih geometrija. Arhitektura sustava na visokoj razini prikazana je na slici. Čvor (eng. *node*) koji je zadužen za svu interkaciju s MoveIt-om zove se *move_group*. Bitno korisničko sučelje za prihvat i manipulaciju objektima je *moveit_commander* interface pomoću kojega se u prostor MoveIt-a mogu dodavati različiti prizmatični objekti, mreže (eng. *mesh*) ili drugi oblici za manipulaciju. Unutar sučelja MoveIt će prepoznavati razliku između objekta koji je uhvaćen u ruku i koji se samo nalazi u prostoru.



Slika 3.1 Prikaz arhitekture move_group čvora [22]

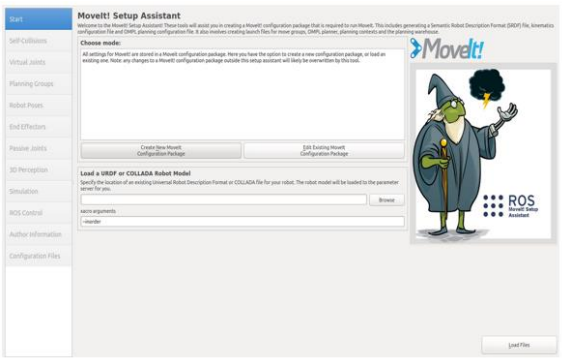
Za implementacija MoveIt-a u robotski sustav potrebno je napraviti sljedeće korake prikazane na slici ispod.



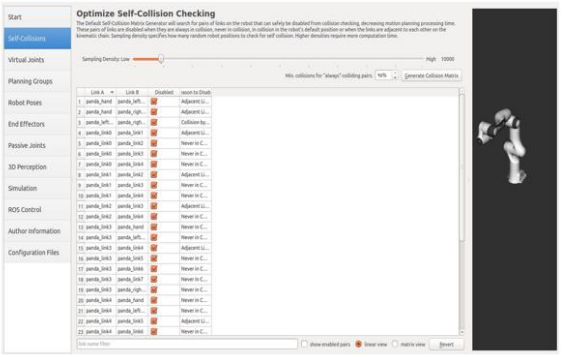
Slika 3.2 Implementacija MoveIt-a s robotoskom rukom

Prvo je potreban opis robota koji se u ROS-u radi uz pomoć URDF (eng. Unified Robotics Description Format) datoteke koja je formata XML. U URDF datoteci određena je boja svakog dijela robota, povezanost, inercija svakog dijela robota, masa, kolizije, njihovi međusobni odnosi i ograničenja svakog zgloba ako ih imaju.

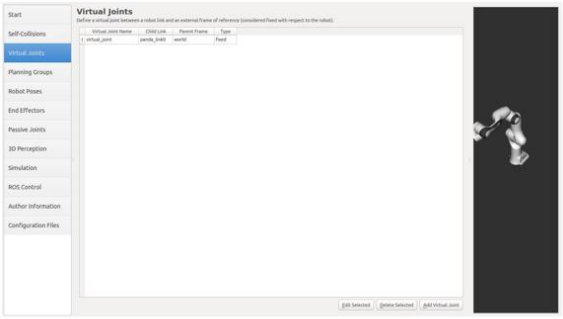
Potrebno je opisani robot unijeti u MoveIt setup Assistent i definirati sve potrebne parametre. Na slikama ispod opisan je postupak opisa robota u MoveIt GUI-u



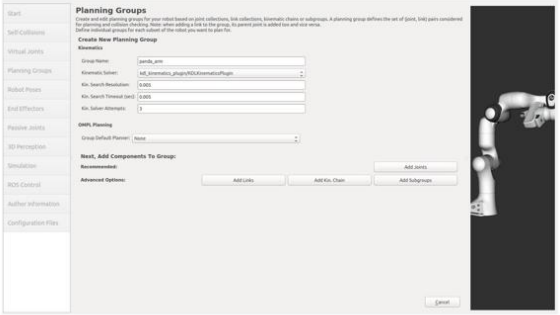
a)



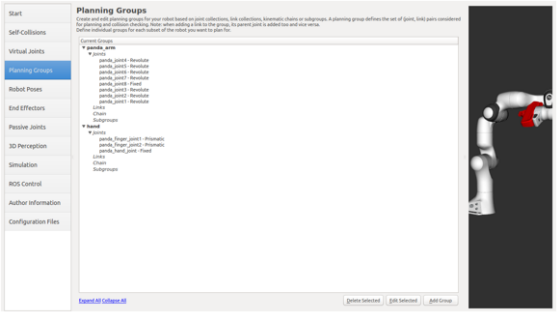
b)



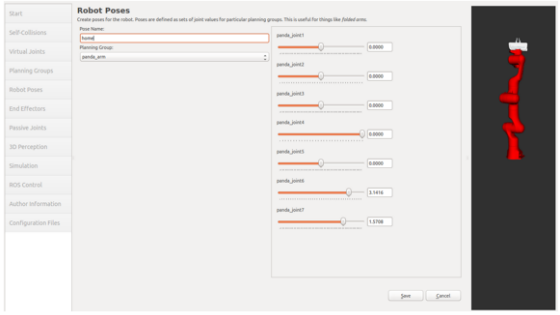
c)



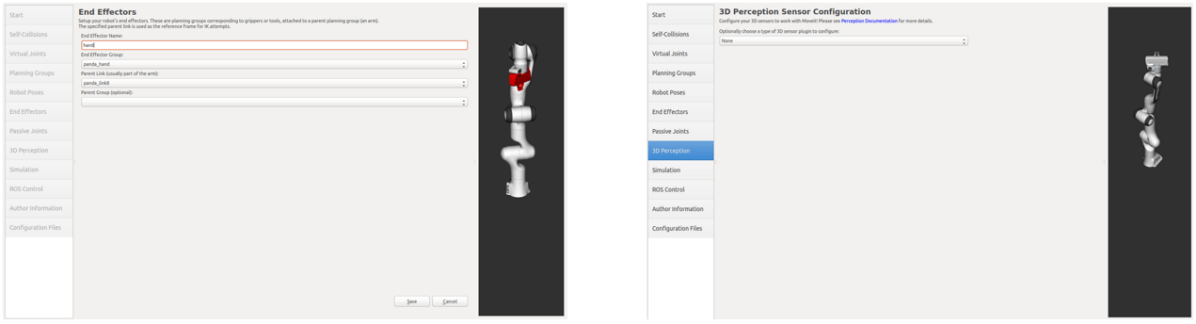
d)



e)



f)



g) h)

Slika 3.3 a) Učitavanje URDF datoteke u MoveIt setup Assistant, b) Izrada Matrice kolizija različitih dijelova unutar robota, c) Povezivanje prvog zgloba robota sa svijetom oko sebe odnosno fiksiranje u prostoru, d) Definiranje različitih grupa robota i softvera za rješavanje kinematike i dinamike robota zajedno sa svim potrebnim parametrima, e) Definiranje grupa za planiranje kretnji, f) Spremanje predefiniranih pozicija robotske ruke, g) Definiranje hvataljke, h) Definiranje vizijskog sustava i podešavanje svih potrebnih parametara, i) Definiranje kontrolera koji će se koristiti za upravljanje električnim motorima na zglobovima robotske ruke, j) generiranje svih potrebnih datoteka za korištenje MoveIt-a

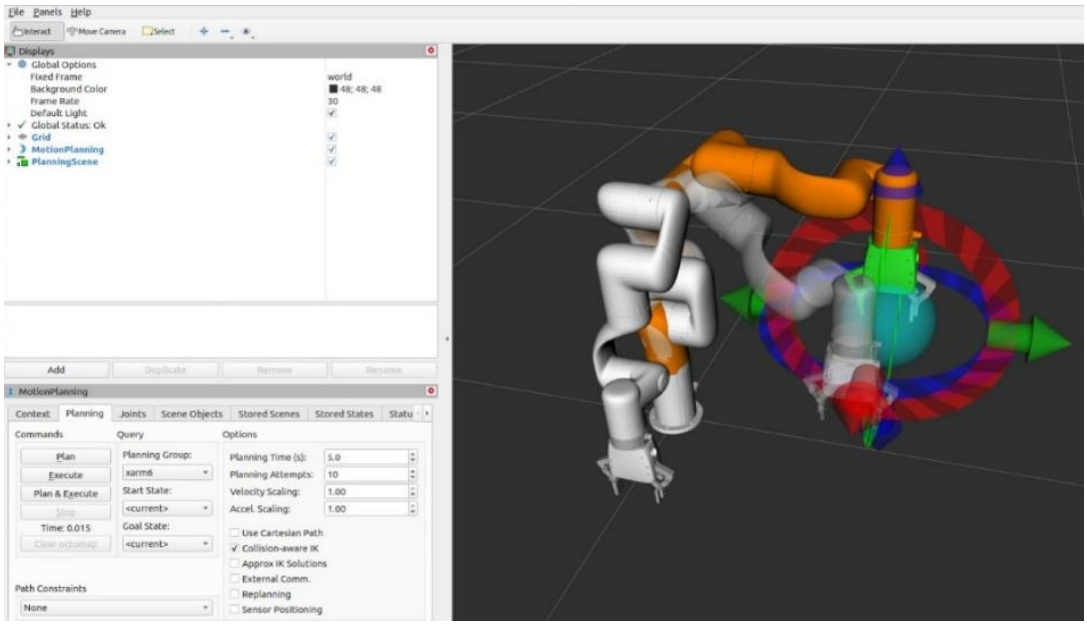
Nakon implementacije MoveIt-a, sljedeći korak je hardversko povezivanje tj. ROS čvor zaprima poruku o stvarnom stanju robota, a hardver će preko komunikacijskog protokola poslati željeno stanje zglobova.

Zadnja dva ključna koraka su komunikacija s hardverom. Nakon procesuiranja svih podataka na kompjuteru koje smo dobili od robota i ulaznih podataka od korisnika putem etherneta, CAN-a ili nekog drugog serijskog protokola šaljemo hardveru, odnosno, mikrokontroleru koji onda izvršava pozicioniranje zglobova. Postoji mogućnost zamjene zadnja dva koraka sa simuliranim robotom . Jedna od takvih simulacija zove se GAZEBO s kojom je moguće testirati različite algoritme bez potrebe za stvarnim hardverom.

3.2. Sučelje za upravljanje robotskom rukom i vizualizacija podataka

Upravljanje robotskom rukom odnosno njezino pomicanje u prostoru moguće je napraviti bez pisanje koda pomoću RViz-a i pokretanja svih potrebnih čvorova generiranih od strane MoveIt-a. Upravljanje robotskom rukom unutar RViz-a sučelja moguće je pomoću

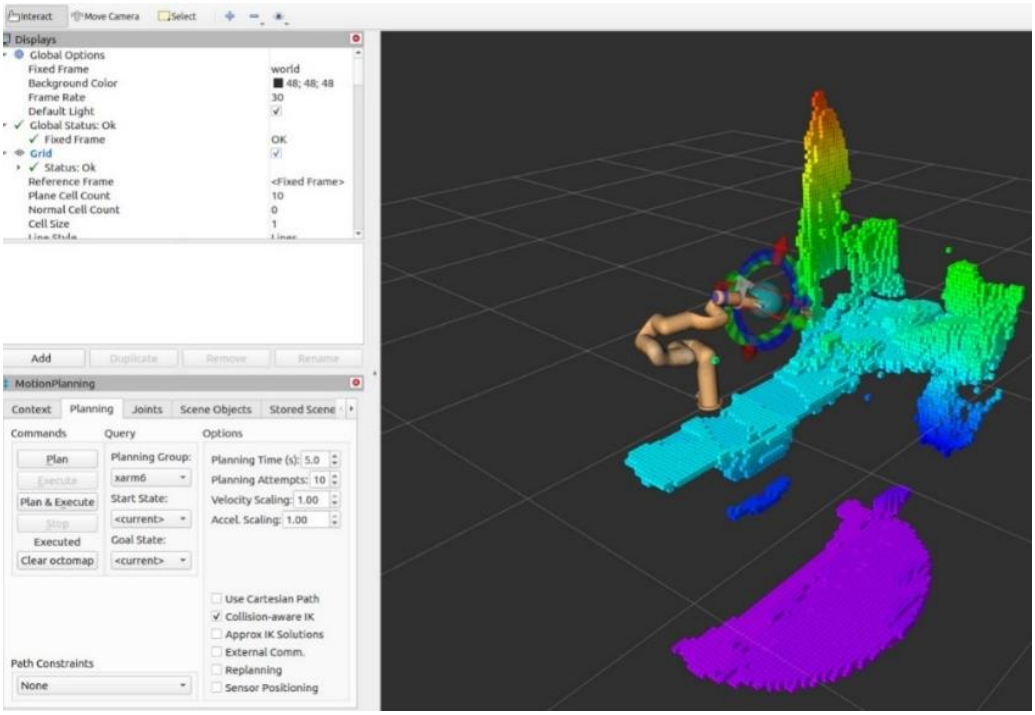
MoveIt-ovog dodatka (eng. *plugin*) *MotionPlanning*. Korištenjem sfere u prostoru moguće je pomicati robota u željenu poziciju što se prikazuje u narančastoj boji robota. Nakon toga softver samostalno računa potrebnu inverznu kinematiku kako bi došao u željenu poziciju zajedno s definiranim OMPL planerom za planiranje optimalne trajektorije iz početne pozicije u konačnu. OMPL (eng. *The Open Motion Planning Library*) sastoji se od kompleta algoritama temeljenih na uzorkovanju (eng. sampling-based) planiranih kretnji. Na slici ispod prikazana je vizualizacija podataka pomoću RViz-a zajedno s *MotionPlanning* dodatkom.



Slika 3.4 RViz vizualizacija MotionPlanning plugin-a

Punom bojom prikazana je trenutna pozicija robotske ruke, narančastom bojom definirana je željena pozicija robotske ruke i prozirno sivom bojom prikazuje se u pokretu trajektorija dolaska iz početne točke u krajnju.

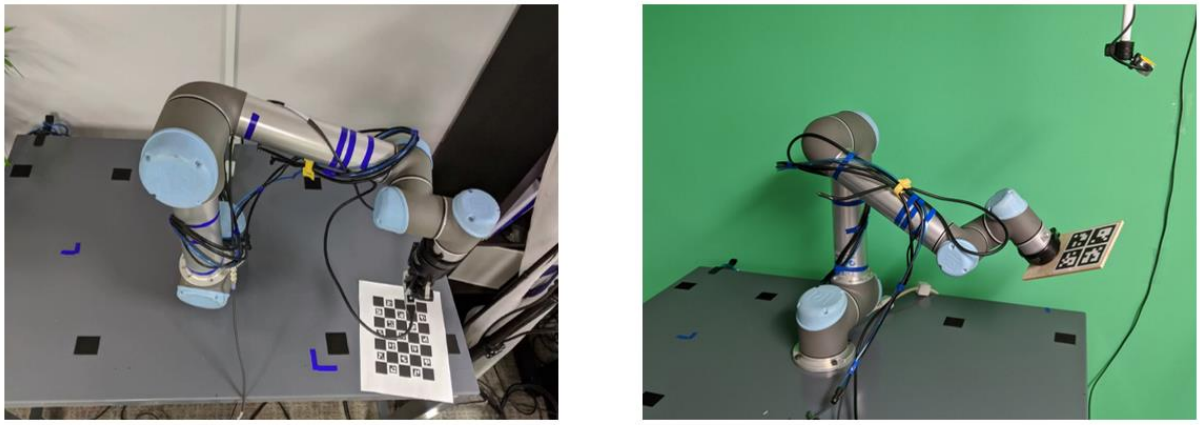
U RViz-u je moguće iz dobivenog oblaka točaka napraviti OctoMap koji onda predstavlja prepreke u prostoru za robotsku ruku. Same dimenzije voksela iz OctoMap-a moguće je podešavati. MoveIt u sebi već ima implementiran OctoMap te će automatski u planiranju budućih putanja izbjegavati voksele u prostoru. Na slici ispod prikazan je spoj robotske ruke sa stereovizijskom kamerom i očitavanja dobivenih podataka iz OctoMap-a u RViz-u.



Slika 3.5 Prikaz dobivenih podataka iz OctoMapa u RViz-u

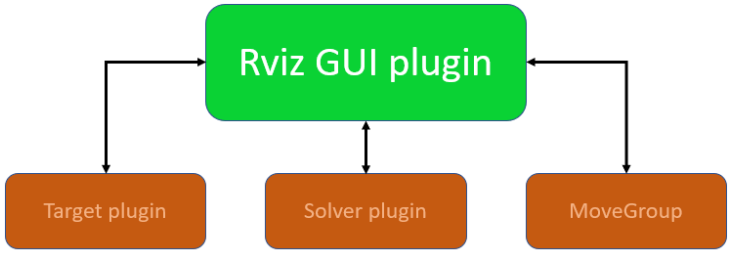
3.3. Kalibracija 3D vizijskog sustava

Prije detekcije i prostorne lokalizacije objekata u prostoru potrebno je napraviti kalibraciju pozicije vizijskog sustava u odnosu na robotsku ruku. Za kalibraciju pozicije kamere u odnosu na robotsku ruku koristi se *Hand Eye* kalibracija. *Hand Eye* kalibracija služi kao veza između koordinatnog sustava kamere i koordinatnog sustava baze robota. Robot u svakom trenutku zna poziciju zgloba i hvataljke u prostoru za razliku od kamere. Postoje dvije vrste *Hand Eye* kalibracije *Eye in Hand* kalibracija i *Hand to Eye* kalibracija. Na slici ispod prikazane su dvije vrste *Hand Eye* kalibracije.



Slika 3.6 a) Eye in Hand kalibracija, b) Hand to Eye kalibracija [23]

Na ekesprementalnom postavu sustava kamera nalazi se na robotskoj ruci zbog toga je potrebno je napraviti *Eye in Hand* kalibraciju robotske ruke. Kalibracija robotske ruke radi se pomoću MoveIt kalibracijskog alata. Plugin ne dolazi automatski instaliran s MoveIt paketom te je potrebno napraviti zasebnu instalaciju. Kalibracijski dodatak nakon instalacije poziva se iz RViz sučelja. Na slici ispod prikazana je arhitektura MoveIt kalibracijskog dodatka.

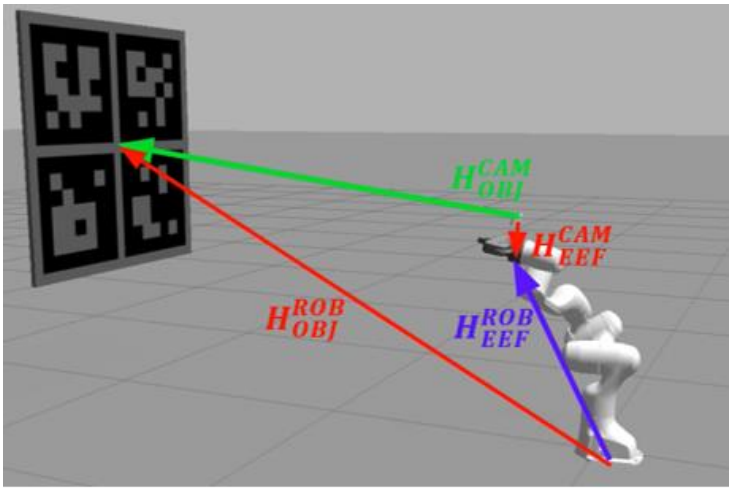


Slika 3.7 Prikaz MoveIt kalibracijske arhitekture [23]

Prednost ovakve arhitekture je njezina jednostavna nadogradnja i prilagodba svakom korisniku. RViz GUI dodatak služi za vizualizaciju sučelja u RViz okruženju te se na njega povezuju ostali programski dodatci. Target plugin modul koristi se za vizualnu detekciju markera odnosno dobivanje transformacije između kamere i markera. Rješavač (eng. *solver*) se koristi za rješavanje jednadžbe $AX = XB$ objašnjene kasnije u poglavlju. *MoveGroup* modul

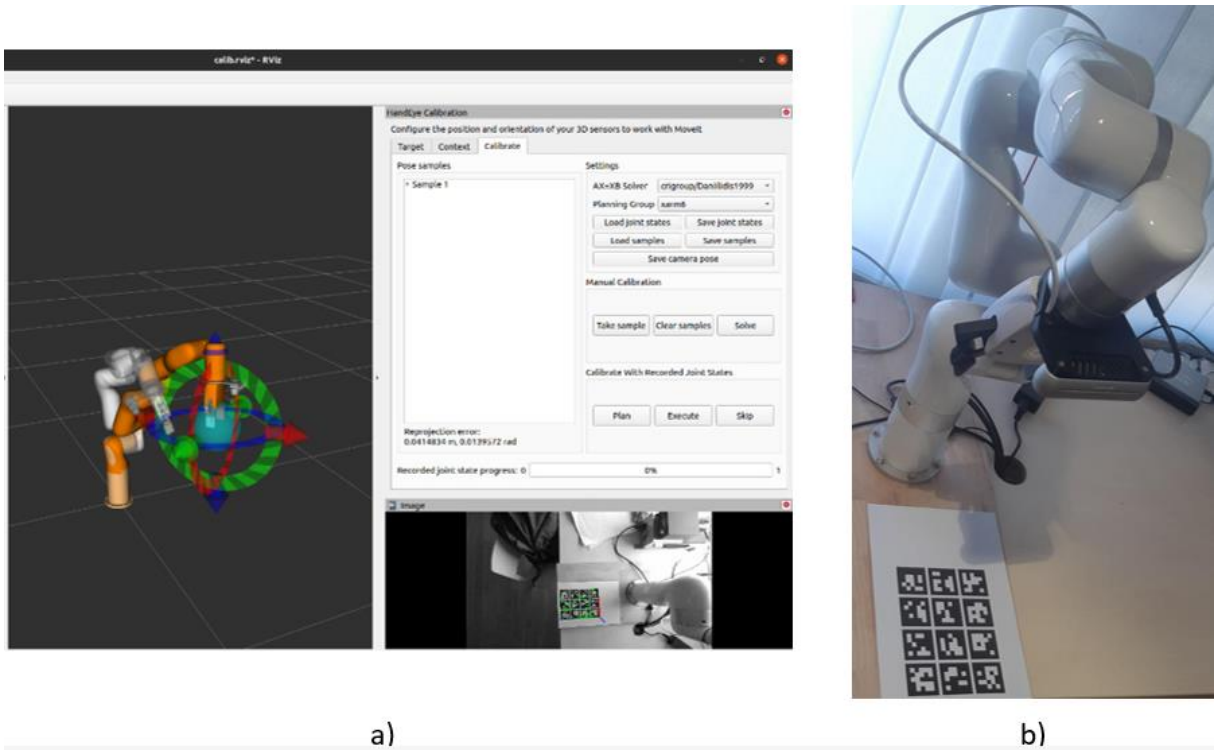
koristi se dobivanje kinematike robotske ruke i omogućavanje provođenja automatske rekalibracije robotske ruke.

Na kraju kalibracijskog postupka dobiven je odnos koordinatnog sustava kamere i koordinatnog sustava prirubnice robotske ruke. Na slici ispod prikazane su poznate homogene transformacije (označene zelenom i plavom bojom) i nepoznate homogene transformacije (označene crvenom bojom) unutar sustava.



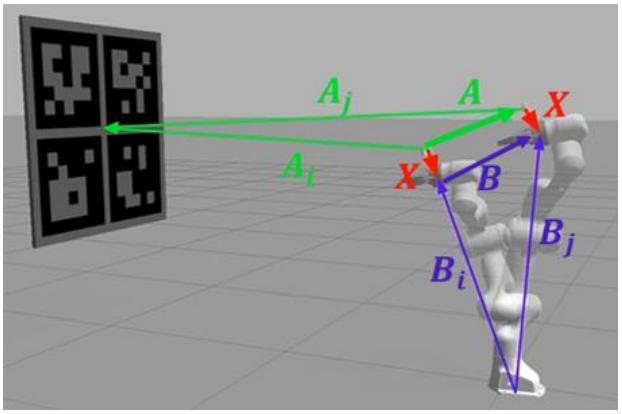
Slika 3.8 Prikaz homogenih transformacija Eye in Hand kalibracije [23]

Poznata je homogena transformacija između baze robota i prihvatnice H_{EEF}^{ROB} zbog poznavanja kinematskog modela robota. Poznavanje homogene transformacije između kamere i markera H_{OBJ}^{CAM} prikazanog na slici moguća je zbog točno poznatih dimenzija markera koji pomoću detektora slike izračunavaju homogenu transformaciju. Dvije nepoznanice koje su označene crvenom bojom su ekvivalentne i izvođenjem bilo koje tri transformacije u pravilnom redoslijedu dati će četvrtu transformaciju. Prilikom kalibracije dobivena transformacija je između kamere i prihvatnice H_{EEF}^{CAM} i prilikom pokretanja robotskog sustava potrebno ju je uvrstiti. Unutar RViz-a korišten je dodatak za kalibraciju robotske ruke te je na slici ispod prikazan RViz i robotska ruka u poziciji lokalizacije markera.



Slika 3.9 a) Kalibracijski alat unutar RViz-a, b) Prikaz robotske ruke prilikom kalibracije

Dobivanje točne transformacije između kamere i markera H_{OBJ}^{CAM} nije jednostavan postupak te je potrebno napraviti više snimanja iz što različitijih kutova kako bi mogli optimizirati homogenu matricu transformacija. Problem koji se rješava pomoću jednadžbe $AX = XB$. Dobivena jednadžba je izvedena iz dvije različite pozicije robota prikazanih na slici ispod.



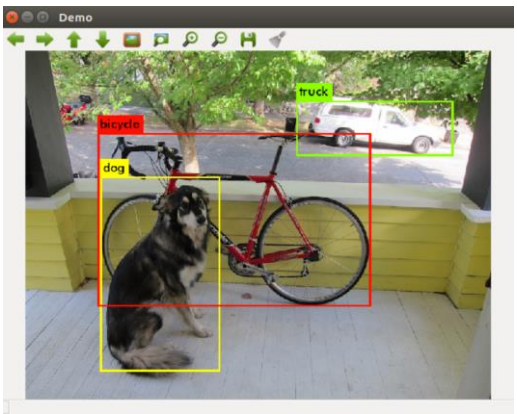
Slika 3.10 Prikaz kalibracijskog postupka Eye in Hand kalibracije [23]

Slovima A_i i A_j označene su transformacije između kamere i markera u dvije različite pozicije robotske ruke, a B_i i B_j označavaju transformacije koordinatnog sustava prirubnice robotske ruke iz dvije različite pozicije. Slovima A i B označeni su pomaci kamere iz jedne u drugu točku, odnosno robota iz jedne u drugu točku prostora. Slovom X označena je nepoznata transformacije između kamere i prirubnice. Postoji puno algoritama za optimalno rješavanje jednadžbe $AX = XB$ te su u MoveIt kalibracijski paket implementirana tri najpouzdanija [24].

3.4. Detekcija i prostorna lokalizacija objekata

Za grubu detekciju i prostornu lokalizaciju objekta u prostoru koristi se konvolucijska neuronska mreža YOLO v5. Kombinacijom neuronske mreže i stereovizijskog sustava moguće je dobiti poziciju željenih objekata u prostoru i grube dimenzije objekata. Zahvaljujući ovim informacijama moguće je u kasnijim obradama oblaka točaka dobiti izolaciju objekta unutar oblaka te samim time omogućiti manipulacije objektom. Dobivena lokalizacija objekta je u koordinatnom sustavu kamere te je za daljnju obradu objekta potrebno njegove koordinate prebaciti u koordinatni sustav baze robotske ruke.

ROS paket koji se koristi za neuronsku mrežu zove se *darknet_ros*. Paket se pretplaćuje (eng. *subscribe*) na 2D kameru vizijskog sustava te obrađuje njenu 2D sliku. Na izlazu daje broj detektiranih objekata, polja gdje se objekti nalaze, pozicije, dimenzije *bounding_boxova* i prikaz slike s označenim svim prepoznatim objektima na slici. Na slici ispod prikazana je jedna takva slika.



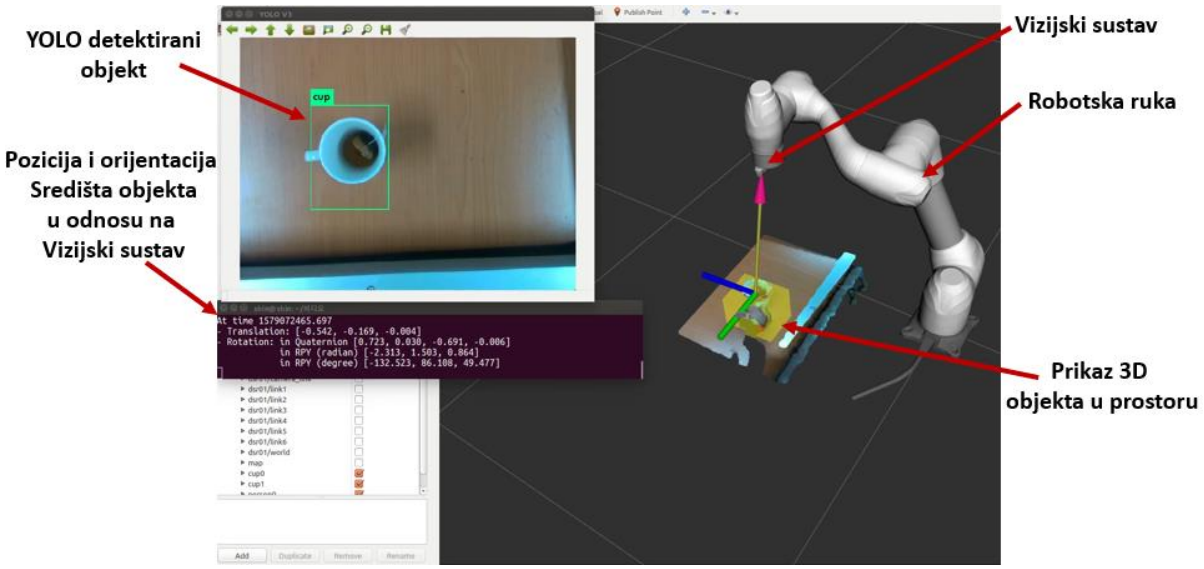
Slika 3.11 Detekcija objekata na slici pomoću YOLO mreže [25]

U paketu je moguće podesiti određene parametre unutar *yolo.yaml* datoteke u konfiguracijskoj datoteci kako bi se rad mreže prilagodio potrebama. Parametri za sortiranje izlaznih podataka mreže su:

- *Darknet_ros/yolo_network_config/cfg/* – je parametar za izbor YOLO mreže koja će se koristiti za detekciju objekata (YOLO v2, YOLO v3, YOLO v4, YOLO v5)
- *yolo_model/threshold/value* – vrijednost iznad koje će se prikazivati detektirani objekti
- *yolo_model/detection_classes/names* – imena klasa korištenih u cfg-u i težinskoj datoteci

Može se primijetiti da je jedan od parametara verzija neuronske mreže koja će se primijeniti u detekciji objekata, što ovaj paket čini raznolikim za implementaciju već gotovih neuronskih mreža u ROS-u iz različitih verzija YOLO mreža. Naposljetku neuronske mreže su implementirane pomoću Darknet [20] razvojne okoline što ih čini optimiziranim za upotrebu na GPU-ovima.

Lokalizacija objekta zajedno s izradom objekta u 3D prostoru omogućava izolaciju objekta unutar oblaka točaka s ciljem kasnije obrade. Vizijski sustav pruža RGBD podatak te je iz detektirane slika u YOLO mreži moguće napraviti 3D *bounding_box*. RGBD podatak je slika koja posjeduje dubinu i zbog definiranih *bounding_box-ova* unutar YOLO mreže moguće je napraviti kvadar te iz definiranog kvadra naći njegovo središte koje je potom orijentir za sve potrebne transformacije koordinatnih sustavu u prostoru. Na slici ispod prikazan je testni postav koji pokazuje sve dobivene podatke iz *gb_visual_detection_3d* ROS paketa. Podatak koji nije dobivan iz ROS paketa je središte objekta za koji je bilo potrebno napisati dodatnu skriptu.



Slika 3.12 Prikaz lokalizacije objekta u prostoru pomoću *gb_visual_detection_3d* ROS paketa[26]

Unutar *gb_visual_detection_3d* ROS paketa moguće je podešavati određene parametre kojima se paket može prilagoditi upotrebi korisnika. Parametri koji se mogu prilagoditi su:

- *interested_classes* – klase koje se žele detektirati. Moraju biti klase koje su prije toga definirane u darknet_ros paketu
- *minimum_detection_threshold* – maksimalna dopuštena udaljenost između bilo kojeg piksela na slici od središnjeg piksela slike
- *minimum_probability* – minimalna dopuštena vjerojatnost objekta iz kojeg će se napraviti 3D objekt

Za pisanje skripte koja izračunava središte objekta potrebno je „slušati“ poruke koje dolaze s */darknet_ros_3d/bounding_boxes_3d* teme. Unutar ove poruke nalaze se minimalne i maksimalne udaljenosti po y, x i z osi unutar definiranog *bounding_box*-a YOLO mreže i RBGD slike. Nakon što se očita dobivena poruka potrebno je zbrojiti minimalnu i maksimalnu udaljenost po osi te ju podijeliti s dva. Ovaj postupak je potrebno ponoviti za sve tri osi te naknadno objaviti koordinatni sustav objekta.

Zbog moguće pojave netočne dubine objekta zbog očitavanja objekta iz gornje perspektive produžuje se dubina *bounding_boxes* kako bi se osiguralo da je cijela dubina objekta uhvaćena.

Na kraju dobivene informacije iz *gb_visual_detection_3d* ROS paketa koriste se u Open3D programskoj knjižnici kako bi se napravila gruba segmentacija određenih dijelova oblaka

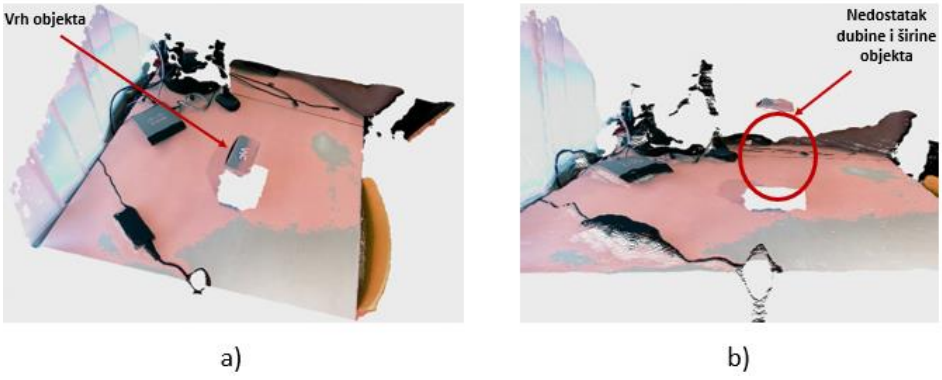
točaka. Na izdvojenom dijelu oblaka naknadno se radi dodatna obrada točaka kako bi se uklonile sve nepravilnosti. Na kraju se izdvojeni oblak preklapa s primitivom kako bi se mogla izvršavati manipulacija s detektiranim objektima pomoću YOLO mreže.

3.5. Segmentacija objekata u prostoru i njihovo preklapanje s primitivima

U ovom radu zbog isprepletenosti i potrebe za korištenjem više različitih programskih knjižnica napravljena je vlastita programska knjižnica za manipulaciju i obradu podataka oblaka točaka u kojoj su implementirane sve potrebne programske knjižnice i njihove metode. U ovoj knjižnici napisane su metode koje objedinjuju metode iz drugih programskih knjižnica i omogućavaju njihovu sinergiju kako bi bilo jednostavnije korištenje istih u implementaciji s ROS-om. Napravljena programska knjižnica zove se *ros_open3d_manipulation*.

Poslije detekcije i grube prostorne lokalizacije objekata u prostoru potrebno je napraviti detaljno snimanje okoline oko nađenih objekata, to jest snimanje okoline iz više različitih kutova. Preduvjet za snimanje okoline iz više različitih kutova je kalibracija kamere, odnosno *Eye in Hand* kalibracija.

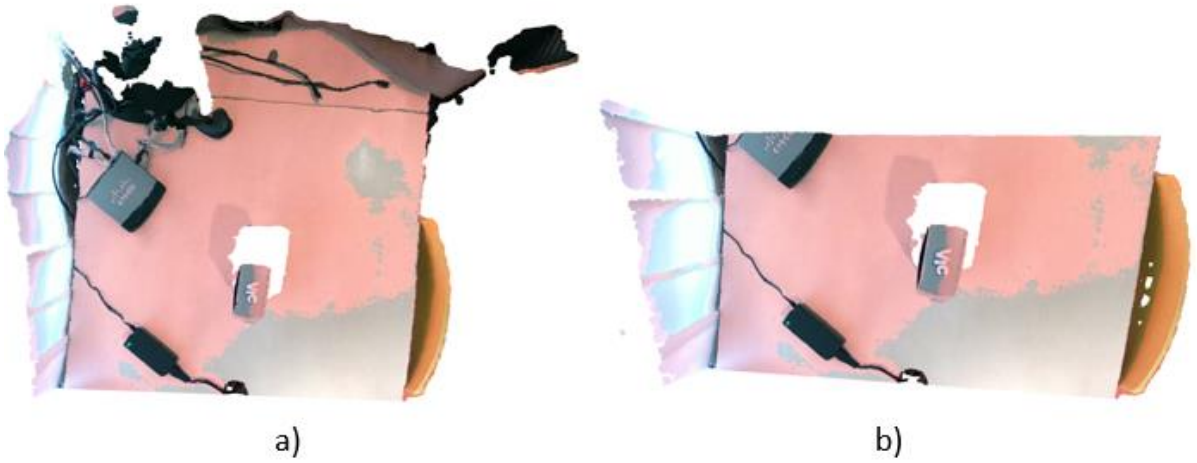
Potreba za snimanje prostora iz više različitih kutova dolazi zbog moguće nekoherentnosti jednog oblaka točaka. Prilikom snimanja iz jednog kuta može doći do nepotpune informacije o objektu što može rezultirati neuspjehom u izuzimanju objekta. Na slici ispod prikazana je jedan takav oblak točaka.



Slika 3.13 a) Prikaz oblaka točaka iz gornje perspektive, b) Prikaz oblaka točaka iz prednje perspektive

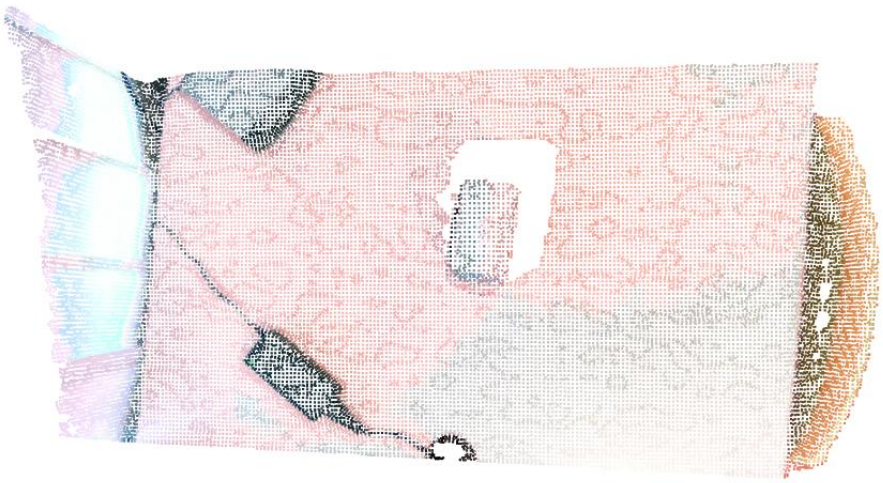
Na slikama iznad može se primijetiti kako je oblak točaka snimljen iz gornje perspektive te nije u mogućnosti sa sigurnošću procijeniti širinu, duljinu i dubinu objekta na sredini stola. Rješenje ovog problema je napravljeno uz pomoć snimanja okoline iz više različitih kutova što se zove šivanje oblaka točaka (*eng. point cloud stitching*). U ovom radu snimali su se oblaci točaka iz tri različita kuta.

Prva pozicija snimanja oblaka točaka je direktno iznad sredine površine, a druge dvije su pomaci robotske ruke za 60 stupnjeva u lijevu i desnu stranu od prve pozicije. Akvizicija oblaka točaka u svakoj od tri pozicije izvršava se pomoću metode *get_pc2_data*. Direktno poslije akvizicije svakog od oblaka točaka potrebno je napraviti transformaciju oblaka iz koordinatnog sustava kamere u kojem je snimljen u koordinatni sustav baze robota, što se radi pomoću metode *ros_transformation*. Nakon što je izvršeno snimanje sva tri oblaka točaka potrebno je napraviti izmjenu vrste podataka oblaka, odnosno način njihovog zapisivanja. Snimljeni oblaci su u oblika *PointCloud2* što je standardni ROS-ov zapis oblaka točaka, a za manipulaciju oblakom unutar programske knjižnice *Open3D* potrebna je prebaciti u drugačiji zapis te je u ovom radu prebacivan u *.ply* zapis oblaka točaka pomoću metode *convert_pc2_to_o3d*. Nakon što je izvršena konverzija zapisa oblaka započinje se s njegovom manipulacijom. Prvo se upotrebljava metoda za brisanje svih točaka izvan radnog prostora robotske ruke. Za ovu manipulacije korišten je *pass through* filtar pozvan metodom *o3d_pass_through_filter*. Na slici ispod prikazan je oblak točaka prije i poslije primjene filtra.



Slika 3.14 a) Oblak točaka prije Pass Through filtra, b) oblak točaka poslije filtra

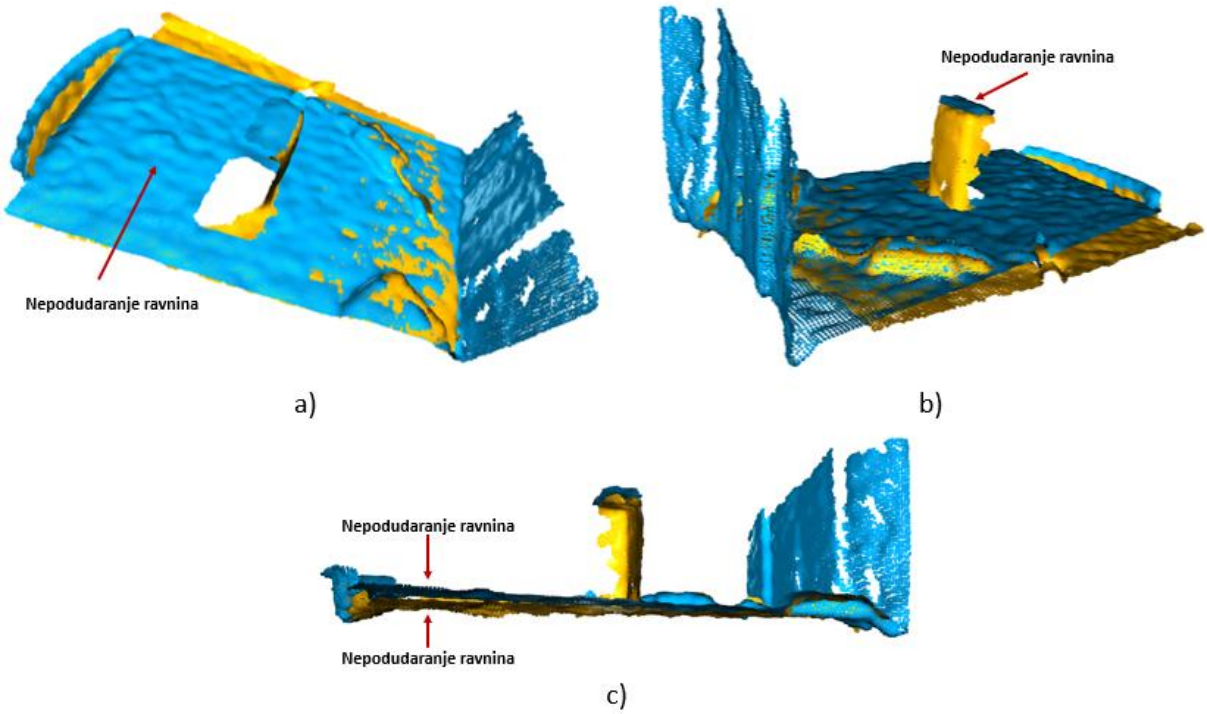
Sljedeći korak je upotreba voksel filtra pomoću kojeg se značajno smanjuje broj točaka unutar oblaka te se ubrzava manipulacija i obrada oblaka točaka. Metoda za filtriranje oblaka točaka je *o3d_voxel_down_sample*. Unutar ove metoda podešeni su svi parametri za korištenje Open3D voksel filtra.



Slika 3.15 Oblak točaka poslije Voxel filtera

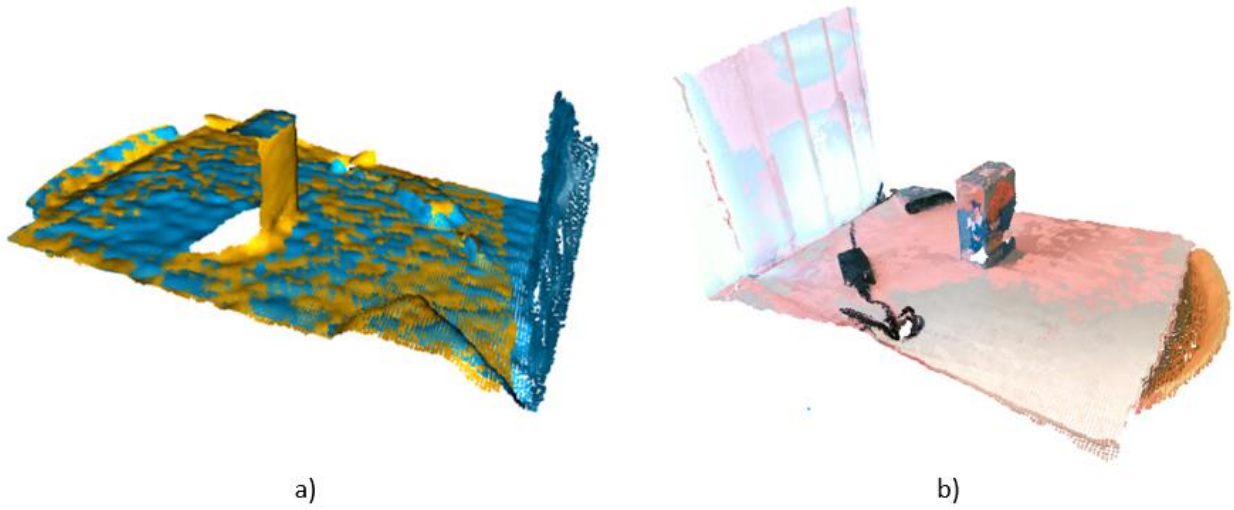
Nakon što je napravljeno pretvaranje oblaka točaka u voksele potrebno je izbrisati preostale točke koje stvaraju šumove. Točke koje stvaraju šumove unutar oblaka se smatraju sve točke izvan prosječne udaljenosti između točaka unutar oblaka. Za čišćenje šuma koristi se metoda *o3d_cleaning_outlier*. Posljednji korak prije spajanja više oblaka je estimacija normala unutar oblaka pomoću metode *o3d_normal_estimate* koje su potrebne za korištenje registracijskih algoritama.

Odabrani registracijski algoritam za spajanje više različitih oblaka točaka je ICP (*eng. Iterative Closest Point*) registracijski algoritam. Unutar registracije moguće je koristi dvije različite metode za registraciju oblaka jednog na drugu. Prva metoda je *TransformationEstimationPointToPoint* i druga metoda je *TransformationEstimationPointToPlane*. Testiranjem na različitim uzorcima metoda *TransformationEstimationPointToPlane* pokazala se kao pouzdanija. Na slikama ispod prikazana su dva oblaka točaka prije ICP registracije i transformacije jednog na drugi (jedan oblak je plave boje, a drugi žute).



Slika 3.16 a)-c) Prikaz nepodudaranja jednog oblaka točaka na drugi

Metoda korištena za ICP registraciju tri oblaka točaka jednog na drugi je *o3d_point_cloud_registration*. Prvi ulazni oblak je glavni te će se na njega raditi transformacije preostala dva oblaka. Odabrani oblak za glavni je ujedno i prvi snimljen prilikom kojeg nije bilo nikakvih zakreta te su najveće šanse da je taj oblak i najtočniji u odnosu na robotsku ruku.Na slikama ispod prikazani su oblaci nakon ICP registracije. Na lijevoj slici prikazan je spoj dva oblaka te na desnoj spoj sva tri oblaka točaka.



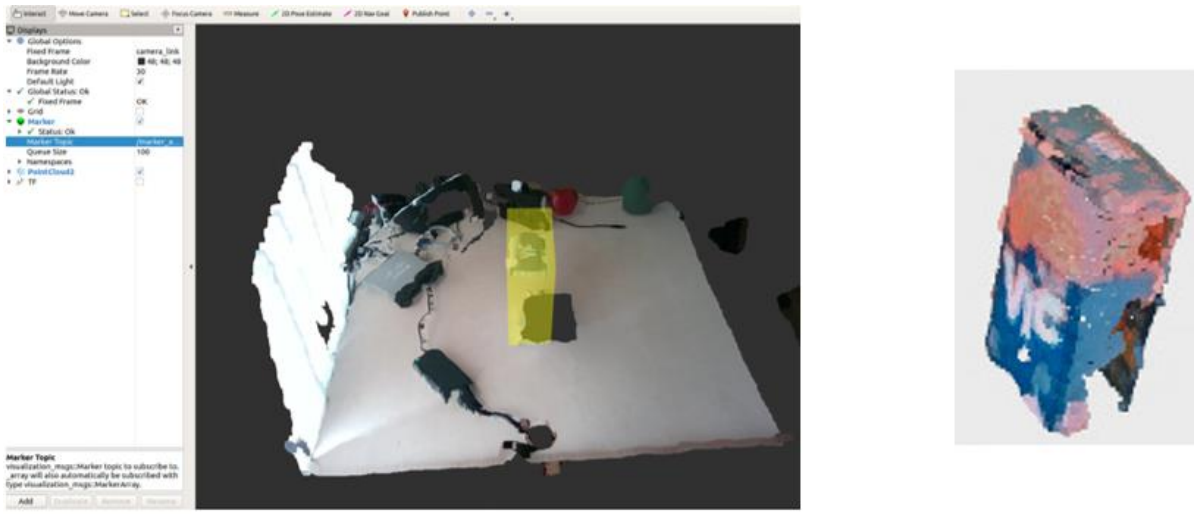
Slika 3.17 a) Prikaz spajanja dva oblaka točaka poslije ICP registracije, b) Prikaz spajanja tri oblaka točaka

Poslije dobivanja spojenog oblaka točaka potrebno je napraviti segmentaciju ravnine. Segmentacija ravnine provodi se iz više razloga, a jedan od glavnih razloga je što se najveći broj točaka u prostoru uvijek nalazi na podu odnosno snimljenoj površini. Pod u ovom slučaju nema nikakvu ulogu nakon registracije oblaka na oblak, a zauzima puno procesorske moći prilikom obrade podataka te se iz tih razloga izdvaja iz oblaka kao zasebni oblak i koristi se kasnije u ROS-u za zadavanje radne plohe. Algoritam koji se koristi za segmentaciju ravnine je *RANSAC*. *RANSAC* algoritam koristi se unutar metode *o3d_plane_segmentation*. Na slikama ispod prikazani su izdvojena ravnina i preostali objekti na istoj.

Slika 3.18 a) Segmentirana ravnina pomoću RANSAC algoritma, b) Prestali objekti u oblaku točaka

Potrebno je ponovno čišćenje šumova nastalih prilikom micanja najveće ravnine iz oblaka točaka. Ovaj korak je napravljen pomoću *DBSCAN* algoritma. *DBSCAN* služi za grubu klasterizaciju oblaka te sve točke koje ne zadovoljavaju zadane uvjete algoritma smatraju se šumom i brišu se. Algoritmu se zadaje maksimalna dozvoljena udaljenost između točaka i minimalan broj točaka unutar novonastalog klastera. Ako su uvjeti maksimalne dozvoljene udaljenosti između točaka i dovoljan broj točaka unutar klastera zadovoljeni, točke se neće smatrati šumom i shodno tome neće biti izbrisane. Ovaj algoritam implementiran je u programskoj knjižnici pod nazivom *o3d_clustering*.

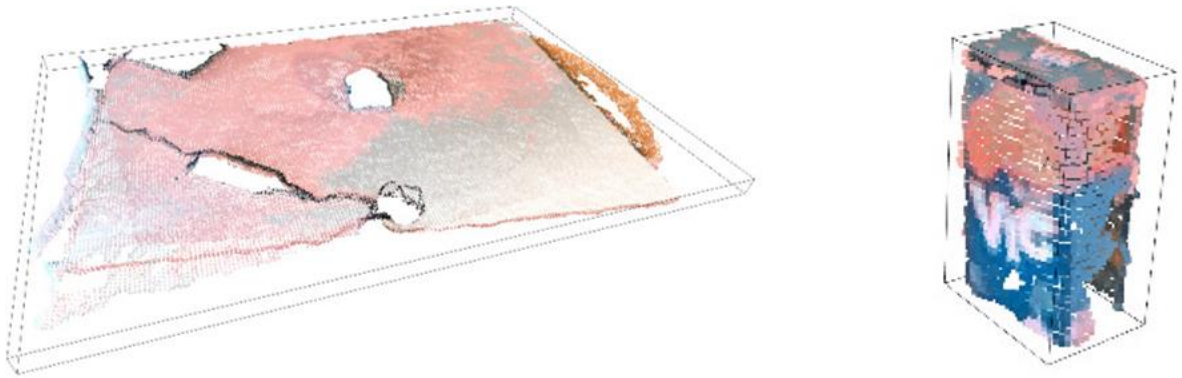
Pomoću YOLO konvolucijske neuronske mreže objašnjene u prethodnom poglavlju dobivena je gruba lokalizacija i dimenzija predmeta u prostoru pomoću koje se radi posljednja manipulacija oblaka odnosno njegovo uređivanje. Pomoću dobivenih procijenjenih dimenzija objekta u prostoru moguće je izolirati objekt od ostatka preostale scene, odnosno izbrisati sve točke koje nisu unutar grube lokalizacije. Ovaj postupak se provodi pomoću metode *o3d_yolo_isolation*. Na slici ispod prikazan je grubo lokaliziran objekt u RViz-u i taj isti objekt izoliran od ostatka oblaka točaka pomoću grube lokalizacije pomoću YOLO mreže.



a) b)

Slika 3.19 a) Gruba izolacija objekta u prostoru pomoću YOLO mreže prikazana u RViz-u, b) Izolirani objekt u prostoru pomoću YOLO mreže od ostatka spojenog oblaka točaka

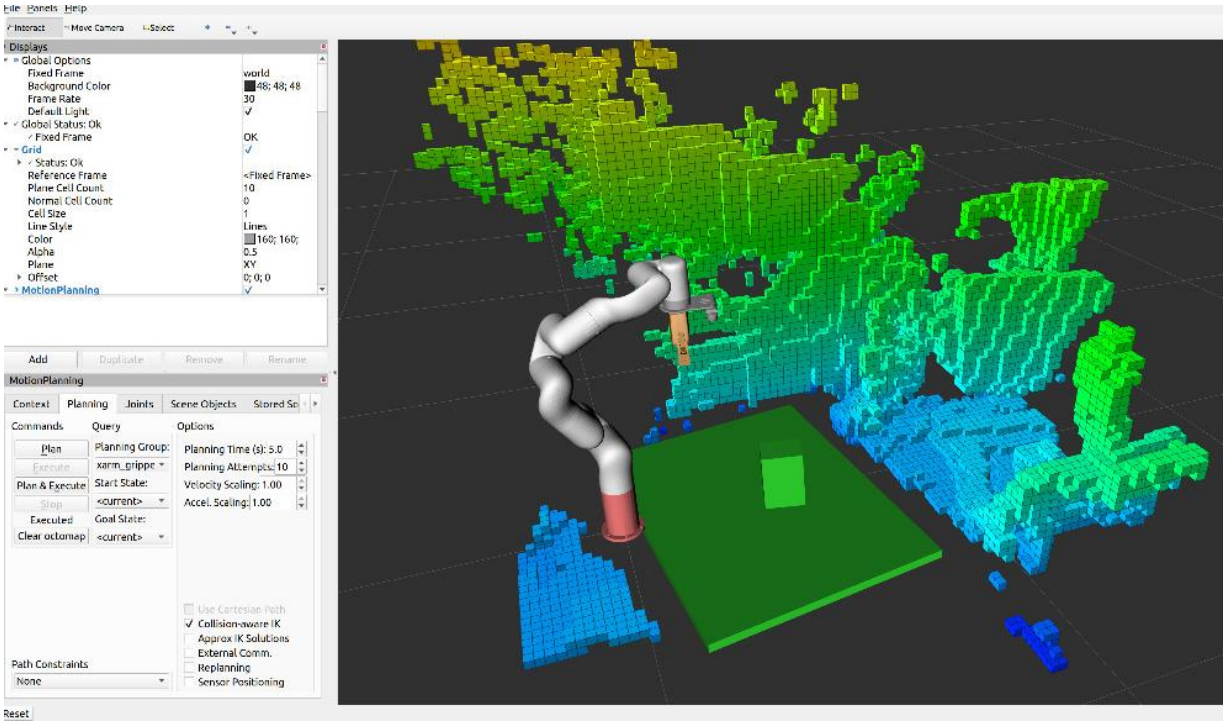
Dobivene oblake točaka podloge i objekta detektiranog YOLO mrežom potrebno je opisati pomoću primitiva, točnije dobiveni oblak točaka opisati geometrijskim oblikom dimenzija maksimalno ispupčenih točaka unutar oblaka po x, y i z osi. Prilikom pretvaranja oblaka točaka objekta u geometrijski primitiv potrebno je uzeti u obzir njegovo težište odnosno glavne osi objekta. Glavne osi objekta predstavljaju u odnosu na bazu robota orijentaciju objekta u prostoru. Glavne osi objekta dobivene su pomoću algoritma analize glavnih komponenti (*eng. Principal component analysis - PCA*). Za dobivanje geometrijskog primitiva orijentiranog na temelju glavnih osi oblaka točaka korištena je metoda *o3d_primitiv_fitting*. Na slici ispod prikazani su dobiveni minimalni kvadratni primitivi orijentirani na temelju glavnih osi u prostoru.



a) b)

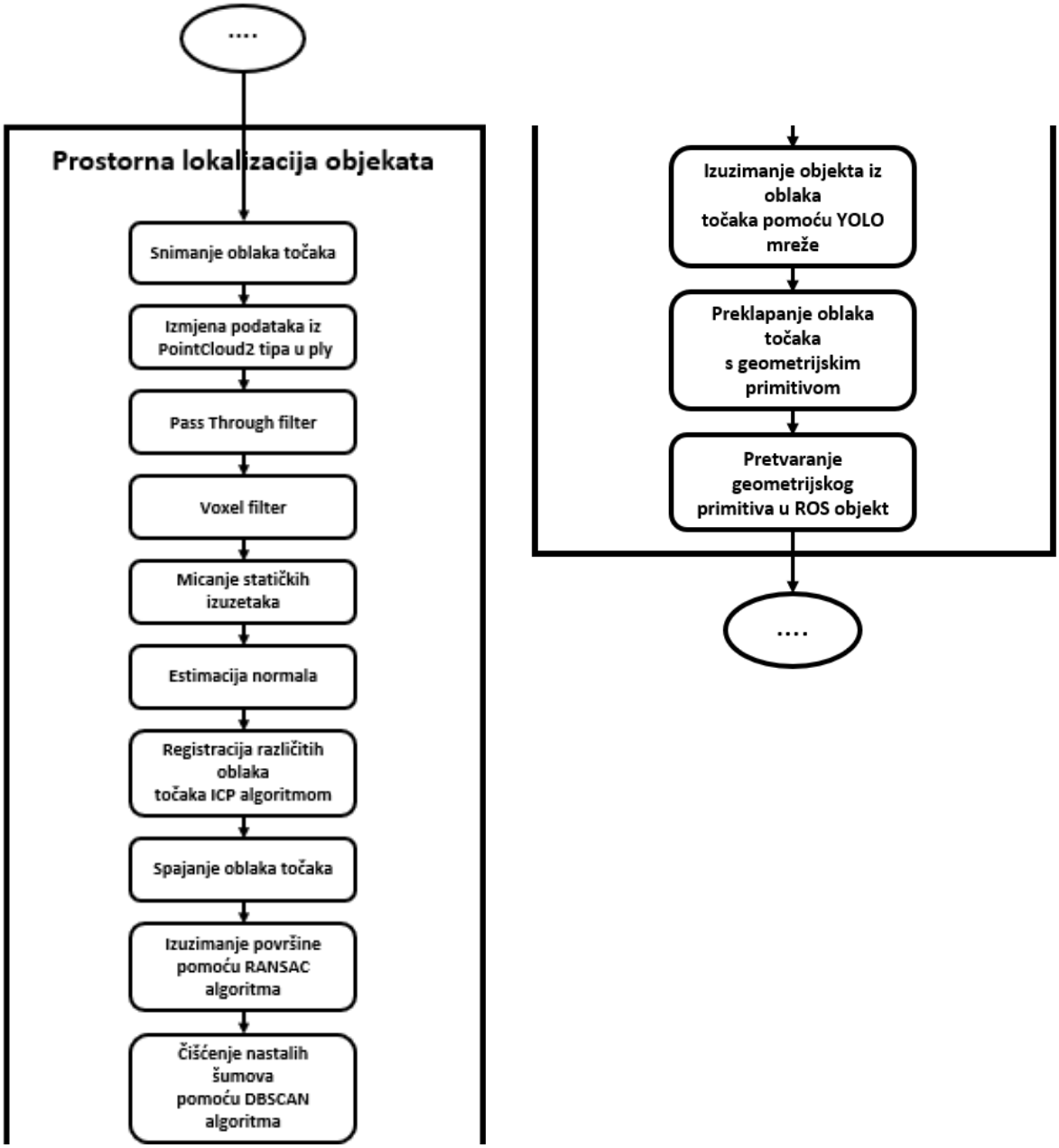
Slika 3.20 a) Preklapanje geometrijskog primitiva za podlogu, b) Preklapanje geometrijskog primitiva za izolirani objekt

Dobivene geometrijske primitive potrebno je transformirati iz Open3d sučelja u ROS sučelje i od definiranih dimenzije i orijentacije primitiva napraviti objekt za manipulaciju unutar ROS okolini. Pretvaranje geometrijskog primitva unutar open3D okoline u ROS objekt za manipulacije izvršava se pomoću metode *ros_primitiv_fitting*. Na slici ispod prikazani su objekt podloge i očitanog objekta pomoću YOLO mreže unutar ROS okoline.



Slika 3.21 Prikaz dodanih geometrijskih primitva u ROS okolinu kao objekti za manipulaciju

Cjelokupni proces od snimanja, manipulacije oblaka točaka i stvaranje objekata za manipulaciju unutar ROS-a prikazan je na slici dijagrama toka ispod.

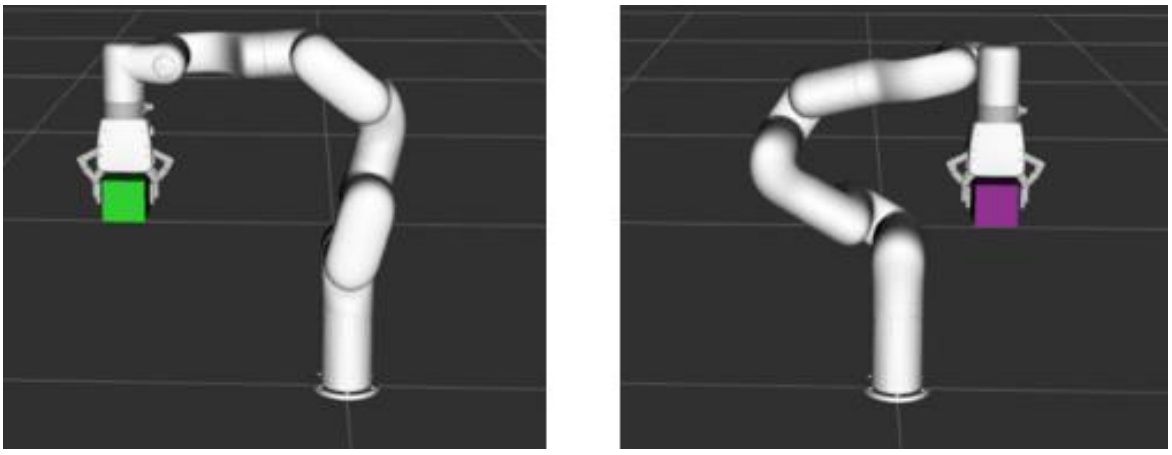


Slika 3.22 Dijagram toka prostorne lokalizacije objekta

3.6. Algoritmi za izuzimanje i manipulaciju objektima

Rukovanje objektima u MoveIt-u moguće je raditi pomoću različitih algoritama. Postoji puno paketa koji se bave ovom problematikom i optimizacijom rukovanja objektima u prostoru. U ovom radu obrađena su dva najpoznatija načina rukovanja objekatima.

Prvi način prihvata objekta je poznavanjem u potpunosti pozicije objekta i točnim definiranjem pozicije hvatanja objekta, aktiviranje hvataljke robotske ruke, izuzimanja objekta, dodavanjem objekta u kinematiku robotske ruke, manipulaciju objektom te na kraju odlaganjem objekta. Na slici ispod može se vidjet proces premještanja zelene kocke s lijeve strane na desnu stranu. Kocka koja se nalazi u prostoru nije dio robotske ruke, zelene je boje te se smatra preprekom prilikom hvatanja objekta. Robotska ruka dolazi iznad objekta te se poziva metoda za hvatanje (izuzimanje) objekta čime kocka mijenja boju iz zelene u ljubičastu. Ljubičasta kocka postaje dio robotske ruke te prilikom kretanja robotske ruke uzima se u obzir i dimenzija kocke koja se zajedno s rukom pomiče u prostoru. Na posljetku kocka se može odložiti te će promijeniti boju iz ljubičaste u zelenu i time postati opet dio prostora i prilikom planiranja putanja smatrat će se preprekom u prostoru.



a) b)

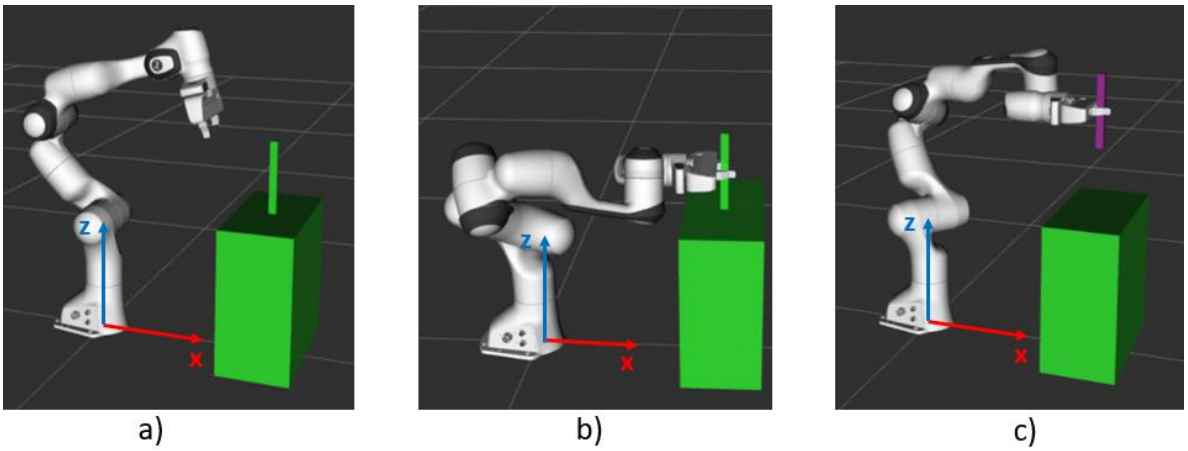
Slika 3.23 a) Dolazak u predefiniranu točku prihvata objekta, b) Pomicanje objekta na točku odlaganja i puštanje objekta

Drugi način rukovanja objektima je pomoću MoveIt metode *pick()*. Za *pick* metodu potrebno je definirati koji objekt u prostoru treba izuzeti i njegovu *grasp* poruku. Prednost korištenja *pick* metode je mogućnost definiranja *grasp* poruke na temelju koje MoveIt samostalno planira ostatak akcije odnosno trajektoriju prilaska predmetu i odmaka od pozicije izuzimanja nakon što se predmet nalazi u hvataljci. Nakon što je predmet u hvataljci on mijenja boju iz zelene u ljubičastu i postaje dio robotske ruke. Svakom objektu u prostoru koji se nalazi

u RViz-u definirano je jedinstveno ime, pozicija, orijentacija, oblik i dimenzija. Za definiciju objekata u prostoru koristi se MoveIt-ov plugin *PlanningScene*. Kako bi se planiranje rukovanja objekta izvelo potrebno je definirati *grasp* poruku. *grasp* poruka se definira pomoću 10 parametara [27]. Parametre koje je potrebno definirati u poruci su:

- *string id* – *Grasp* posjeduje ime jer jedna metoda pick može imati definirano više grasp-ova
- *pre_grasp_posture* – definiranje pozicije hvataljke prije hvatanja
- *grasp_posture* – definiranje pozicije hvataljke u točki prihvata
- *grasp_pose* – pozicija i orijentacija hvataljke u točki prihvata objekta
- *grasp_quality* – definiranje kvalitete prihvata. Moguće je definirati kvalitetu prihvata pomoću više različitih parametara
- *pre_grasp_approach* – u ovom dijelu poruke definira se vektor prilaza objektu, te minimalna i maksimalna dopuštena udaljenost prilaska objektu
- *post_grasp_retreat* – u ovom dijelu poruke definira se vektor izlaza od objekta zajedno s objektom, te minimalna i maksimalna dopuštena udaljenost odmaka od pozicije hvatanja
- *post_grasp_retreat* – mogućnost definiranja vektora izlaska odnosno kretnju koja će se izvesti nakon što se objekt odloži na za to predviđenu poziciju
- *max_contact_force* – maksimalna dopuštena sila kontakta u slučaju da hvataljka posjeduje mjerenje sile u sebi
- *allowed_touch_objects* – moguće je definirati listu objekata koji se neće gledati kao nepomične prepreke nego ih se može dotaknuti, gurnuti ili pomaknuti prilikom hvatanja objekta

Na slikama ispod prikazan je proces hvatanja i pomicanja objekta definiranog pomoću *graspa* i pozivom metode *pick()*.



Slika 3.24 a) Početna pozicija robotske ruke prije pozivanja metode *pick*, b) Pozivanje metode *pick* te gdje je definiran vektora prilaza objektu robotske ruke po osi x, c) Odmicanje od pozicije prihvata objekta po definiranom vektoru odmaka po osi z zajedno s objektom

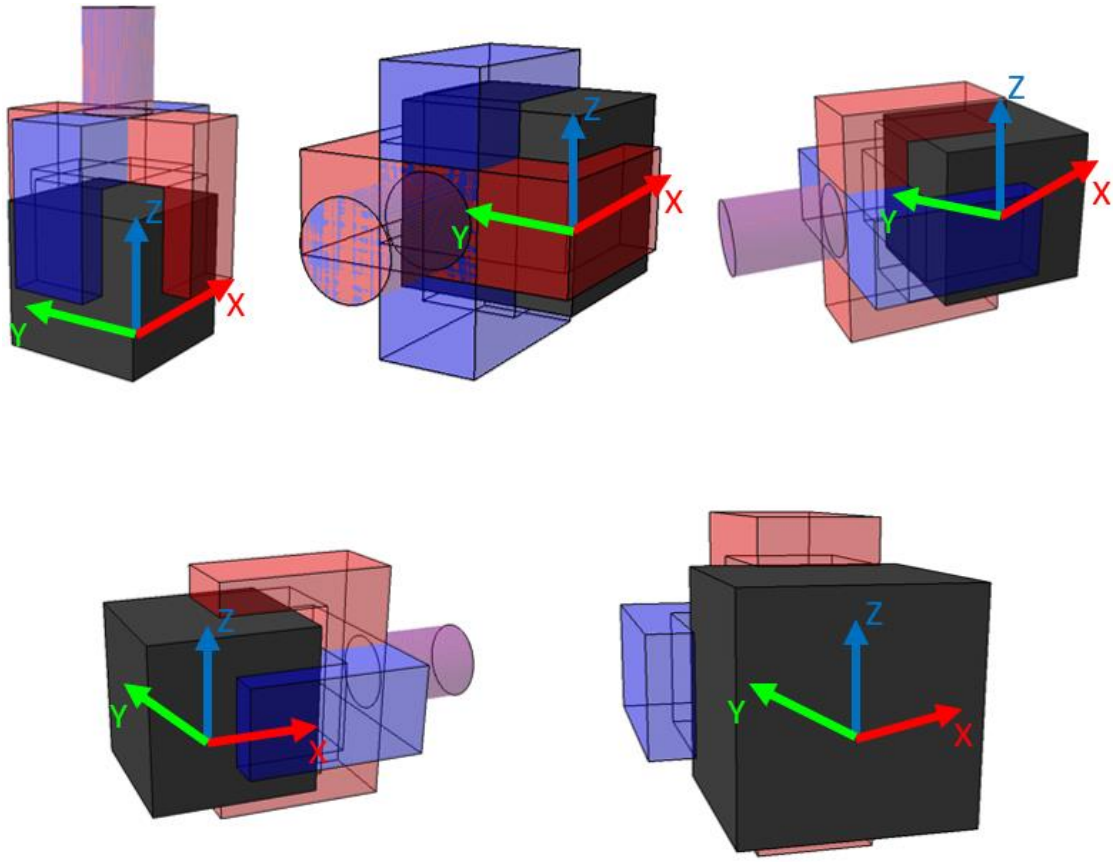
Metodi *pick* moguće je definirati polje *grasp* poruka te će metoda samostalno interakcijom prolaziti od nultog *grasp-a* prema sljedećem u listi, ako nije u mogućnosti izvesti trajektoriju unutar zadanih parametara *grasp-a*. Zbog toga što *pick* metoda ide od nultog člana liste prema sljedećim članovima moguće je definirati važnost prilaska robotske ruke iz određenih pravaca tako što će bitnije vektori prilaska biti na početku liste.

Zbog potrebe za automatizacijom robotske ruke i definiranjem različitih funkcija nastala je potreba za izradom programske knjižnice koja će automatski generirati listu *grasp* poruka za objekte u prostoru koje je moguće uhvatiti s robotskom rukom, definiranjem različitih pozicija robota za različite radnje kao što su slikanje podloge iz različitih kutova, početne pozicije robota, pozicije odlaganja otpada i općenitom manipulacijom objektima u prostoru. Napisana klasa u ovom radu za manipulacijom objekata zove se *ObjectManipulation*. Prednost pisanja vlastite programske knjižnice za rukovanje objektima je mogućnost prilagođavanja knjižnice vlastitim potrebama. Na početku knjižnice velikim tiskanim slovima definirane su globalne varijable *xArm6* robotske ruke te je za korištenje drugih robotskih ruku potrebno izmijeniti globalne varijable kako bi knjižnica radila za njih.

Unutar programske knjižnice napisane su različite metode za rukovanje objektima u prostoru i upravljanje robotskom rukom. Prilikom inicijalizacije klase, klasa započinje s inicijalizacijom svih potrebnih objekata i metoda za komunikaciju s ROS-om. Prva

inicijalizacija je *moveit_commander.roscpp_initialize(sys.argv)* što pokreće *moveit_commander* plugin. Zatim se definiraju objekti za manipulaciju robotske ruke pomoću *moveit_commander.MoveGroupCommander(GROUP_NAME_ARM)* i *moveit_commander.MoveGroupCommander(GROUP_NAME_HAND)*. Pomoću ova dva objekta MoveIt Commander plugin upravlja robotskom rukom i njenom hvataljkom. Poslijednji objekt koji se inicijalizira prilikom pokretanja klase je *moveit_commander.PlanningSceneInterface()*. Pomoću ovog objekta izvršava se sva manipulacija objekata unutar ROS-a.

Za pokretanje rukovanja (izuzimanja) objektom pomoću robotske ruke poziva se metoda *objectPickAndPlace*. Pozivanjem ove metode započinje traženja objekata u prostoru koje je robotska ruka u mogućnosti uhvatiti pomoću metode *graspableObjectDetection*. Ulazni parametri *graspableObjectDetection* su radno polje robotske ruke i maksimalna dopuštena dimenzija objekta. Metoda kao izlaz daje listu imena objekata koje je moguće uhvatiti u prostoru. Za listu definiranih objekata koje je moguće uhvatiti u prostoru potrebno je napraviti željenu listu prilaznih putanja objektu, odnosno napraviti algoritam ili metodu koja će svojim pozivanjem generirati listu *grasp* poruka za hvatanje objekata u prostoru. U ovom radu definirano je 10 putanja prilaska robotske ruke prema objektu. Na slici ispod prikazane su sve moguće pozicije hvatanja objekta pomoću hvataljke i njene prilazne putanje. Crnom bojom je označen objekt, a plavom i crvenom bojom dvije moguće pozicije hvataljke iz definiranog vektora prilaska.

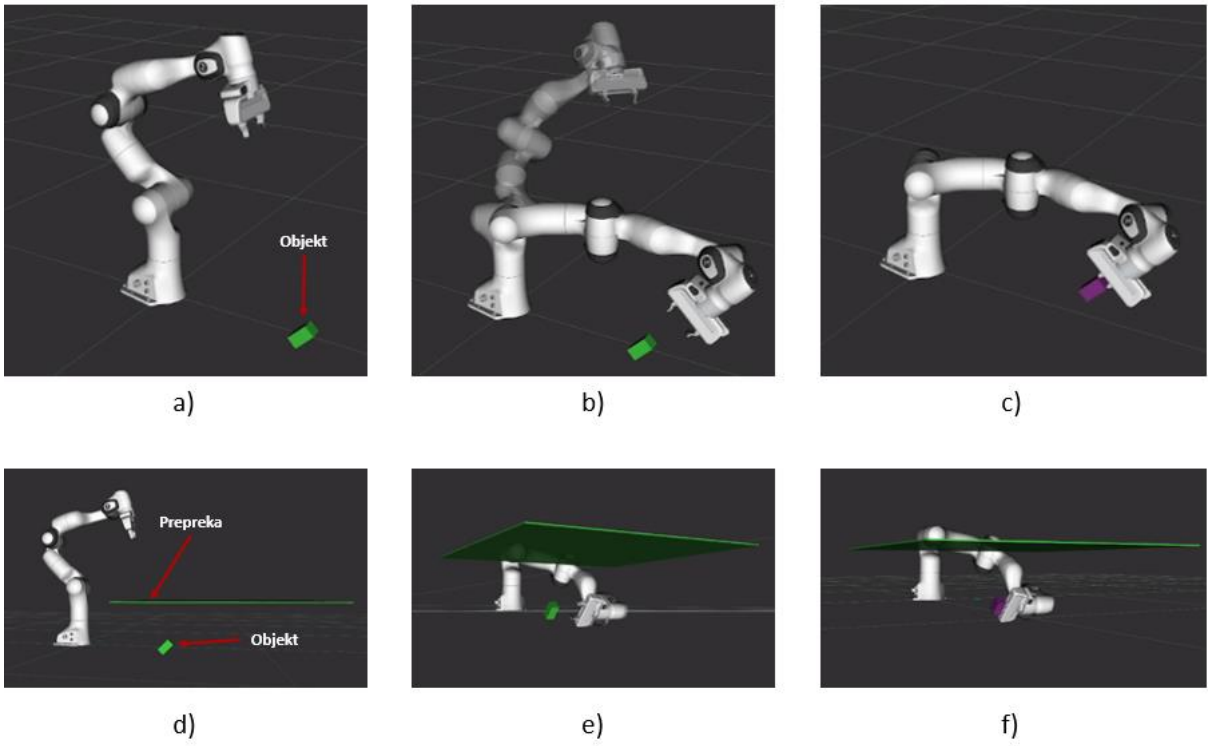


Slika 3.25 Prikaz predefiniranih putanja prilaska objektu zajedno s pozicijama hvataljki

Prilikom definiranje liste *grasp* poruka, odnosno algoritam za generiranje prilaznih putanja poziva se pomoću metode *make_grasps*. Algoritam prilikom prolaska kroz sve definirane prilaze pazi na dimenziju objekt. U slučaju da je dimenzija objekta veća od maksimalne moguće dimenzije raspona hvataljke za prihvat objekta putanja prilaska iz te pozicije neće biti generirana. Ovim se dodatno daje na oprezu kako ne bi došlo do kolizije s objektom prilikom hvatanja.

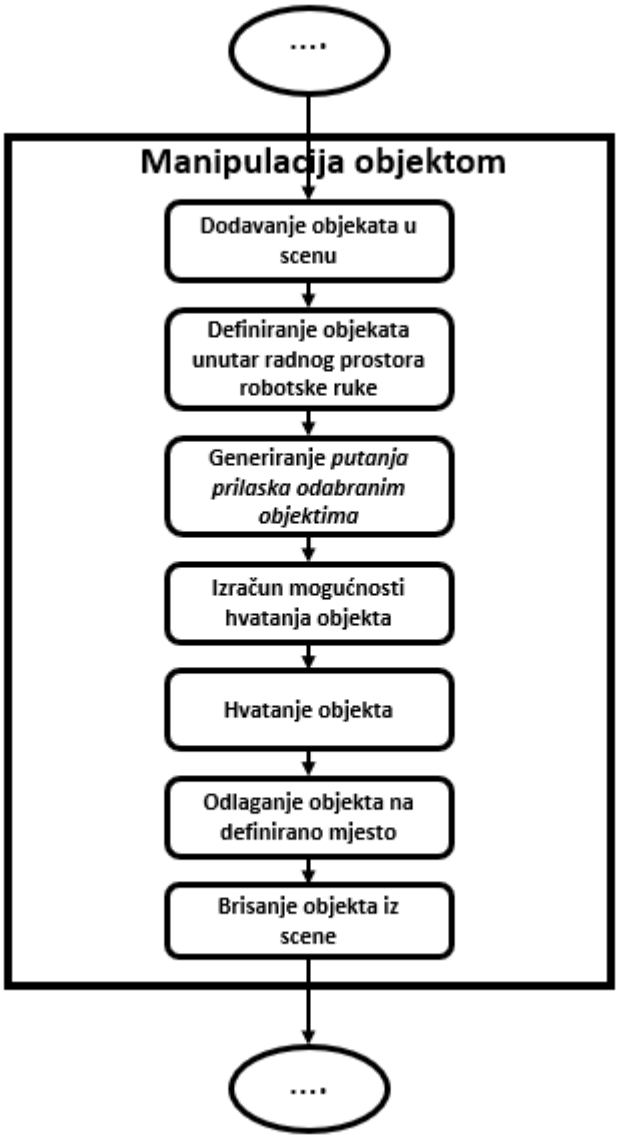
Definirani su svi objekti u prostoru koji se mogu uhvatiti i za njih su definirane sve potrebne putanje prilaska prilikom hvatanja. Poziva se metoda *pick* koja prolazi kroz zadane objekt s njihovim prihvatima te hvata objekt po objekt i odlaže ih na za to predviđeno mjesto. Algoritam je napravljen da preferira hvatanje objekta iz vertikalne pozicije ako je to moguće. U slučaju da hvatanje iz vertikalne pozicije nije moguće pokušava uhvatiti objekt s prednje strane te zatim s bočnih strana i na posljetku sa stražnje strane. Na slikama ispod prikazan je

rad robotske ruke te hvatanje objekta bez preprekom između ruke i objekta i s preprekom između ruke i objekta.



Slika 3.26 a)-c) hvatanje objekta iz vertikalnog prilaska bez prepreke, d)-f) hvatanje objekta s preprekom u vertikalnom prilasku i nemogućnosti prilaska s prednje strane objekta

Na slici ispod prikazan je dijagram toka za manipulaciju objektom u prostoru.



Slika 3.27 Dijagram toka za manipulaciju objektima u prostoru

4. EVALUACIJA EKSPERIMENTALNOG POSTAVA SUSTAVA

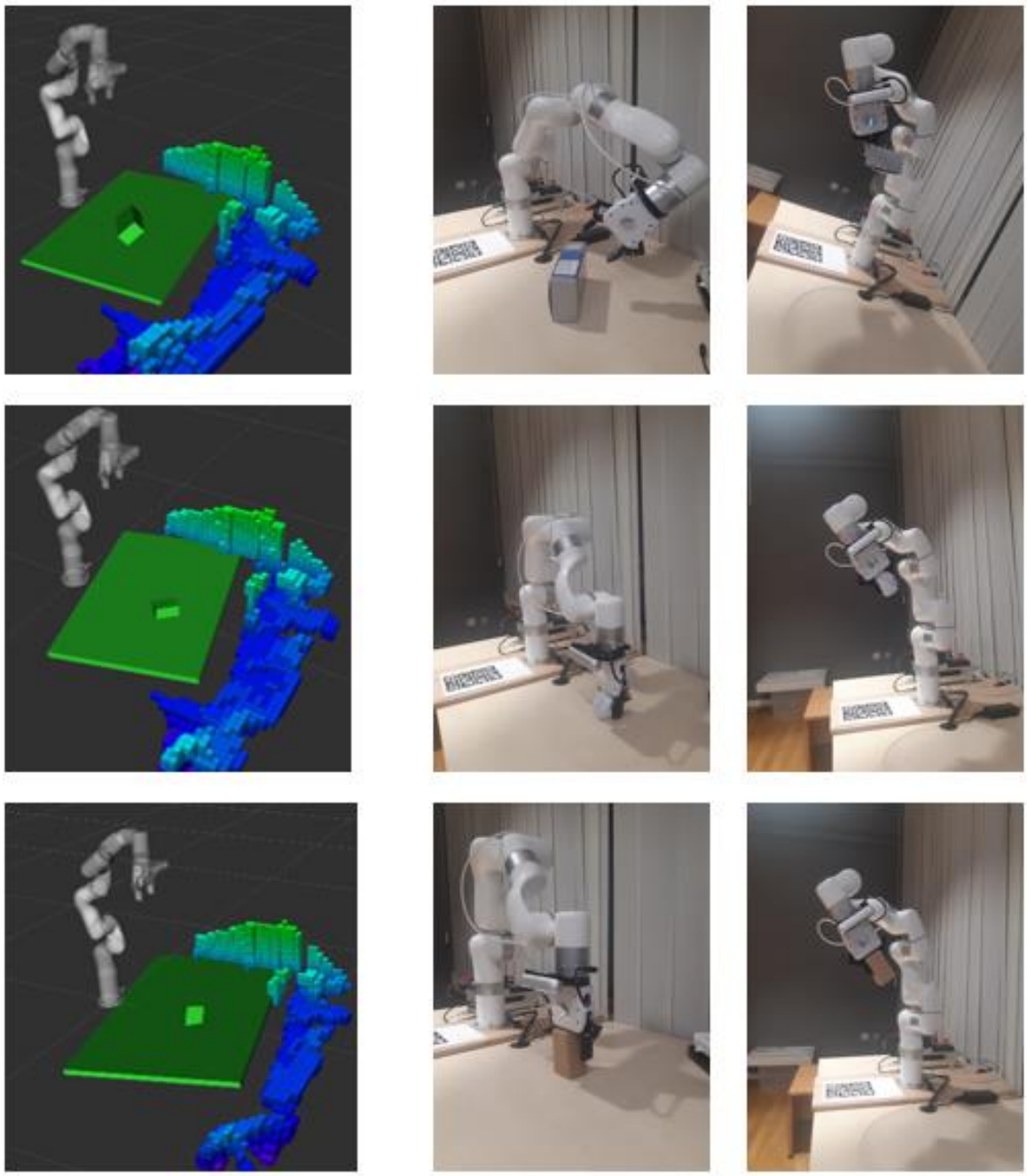
Za evaluaciju eksperimentalnog postava sustava i cjelokupnog softvera napravljen je eksperiment strategije sakupljanja otpada. Evaluacija se sastojala od nasumično postavljanih objekata u prostoru različitih dimenzija i orijentacija te sakupljanje istih.

Postupak sakupljanja sastojao se od akvizicije oblaka točaka iz tri različita kuta prema definiranoj površini. Nakon sakupljanja više različitih oblaka točaka izvršavala se njihova obrada te na kraju klasifikacija objekta u prostoru pomoću YOLO mreže. Nakon detekcije i izolacije objekta u prostoru radilo se preklapanje primitiva s oblakom točaka kako bi se u ROS-u mogao dodati objekt u okolinu za manipulaciju.

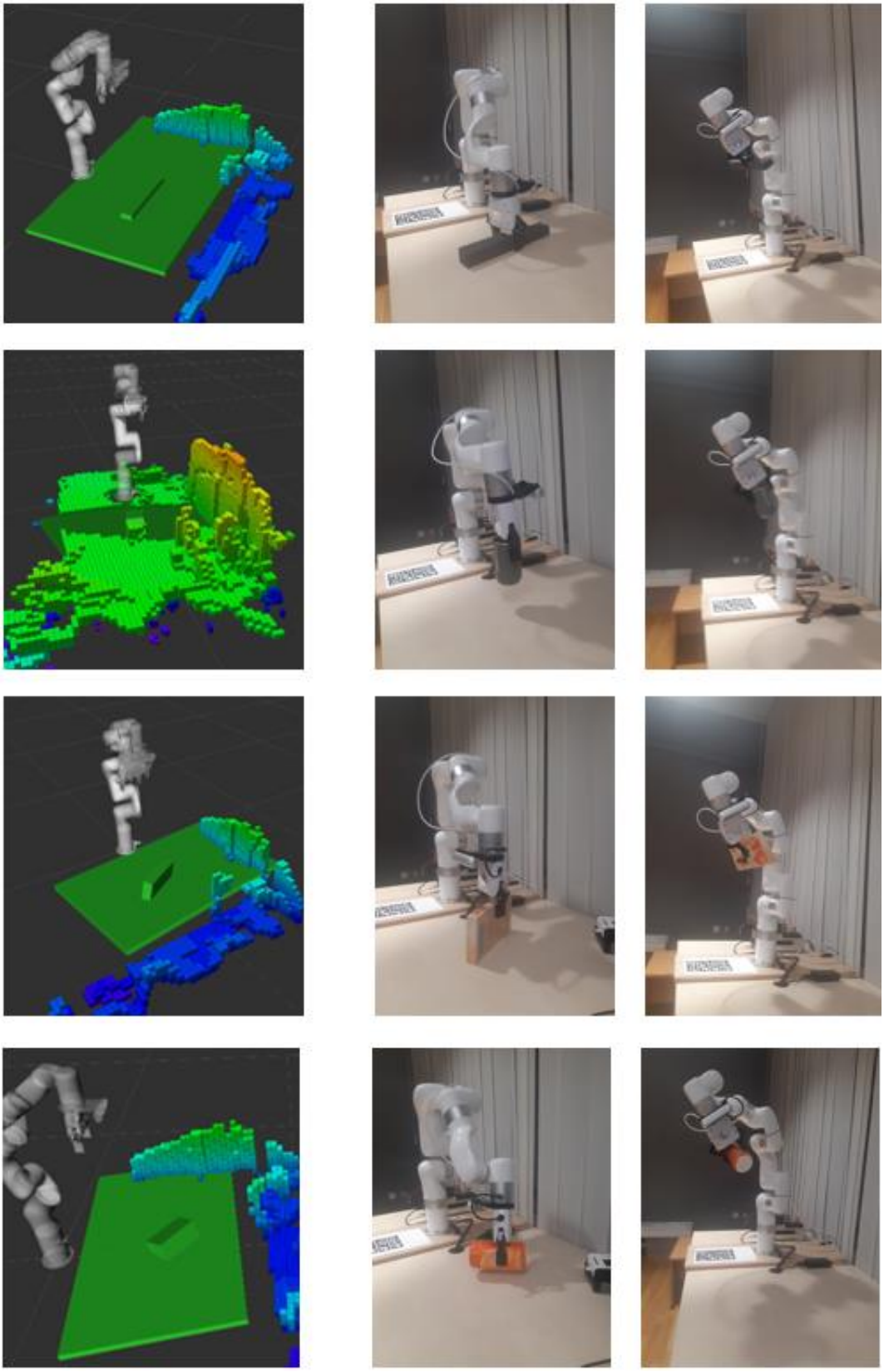
Strategija sakupljanja otpada pomoću robotske ruke u 90% slučajeva uspješno će izvršiti radnju sakupljanja otpada. Ovaj zaključak izveden je na temelju 20 različitih uzastopnih testova. U 10% slučajeva gdje sakupljanje otpada nije bilo uspješno dolazilo je zbog nepotpune snimke objekta što je rezultiralo krivom orijentacijom objekta u prostoru. Zbog krive orijentacije u prostoru dolazilo je do krivih ulaznih putanja robotske ruke prema objektu, što je rezultiralo neuspjehom u hvatanju istog.

Prilikom testiranja softverskog rješenja primijećeno je da parametri unutar ROS-a za kretanje ruke moraju biti egzaktno i precizno definirani inače ruka neće biti u mogućnosti isplanirati putanje za hvatanje objekta. Dobivena površina u prostoru morala se spustiti po z osi. Automatskim generiranjem iz oblaka točaka površina je uzrokovala softversku koliziju prilikom hvatanja objekta što je znalo rezultirati nemogućnošću hvatanja objekta.

Posljednji problem koji se javljao prilikom sakupljanja otpada u prostoru je moguća promjena orijentacije/smjera koordinatnog sustava objekta, odnosno da je pozitivni smjer z osi može biti orijentiran prema podlozi, a ne prema gore. Ovaj fenomen je znao rezultirati nemogućnosti stvaranja putanji prilaska objektu što je rezultiralo neuspjehom u MoveIt-u planiranja putanja hvatanja objekta. Problem krive orijentacije koordinatnog sustava mogao bi se riješiti povećanjem prilaznih trajektorija robotske ruke prema objektu. Na slikama ispod prikazan je proces strategije hvatanja objekta u prostoru.



Slika 4.1 Strategija sakupljanja otpada u prostoru



Slika 4.2 Strategija sakupljanja otpada u prostoru

5. ZAKLJUČAK

U okviru ovog diplomskog rada uspješno je razvijen eksperimentalni postav sustava za sakupljanje otpada pomoću robotske ruke i stereovizijskog sustava. Unutar rada detaljno je opisan, dokumentiran i obrazložen način izrade sustava, komunikacija unutar sustava, akvizicije i obrada podataka dobivenih iz senzora. Eksperimentalnom fazom ovog rada potvrđena je mogućnost integracije robotskog sustava u različite mobilne platforme te mogućnost raznolike primjene sustava.

Sustav je razvijen pomoću moćnog robotskog operativnog sustava što mu pruža mogućnost jednostavne distribucije na različite robotske ruke i mnogobrojne 3D senzore. U radu je implementiran ROS paket za *Eye in Hand* kalibraciju pozicije kamere na robotskoj ruci što je omogućilo interakciju sustava s okolinom. Pomoću sterevozijskog sustava vrši se akvizicija oblaka točaka podloge, odnosno okoline iz različitih pozicija kako bi slika svijeta oko njega bilo što potpunija. Iz prikupljenih 3D podataka pomoću MoveIt ROS paketa planirane su trajektorije robotske ruke u područjima gdje je to bilo moguće napraviti bez kolizije s okolinom. Pomoću YOLO v5 mreže detektirani su i grubo lokalizirani objekti u prostoru što je omogućilo modularnu primjenu sustava u različite svrhe. Koristeći se različitim algoritmima unutar programske knjižnice Open3D napravljena je manipulacija, segmentacija, klasifikacija i preklapanje oblaka točaka s različitim geometrijskim primitivima. Dobiveni geometrijski primitivi s pripadajućim dimenzijama i orijentacijama u prostoru dodavani su u ROS scenu s ciljem izuzimanja i rukovanja robotskom rukom. U ovom radu razvijen je algoritam za automatsko generiranje različitih vektora prilaska objektima u ROS okolini i manipulaciju istima.

Analizom rezultata razvijenog sustava za robotsku manipulaciju otpadom u više različitih scenarija okoline, zaključuje se da sustav ima visoku razinu uspješnosti hvatanja zasebnih objekata na podlozi. Razvijen eksperimentalni sustav posjeduje visoku razinu autonomije i „svjesnosti“ okoline, što ga čini dobrom eksperimentalnom platformom za daljnji razvoj i buduću primjenu u realnim sustavim za rukovanje otpadom.

LITERATURA

[1] https://more.slobodnadalmacija.hr/om/vijesti/europski-parlament-trazi-hitno-smanjenje-morskog-otpada-prosjecni-potrosac-sredozemnih-skoljkasa-svake-godine-proguta-oko-11-000-komadica-plastike-1086803, Pristupljeno : 6.7.2022.
[2] https://www.ronis.hr/slike/velike/robotski-usisavac-irobot-roomba-694-roomba-694_4.jpg, Pristupljeno : 6.7.2022.
[3] https://www.bostondynamics.com/sites/default/files/2021-08/atlas-dynamic.jpg, Pristupljeno : 6.7.2022.
[4] https://www.researchgate.net/profile/Dimitrios-Kanoulas/publication/351662241_Garbage_Collection_and_Sorting_with_a_Mobile_Manipulator_using_Deep_Learning_and_Whole-Body_Control/links/60a3e620299bf1569527998f/Garbage-Collection-and-Sorting-with-a-Mobile-Manipulator-using-Deep-Learning-and-Whole-Body-Control.pdf?origin=publication_detail, Pristupljeno : 6.7.2022.
[5] https://clearpathrobotics.com/blog/2018/11/warthog-ugv-mobile-manipulation-research-autotrans/, Pristupljeno : 6.7.2022.
[6] https://www.mybotshop.de/Franka-Emika-Panda-FCI-Licence, Pristupljeno : 6.7.2022.
[7] https://ste.education/en/product/xarm/, Pristupljeno : 6.7.2022.
[8] https://www.robotshop.com/en/xarm-6-dof-robotic-arm.html, Pristupljeno : 6.7.2022.
[9] https://github.com/xArm-Developer, Pristupljeno : 6.7.2022.
[10] https://www.robotshop.com/en/xarm-gripper.html, Pristupljeno : 6.7.2022.
[11] https://developer.nvidia.com/embedded/jetson-xavier-nx-devkit, Pristupljeno : 6.7.2022.
[12] https://developer.nvidia.com/embedded/jetson-xavier-nx, Pristupljeno : 6.7.2022.
[13] https://github.com/IntelRealSense, Pristupljeno : 6.7.2022.
[14] https://www.intelrealsense.com/depth-camera-d455/, Pristupljeno : 6.7.2022.
[15] https://www.researchgate.net/publication/340464313_Evaluation_of_ROS_and_Gazebo_Simulation_Environment_using_TurtleBot3_robot, Pristupljeno : 6.7.2022.
[16] https://apps.unizg.hr/rektorova-nagrada/javno/akademske-godine/2020/nagradeni-radovi, Pristupljeno : 6.7.2022.
[17] https://pjreddie.com/darknet/yolo/, Pristupljeno : 6.7.2022.
[18] https://machinelearningmastery.com/object-recognition-with-deep-learning/, Pristupljeno : 6.7.2022.
[19] https://github.com/leggedrobotics/darknet_ros, Pristupljeno : 6.7.2022.
[20] https://pjreddie.com/darknet/, Pristupljeno : 6.7.2022.
[21] http://www.open3d.org/docs/release/tutorial/geometry/pointcloud.html, Pristupljeno : 6.7.2022.
[22] https://moveit.picknik.ai/galactic/doc/concepts/move_group.html, Pristupljeno : 6.7.2022.
[23] https://www.youtube.com/watch?v=xQ79ysnrzUk&ab_channel=PickNikRobotics, Pristupljeno : 6.7.2022.
[24] https://github.com/ros-planning/moveit/issues/1070, Pristupljeno : 6.7.2022.
[25] https://github.com/leggedrobotics/darknet_ros, Pristupljeno : 6.7.2022.
[26] https://www.youtube.com/watch?v=nxd4pZm3Wfo&ab_channel=%EC%A0%95%EC

%9B%90%EC%97%90%EC%8A%A4%EC%97%90%ED%94%84%EC%97%90%EC%9D%B4-CWSFA, Pristupljeno : 6.7.2022.
[27] http://docs.ros.org/en/noetic/api/moveit_msgs/html/msg/Grasp.html, Pristupljeno : 6.7.2022.

PRILOZI

Cjelokupni kod dostupan je na git-u: https://github.com/Jan190