# Robotsko rukovanje predmetima rada u nestrukturiranoj radnoj okolini

**Cicvarić, Eugen**

**Master's thesis / Diplomski rad**

**2018**

*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:235:080050

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2024-04-25**

*Repository / Repozitorij:*

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

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

# DIPLOMSKI RAD

**Eugen Cicvarić**

Zagreb, 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

# DIPLOMSKI RAD

Mentor: Student:

Prof. dr. sc. Bojan Jerbić, dipl. ing. Eugen Cicvarić

Zagreb, 2018.

*Izjavljujem da je ovaj rad izrađen samostalno koristeći se navedenom literaturom te stečenim znanjem na Fakultetu strojarstva i brodogradnje u Zagrebu.*

*Zahvaljujem mentoru prof. dr.sc. Bojanu Jerbiću, te dr. sc. Filipu Šuligoju na savjetima i stručnom znanju koje su pružili prilikom izrade ovog rada.*

*Zahvaljujem se svim djelatnicima laboratorija za robotiku i automatizaciju proizvodnih sustava na savjetima i pozitivnoj radnoj atmosferi.*

*Također zahvaljujem obitelji i prijateljima koji su bili podrška tijekom cijelog studija.*

*U Zagrbu, 20. studenog 2018.*

*Eugen Cicvarić*

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<br>Fakultet strojarstva i brodogradnje | |
|---|---|
| Datum: | Prilog: |
| Klasa: | |
| Ur. broj: | |

# DIPLOMSKI ZADATAK

Student: **EUGEN CICVARIĆ** Mat. br.: 0069059782

Naslov rada na hrvatskom jeziku: **Robotsko rukovanje predmetima rada u nestrukturiranoj radnoj okolini**

Naslov rada na engleskom jeziku: **Robotic handling of workpieces in unstructured working environment**

Opis zadatka:

U sklopu diplomskog zadatka potrebno je riješiti problem robotskog izuzimanja predmeta rada iz transportne kutije (eng. bin picking) koristeći 3d vizijski sustav (MS Kinect) i robotsku ruku (Universal robot UR5). Potrebnc je proučiti postojeće algoritme i primijeniti metode za filtriranje šumova, prepoznavanje prostornih značajki na temelju oblaka točaka i registraciju različitih 3d snimaka koristeći PCL (Point Cloud Library) programski paket. Potom je potrebno razviti odgovarajući program za vizijski sustav koji će omogućiti vizualno prepoznavanje te lokalizaciju 3D objekata u radnoj okolini robota. Nakon prepoznavanja i lokalizacije objekta, dobivene informacije treba transformirati u koordinatni sustav robota i povezati s robotskim programom za rukovanje predmetima rada u nestrukturiranoj okolini. Za komunikaciju vizijskog sustav i robota koristiti TCP/IP protokole. Razvijenu primjenu potrebno je provjeriti na raspoloživoj laboratorijskoj opremi.

Zadatak zadan:
27. rujna 2018.

Rok predaje rada:
29. studenog 2018.

Predviđeni datum obrane:
05. prosinca 2018.
06. prosinca 2018.
07. prosinca 2018.

Zadatak zadao:
prof. dr. sc. Bojan Jerbić

Predsjednica Povjerenstva:
prof. dr. sc. Biserka Runje

# SADRŽAJ

POPIS SLIKA ..... IV
POPIS TABLICA ..... VII
POPIS OZNAKA ..... IX
SAŽETAK ..... X
SUMMARY ..... XI
1. UVOD ..... 1
2. ELEMENTI SUSTAVA ..... 3
2.1 Point Cloud Library ..... 4
2.1.1 PCD format datoteke ..... 7
2.2 Cmake ..... 8
2.2.1 Konfiguriranje CMake-a ..... 9
2.3 Visual Studio ..... 10
2.4 Microsoft Kinect v2 ..... 11
2.4.1 Način rada Kinect v2 kamere ..... 12
3. PROBLEM PREPOZNAVANJA I PROSTORNA LOKALIZACIJA OBJEKATA ..... 14
3.1 Teoretska podloga ..... 15
3.1.1 Matrica homogene transformacije ..... 15
3.1.2 Eulerovi kutovi ..... 16
3.1.3 Axis – angle zapis ..... 17
3.1.4 Euklidska udaljenost ..... 18
3.2 Metoda ekstrakcije klastera ..... 19
3.2.1 Filtracija ..... 20
3.2.1.1 Passthrough Filter ..... 21
3.2.1.2 Voxel Filter ..... 22
3.2.1.3 Statistical Outlier Removal Filter ..... 23
3.2.2 Rekonstrukcija površine ..... 23

3.2.3 Uklanjanje podloge ..... 25
3.2.3.1 RANSAC segmentacija ..... 25
3.2.3.2 RGB filtracija ..... 27
3.2.4 Ekstrakcija klastera ..... 27
3.2.5 Granične kutije ..... 28
3.2.6 Prednosti i nedostaci metode ekstrakcije klastera ..... 29
3.3 Metoda lokalnih značajki ..... 31
3.3.1 Obrada oblaka točaka ..... 32
3.3.2 Normale ..... 34
3.3.3 Ključne točke ..... 36
3.3.4 Deskriptori ..... 41
3.3.4.1 Shot deskriptor ..... 46
3.3.4.2 CSHOT deskriptor ..... 46
3.3.6 ICP registracija ..... 47
3.3.7 Konačna matrica transformacije ..... 48
3.3.8 Izrada modela ..... 49
3.3.9 Prednosti i nedostaci metode lokalnih značajki ..... 50
4. IMPLEMENTACIJA METODA PREPOZNAVANJA POMOĆU PCL ..... 52
4.1 Metoda ekstrakcije klastera ..... 52
4.1.1 Passthrough filter ..... 53
4.1.2 Voxel filter ..... 54
4.1.3 Statistical outlier removal filter ..... 55
4.1.4 Rekonstrukcija površine ..... 56
4.1.5 Uklanjanje podloge – RANSAC segmentacija ..... 57
4.1.6 Uklanjanje podloge – RGB filtracija ..... 58

4.1.7 Ekstrakcija klastera ..... 59
4.1.8 Granične kutije ..... 60
4.2 Metoda lokalnih značajki ..... 61
4.2.1 Izračun normala ..... 61
4.2.2 Izračun ključnih točaka ..... 62
4.2.3 Izračun deskriptora ..... 63
4.2.4 Usporedba i vrednovanje deskriptora ..... 63
4.2.5 ICP registracija ..... 66
5. POVEZIVANJE S ROBOTOM ..... 67
5.1 Ostvarivanje komunikacije ..... 68
5.2 Program na robotu ..... 70
6. EKSPERIMENTALNI POSTAV I REZULTATI ..... 71
6.1 Objekt s više od tri ravnine simetrije ..... 73
6.2 Objekt s tri ravnine simetrije ..... 75
6.3 Objekt s jednom ravnine simetrije ..... 77
6.4 Objekt bez ravnine simetrije ..... 79
7. ZAKLJUČAK ..... 81
LITERATURA ..... 83
PRILOZI ..... 85

# POPIS SLIKA

Slika 2.1 Elementi sustava i međusobna komunikacija ..... 3

Slika 2.2. Point Cloud Library logo [1] ..... 4

Slika 2.3. Primjer 3D podatka u obliku oblaka točaka ..... 4

Slika 2.4. Podjela PCL-a u manje segmente [1] ..... 5

Slika 2.5. ASCII format PCD datoteke ..... 7

Slika 2.6. Binarni format PCD datoteke ..... 7

Slika 2.7. CMake logo [4] ..... 8

Slika 2.8. CMake sučelje na Windows operativnom sustavu ..... 8

Slika 2.9. Konfigurirane datoteke putem CMake-a ..... 9

Slika 2.10. Visual Studio logo [6] ..... 10

Slika 2.11. Kinect v2 – pozicija kamera i IR emitera ..... 11

Slika 2.12. Stvaranje 3D podatka Kinecta v2 koristeći ToF tehnologiju i senzor boje ..... 12

Slika 3.1. Problematika određivanja matrice tranfsormacije između dva koordinatna sustava 15

Slika 3.2. Desnokretni pravokutni koordinatni sustav [10] ..... 16

Slika 3.3. Eulerovi kutovi ..... 16

Slika 3.4. Axis angle vektor ..... 17

Slika 3.5. Euklidska udaljenost između dvije točke ..... 18

Slika 3.6. Dijagram toka – metoda ekstrakcija klastera ..... 19

Slika 3.7. Scena prije primjenjenih filtracija ..... 20

Slika 3.8. Voxel filter u koracima, a) oblak točaka se aproksimira graničnim kutijama, b) u graničnoj kutiji se nalazi određeni broj točaka, c) aproksimira se centroid granične kutije dok se ostale uklanjaju ..... 22

Slika 3.9. Položaj normala na površini prije i nakon korištenja rekonstrukcije površine ..... 24

Slika 3.10. Dijagram algoritma za metodu lokalnih značajki ..... 32

Slika 3.11. Normala na površinu, crveno područe – površina objekta, plavo područje – tangnetna ravnina ..... 34

Slika 3.12. Dvostruka orijentacija normala ..... 34

Slika 3.13. Različita orijentacija normala ..... 35

Slika 3.14. Ujednačena orijentacija normala ..... 35

Slika 3.15. Plavom bojom označene ključne točke na predmetima ..... 36

Slika 3.16. Primjer objekata promatranih u testu ..... 37

Slika 3.17. Ponovljivost za rotaciju od 5° ..... 38

Slika 3.18. Ponovljivost za rotaciju od 15° ..... 38

Slika 3.19. Ponovljivost za rotaciju od 25° ..... 38

Slika 3.20. Ponovljivost za rotaciju od 35° ..... 38

Slika 3.21. Ponovljivost za rotaciju od 45° ..... 38

Slika 3.22. Ponovljivost za translaciju ..... 38

Slika 3.23. Ponovljivost za rotaciju od 10° - tipkovnica ..... 39

Slika 3.24. Ponovljivost za rotaciju od 20° - tipkovnica ..... 39

Slika 3.25. Ponovljivost za rotaciju od 10° - dječja kolica ..... 39

Slika 3.26. Ponovljivost za rotaciju od 20° - dječja kolica ..... 39

Slika 3.27. Ponovljivost za rotaciju od 10° - vrč ..... 39

Slika 3.28. Ponovljivost za rotaciju od 20° - vrč ..... 39

Slika 3.29. Usporedba deskriptora s dvije identične scene, , točka $\boldsymbol{P1} \cong \boldsymbol{Q1}$, $\boldsymbol{P2} \cong \boldsymbol{Q2}$, $\boldsymbol{P3} \cong \boldsymbol{Q3}$ ..... 42

Slika 3.30. Različiti setovi podataka, a) miješani set podataka različite kvalitete, b) spacetime stereo set – srednja kvaliteta c) kinect – niska kvaliteta, d) laser skener – visoka kvaliteta, e) skeniranje objekata – visoka kvaliteta ..... 43

Slika 3.31. Rezultati za set podataka a) ..... 44

Slika 3.32. Rezultati za set podataka b) ..... 44

Slika 3.33. Rezultati za set podataka c) ..... 44

Slika 3.34. Rezultati za set podataka d) ..... 44

Slika 3.35. Rezultati za set podataka e) ..... 45

Slika 3.36. Struktura SHOT deskriptora oko središnje točke ..... 46

Slika 3.37. Poravnanje dvaju oblaka u *n* iteracija ICP algoritmom [24] ..... 47

Slika 3.38. Transformacije između oblaka točaka ..... 48

Slika 3.39. Fino podešavanje matrice transformacije prepoznatog objekta ..... 49

Slika 4.1. Početni nefiltrirani oblak točaka ..... 52

Slika 4.2. Scena nakon passthrough filtera ..... 53

Slika 4.3. Scena nakon voxel filtera ..... 54

Slika 4.4. Scena nakon statistical outlier removal filtera ..... 55

Slika 4.5. Prije (lijevo) i nakon (desno) MLS rekonstrukcije površine ..... 56

Slika 4.6. Scena bez poodloge ..... 57

Slika 4.7. Prikaz objekta omeđenog graničnom kutijom ..... 60

Slika 4.8. Dobro odabran k broj susjeda (lijevo) i loše odabran k broj susjeda (desno) [25] ..... 61

Slika 4.9. ISS3D ključne točke ..... 62

Slika 4.10. Pronađene korespondencije između modela i scene ..... 65

Slika 5.1. Zglobovi i segmenti UR5 robotske ruke [28] ..... 67

Slika 5.2. Odlaganje objekata nakon robotskog izuzimanja ..... 70

Slika 6.1. Eksperimentalni postav ..... 72

Slika 6.2. Fiksiranje kamere ..... 72

Slika 6.3. Fotografija i .pcd datoteka testiranog objekta ..... 73

Slika 6.4. Fotografija i .pcd datoteka ispitanog objeka ..... 75

Slika 6.5. Fotografija i .pcd datoteka ispitanog objekta ..... 77

Slika 6.6. Fotografija i .pcd datoteka ispitanog objekta ..... 79

POPIS TABLICA

Tablica 1. Osnovne specifikacije Kinecta v2 ..... 11

Tablica 2. Prosječno vrijeme potrebno detektorima za izračun ključnih točaka izračunato na temelju više objekata [15] ..... 40

Tablica 3. Neki od prisutnih lokalnih deskriptora unutar PCL-a ..... 43

Tablica 4. Passthrough filter ..... 53

Tablica 5. Voxel filter ..... 54

Tablica 6. Statistical outlier removal filter ..... 55

Tablica 7. Rekonstrukcija površine ..... 56

Tablica 8. RANSAC segmentacija ..... 57

Tablica 9. RGB filtracija ..... 58

Tablica 10. Ekstrakcija klastera ..... 59

Tablica 11. Izračun normala ..... 61

Tablica 12 ISS3D ključne točke ..... 62

Tablica 13. Izračun deskriptora ..... 63

Tablica 14. Lokalni referentni koordinatni sustav ..... 64

Tablica 15. Hough3D vrednovanje korespondencija ..... 64

Tablica 16. ICP registracija ..... 66

Tablica 17. Specifikacije UR5 robotske ruke [27] ..... 67

Tablica 18. Dostupni serveri na UR kontroleru ..... 69

Tablica 19. Rezultati translacije za objekt s neograničenim brojem ravnine simetrije ..... 73

Tablica 20. Rezultati rotacije za objekt s neograničenim brojem ravnine simetrije ..... 74

Tablica 21. Rezultati translacije za objekt s tri ravnine simetrije ..... 75

Tablica 22. Rezultati rotacije za objekt s tri ravnine simetrije ..... 76

Tablica 23. Rezultati translacije za objekt s jednom ravninom simetrije ..... 77

Tablica 24. Rezultati rotacije za objekt s jednom ravninom simetrije ..... 78

Tablica 25. Rezultati translacije za objekt bez ravnine simetrije ..... 79

Tablica 26. Rezultati rotacije za objekt bez ravnine simetrije ..... 80

# POPIS OZNAKA

| Oznaka | Jedinica | Opis |
|---|---|---|
| $\varphi$ | rad | Fazni pomak između emitiranog i primljenog signala |
| $\theta$ | - | Trodimenzionalni vektor rotacije |
| $\emptyset$ | rad | Kut skretanja |
| $\theta$ | rad | Kut nagiba |
| $\psi$ | rad | Kut valjanja |
| A | - | Matrica transformacije |
| $b_{gk}$ | mm | Širina granične kutije |
| c | m/s | Brzina svjetlosti |
| d | m | Udaljenost točke od senzora kamere |
| $d_{gk}$ | m | Duljina granične kutije |
| E | m | Euklidska udaljenost |
| e | - | Jedinični vektor |
| f | Hz | Frekvencija |
| $h_{gk}$ | mm | Visina granične kutije |
| k | - | Broj najbližih susjeda |
| P | - | Oblak točaka |
| $P_i$ | - | Točka *i* u skupu P |
| R | - | Matrica rotacije |
| T | - | Vektor translacije |

# SAŽETAK

U radu su razvijena i implementirana dva načina prepoznavanja te prostorne lokalizacije objekata na sceni koristeći senzor Kinect v2 i softversku pozadinu Point Cloud Library. Pri tome korišten je robot UR5 koji sa servera, putem TCP protokola, prima podatke o lokaciji i pretvara ih u naredbe gibanja. Kako bi dohvaćen oblak točaka iz senzora bio prikazan sa što više konzistentnosti potrebno je primjeniti filtraciju i algoritme za rekonstrukciju površine.

Za obje metode, ekstrakcije klastera i lokalnih značajki, napravljena je teoretska pozadina i softverska implementacija na temelju koje su utvrđene prednosti, odnosno nedostaci. Za metodu lokalnih značajki napravljena su eksperimentalna ispitivanja preciznosti i prikaz rada u praktičnoj primjeni. Ovakvi pristupi prepoznavanja i lokalizacije neovisni su o geometriji promatranih objekata i moguće ih je prilagoditi za sve industrijske robote.

Osim spomenutog Point Cloud Library-a korišteni su softveri CMake za izradu projekta, te Visual Studio 2015 za C++ programiranje.

Struktura rada sastoji se od uvoda, osnovnih informacija o korištenim elementima sustava i njihovoj interakciji, zatim je razrađena teorija za razvijene metode, slijedi implementacija teoretske pozadine pomoću PCL-a, povezivanje s robotom i u konačnici ispitivanje preciznosti te zaključak.

Ključne riječi: Robotska vizija, prepoznavanje, lokalizacija, Point Cloud Library, oblak točaka, UR5

# SUMMARY

Two ways of detecting and spatial localization of objects are developed and implemented in thesis using the Kinect v2 sensor and the Point Cloud Library as software background. The UR5 robot, which uses the TCP connection, receives location data and converts them into the motion instructions. In order to capture consistent points cloud from the sensor, it is necessary to apply filtration and surface reconstruction algorithms.

For both methods, cluster extraction and local features, a theoretical background and software implementation were developed based on which the advantages and disadvantages were identified. For the local feature method, experimental examinations of precision and work presentation were performed in practical application. Such approaches to recognition and localization are independent of the geometry of the observed objects and can be adapted to all industrial robots.

In addition to the mentioned Point Cloud Library, used softwares are also CMake and Visual Studio 2015 for C ++ programming.

The structure of the work consists introduction, basic information of the system elements and their interaction, then elaborated theory for developed methods followed by the implementation using PCL, realization of connection with the robotic arm, also experimental testing about the precision and conclusion.

Key words: Robotic vision, detection, localization, Point Cloud Library, point cloud, UR5

# 1. UVOD

Automatizacija proizvodnih procesa značajno je utjecala na mogućnost povećanja proizvodnih količina, efikasniju uporabu materijala, bolju i ujednačeniju kvalitetu proizvoda, sigurnost osoblja, a istodobno eliminirajući grešku uzrokovanu ljudskim faktorom kao produktom umora i gubitkom koncentracije, posebice kod monotonih i jednoličnih poslova. Čovjek je također suočen s ostalim ograničenjima kao što su nedovoljno dobra ponovljivost i preciznost, doseg nepristupačnih mjesta te rukovanje samo teretima manje mase, stoga se u serijskoj proizvodnji teži pronalasku odgovarajućih robota ili manipulatora i senzora.

Robotika, kao jedan od vodećih čimbenika u razvoju automatskih rješenja, snažno je utjecala na unaprjeđenje industrijske proizvodnje, međutim, još uvijek postoji veliki izazov za njen napredak i razvoj u korist povećanja proizvodnje i bolje integracije s okolinom.

Razvojem senzora u kombinaciji s pametnim softverskim rješenjima ideja je robotu osigurati što kvalitetnije informacije o okolini u kojoj se nalazi, što podrazumijeva rješavanje niza zadataka poput izbjegavanja statičkih i dinamičkih prepreka, ostvarivanja sigurnog okruženja za čovjeka, paralelan i sinkron rad s čovjekom i/ili s drugim robotima, te u konačnici uspješno izvršavanje zadanog problema kao što je primjerice izuzimanje objekata.

U okviru diplomskog zadatka razmotrit će se problem prepoznavanja objekata u nesređenoj okolini i njihova prostorna lokalizacija. Problemu će se prvotno pristupiti s teorijskog stajališta nakon čega slijedi eksperimentalna provjera i analiza dobivenih rezultata.

Podaci sa scene prikupljeni su u obliku oblaka točaka, tj. setu podataka koji ima mogućnost prikaza prostora u tri dimenzije, sa Microsoft Kinect v2 vizijskim senzorom.

Pad cijena 3D vizijskih senzora povećao je njihovu sveopću dostupnost široj masi korisnika što je rezultiralo stvaranjem kreativnih zajednica, a u konačnici i stvaranju složenih alata za procesiranje 3D podataka. Jedan od takvih alata, koji ujedno predstavlja i glavnu softversku pozadinu ovog rada, je PCL (Point Cloud Library). PCL je knjižnica algoritama otvorenog koda za obradu 3D podataka, odnosno podataka u obliku oblaka točaka.

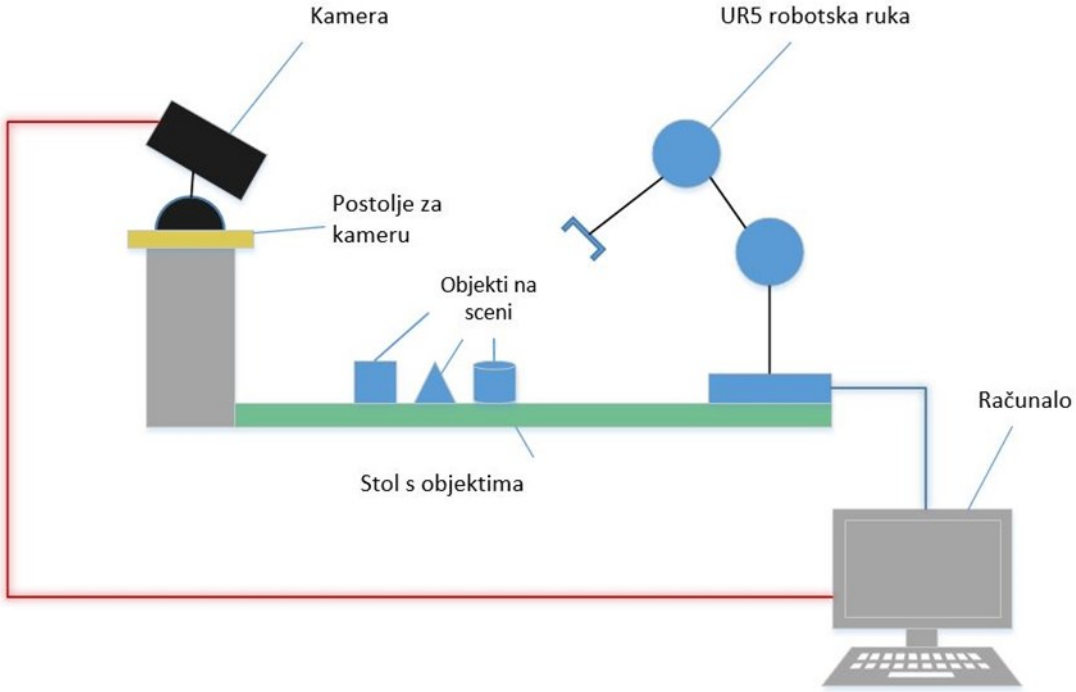
Obrada podataka uključuje filtracije u svrhu uklanjanja nepotrebnih podataka, poboljšavanje kvalitete preostalih podataka, prikupljanje korisnih značajki iz scene koje opisuju traženi objekt, te u konačnici detekciju željenog objekta i pronalazak njegove prostore lokacije.

Rješavanje problema vizijskih sustava često nije egzaktno, tj. postoji više načina kako pristupiti problemu, a sustavi obično nisu robusni što znači da male promjene osvjetljenja, položaja kamere ili promjena jednog i/ili više parametara u softveru može značajno utjecati na konačni rezultat. Tako da će se u ovome radu pristupiti na više načina i korištenjem više elemenata kako bi se mogao donijeti što konkretniji zaključak o utjecajnim parametrima u procesu.

# 2. ELEMENTI SUSTAVA

Tehnički sustav predstavlja uređenu cjelinu, odnosno skup elemenata koji međusobno dijele potrebne informacije i izvršavaju željenu radnju. U diplomskom radu od softverskih elemenata korišten je Point Cloud Library 1.8.1 (PCL), CMake 3.10.1 i Visual Studio 2015, dok hardverske komponente čini Kinect v2 kamera i industrijski robot UR5. Funkcija navedenih elemanata te njihova međusobna interakcija detaljnije je opisana u narednim poglavljima.

Kako bi se zadatak uspješno izveo potrebno je osigurati međusobnu kompatibilnost i komunikaciju između softverskih i hardverskih elemenata.



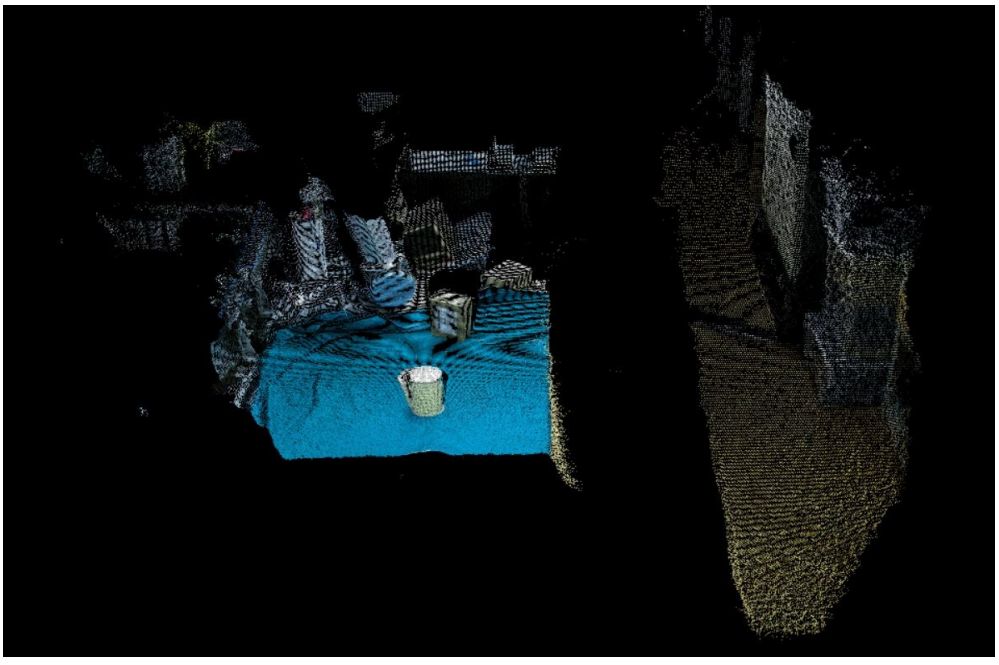

**Slika 2.1 Elementi sustava i međusobna komunikacija**

## 2.1 Point Cloud Library

Point Cloud Library je knjižnica otvorenog koda namijenjena za procesiranje 2D/3D slika i podataka u obliku oblaka točaka, te ujedno predstavlja i glavnu softversku pozadinu ovog rada.

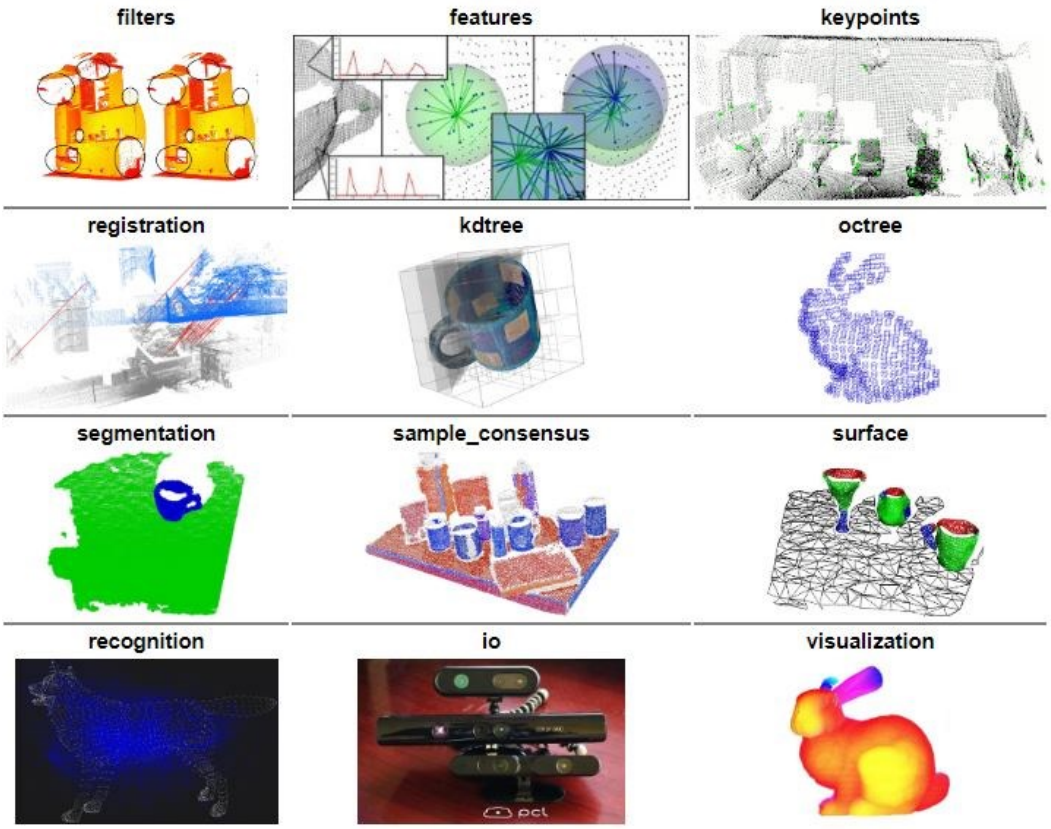
**Slika 2.2. Point Cloud Library logo [1]**

Pojavom cjenovno pristupačnih 3D senzora, kao što je primjerice Kinect v1 namijenjen za igraču konzolu Xbox 360, otvorio se prostor za niz istraživanja i projekata na temu robotske vizije. Gledajući unazad, 3D senzori su do tada bili veoma skupi, a prostorna koordinata dubine sastojala se od tek nekoliko bajtova informacija dobivenih sonarnim ili infracrvenim senzorima. Pošto je robot dobio mogućnost „vidjeti“ prostor u sve tri dimenzije razvio se efikasan alat potreban za obradu takvih informacija – PCL. [2]



**Slika 2.3. Primjer 3D podatka u obliku oblaka točaka**

U cilju jednostavnijeg razvoja i korištenja PCL je podijeljen u nekoliko manjih dijelova kao što je prikazano na slici 3. [3]




**Slika 2.4. Podjela PCL-a u manje segmente [1]**

**pcl_filters –** sadrži funkcije za uklanjanje šumova iz oblaka točaka, odstranjivanje točaka iz podataka na temelju neke značajke kao što je primjerice boja, smanjivanje ukupne veličine podataka radi bržeg procesiranja i slično.

**pcl_features** – mehanizmi za izračunavanje značajki iz 3D podataka. Značajka se može odrediti iz točke, skupine točki ili lokacije u prostoru te daje pojedine geometrijske informacije o lokalnom području oko izabrane točke ili skupine točki.

**pcl_keypoints** – ključne točke ili točke interesa imaju svrhu predstaviti neki objekt ili značajku u manjem broju točaka bez da se izgubi smisao tog objekta ili značajke. Ključne točke trebale bi biti stabilne i karakteristične za promatrani oblak točaka.

**pcl_registration –** ključna ideja je identificirati sličnosti između točaka u dva ili više setova podataka i pronaći podatke o prostornoj transformaciji korištenih oblaka točaka.

**Pcl_kdtree –** omogućava strukturu podataka u kd – stablu koje u kombinaciji s FLANN implementacijom dopušta brzo traženje najbližeg susjeda. Kd stablo je *k* dimenzijsko stablo, u ovom radu ponajviše 3d pošto se manipulira s 3D podacima. Kd stablo je jedna od osnovnih funkcija za traženje korespodencija između grupe točaka ili grupe deskriptora za definiranje lokalnih susjeda.

**pcl_octree –** pruža odgovarajuće funkcije za stvaranje hijerarhijske strukture podataka koja se koristi za prostorno particioniranje, smanjenje rezolucije i pretraživanje podataka.

**pcl_segmetation –** sadrži algoritme potrebne za segmentaciju, odnosno definiranje skupova podataka koji se dijele u klastere. Segmentacija omogućava daljnji rad s manjom količinom podataka što štedi procesorsko vrijeme.

**pcl_sample_consensus –** omogućava detektiranje specifičnih obilježja unutar podataka kao što su linijski, ravninski, cilindrični i/ili sferni modeli.

**pcl_surface –** sadrži funkcije za rekonstrukciju površina dobivenih pomoću 3D senzora, poput primjerice smanjivanja utjecaja konkavnih ili konveksnih površina modela koji nastaju zbog greške senzora. Često se koristi za rekonstrukciju površine podataka s puno šuma ili za podatke koji su nastali iz više skeniranja da bi se postigla njihova jednolična građa.

**pcl_recognition –** sadrži algoritme potrebne za prepoznavanje objekata na sceni, izračunava grubu transformaciju između modela i modela na sceni nakon koje slijedi fino podešavanje i korištenje modula registracije.

**pcl_io –** klase i funkcije za čitanje, odnosno zapisivanje podataka u obliku oblaka točaka koristeći .pcd ili .ply ekstenziju. Također sadrži funkcije za dohvaćanje podataka iz senzora.

**pcl_visualization** – knjižnica koja je napravljena u svrhu prikaza 3D podataka koji su obrađeni algoritmima iz preostalih dijelova PCL-a.

Pristup rješavanju nekog problema, tj. zadatka zahtjeva korištenje više segmenata iz kojih je Point Cloud Library sastavljen.

### *2.1.1 PCD format datoteke*

Unutar Point Cloud Library-a razvijen je i vlastiti format, s ekstenzijom .pcd, za pohranjivanje više-dimenzijskih podataka i njihovo obrađivanje kroz funkcije, tj. algoritme.

Za svaki skup podataka potrebno je prethodno, prije obrade kroz algoritme, pravilno definirati njegovu strukturu, pa tako za primjerice senzor poput Kinecta koji ima sposobnost prikazati prostor u tri dimenzije i u boji, definira se PointXYZRGB datoteka. Ukoliko se ne želi koristiti svojstvo boje definira se PointXYZ podatak, tj. bez RGB nastavka. Unutar PCL-a postoji još niz različitih vrsta podataka, a koji će se koristiti često ovisi o samom algoritmu koji zahtjeva unaprijed definirane ulazne i izlazne značajke.

PCD datoteka nadalje može biti u binarnom ili ASCII formatu, oba formata su potpuno kompatibilna sa svim algoritmima unutar PCL-a i moguće je jednostavno raditi njihovu međusobnu konverziju. U ovom radu korišten je samo binarni format zbog osjetno bržeg procesiranje podataka. Prednost ASCII formata dolazi do izražaja kada je potrebno napraviti samoinicijativnu inspekciju unutar same PCD datoteke jer su jasno čitljive X, Y, Z koordinate točaka ili pojedine izračunate značajke.

```
# .PCD v0.7 - Point Cloud Data file format
VERSION 0.7
FIELDS x y z rgb
SIZE 4 4 4 4
TYPE F F F U
COUNT 1 1 1 1
WIDTH 287
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 287
DATA ascii
0.26287976 -0.054877248 0.50077277 4278255620
0.26591399 -0.053378001 0.50099355 4278255620
0.27104634 -0.048130054 0.50061721 4278255620
0.27127591 -0.046988111 0.50014067 4278255620
0.27614361 -0.0486749 0.50427771 4278255620
0.24364375 -0.04745771 0.50015658 4278255620
0.22883278 -0.014790588 0.50388223 4278255620
0.23937728 -0.05724857 0.50761205 4278386952
0.24108258 -0.055913534 0.50568128 4278386952
0.23789953 -0.050791204 0.50532556 4278255620
0.27732444 -0.050302658 0.50616527 4278255620
0.28690159 -0.029742219 0.50487554 4278386952
0.2705577 -0.010795674 0.50684714 4278386952
0.25175735 -0.00919462 0.50992334 4278255620
0.29281956 -0.0083988868 0.50857365 4278255620
0.25427186 -0.073341876 0.51066101 4278255620
0.25755996 -0.073682174 0.51062769 4278255620
0.2409841 -0.062833317 0.50960141 4278386952
0.27720809 -0.057392932 0.50997317 4278255620
0.24049404 -0.055728115 0.50592875 4278255620
0.22816396 -0.033631731 0.50987422 4278255620
0.2276974 -0.029849708 0.51003325 4278255620
0.22595298 -0.017553307 0.50931543 4278255620
```

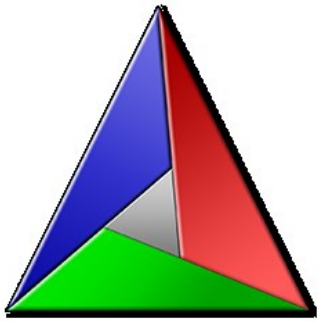
**Slika 2.5. ASCII format PCD datoteke**

```
# .PCD v0.7 - Point Cloud Data file format
VERSION 0.7
FIELDS x y z rgb
SIZE 4 4 4 4
TYPE F F F U
COUNT 1 1 1 1
WIDTH 1445
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 1445
DATA binary
```

**Slika 2.6. Binarni format PCD datoteke**

## 2.2 Cmake

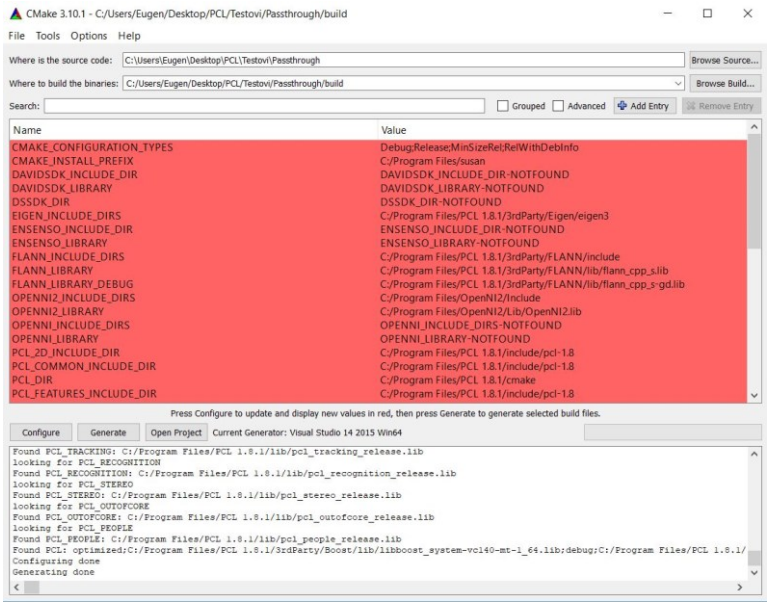
CMake je softver otvorenog koda koji sudjeluje u pripremi projekta automatiziranim dodavanjem izvornih datoteka u projekt i adaptacijom odgovarajućih datoteka potrebnih za izradu konačnog programa, odnosno .exe datoteke u Windowsu, a također podržava i ostala okruženja ovisna o operativnom sustavu koji se koristi.



**Slika 2.7. CMake logo [4]**

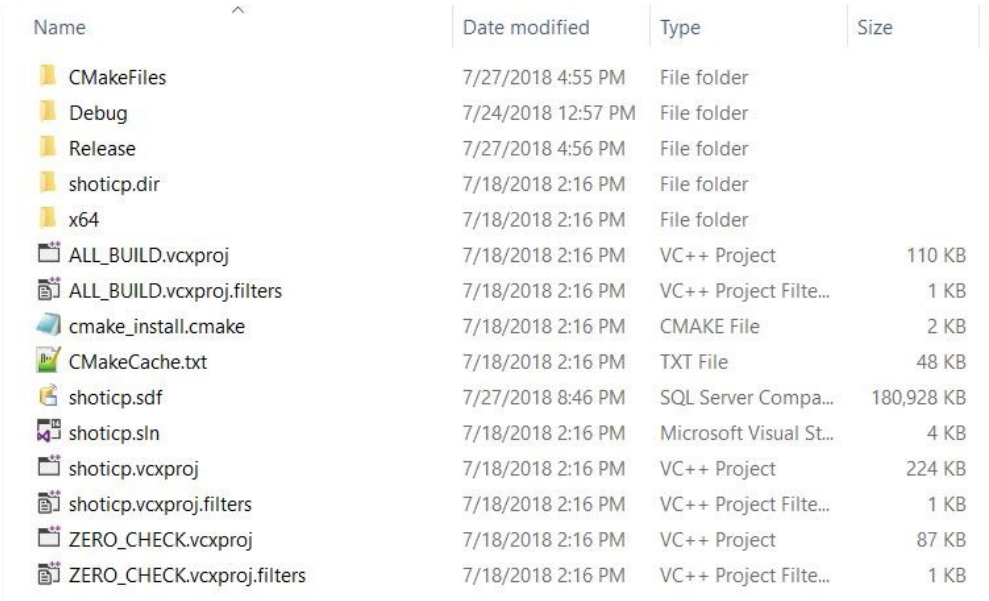
Konfiguracijska datoteka za CMake nalazi se u izvornom direktoriju projekta, s nazivom CMakeLists.txt, preko koje se pozivaju potrebne knjižice, u slučaju diplomskog rada one iz PCL-a, da bi se u konačnici izgradilo nativno okruženje prilagođeno Windowsu, tj. Microsoft Visual Studiu u kojem se gradi izvorni kod. [4]

Jednostavno rečeno CMake predstavlja softver koji povezuje instalirani Point Cloud Library na računalu s kompajlerom, tj. MS Visual Studiom i znatno skraćuje vrijeme pripreme projekta.




**Slika 2.8. CMake sučelje na Windows operativnom sustavu**

Ulaskom u CMake korisnik je nužan definirati direktorij u kojem se nalazi izvorni kod, odnosno algoritam koji obrađuje podatke oblaka točaka pomoću PCL-a s .cpp ekstenzijom datoteke. Zatim se u prethodno odabranom direktoriju napravi poddirektorij, obično naziva *build*, u kojem se pritiskom na Configure i Generate izgrade sve potrebne datoteke prilagođene za MS Visual Studio.



| Name | Date modified | Type | Size |
|---|---|---|---|
| CMakeFiles | 7/27/2018 4:55 PM | File folder | |
| Debug | 7/24/2018 12:57 PM | File folder | |
| Release | 7/27/2018 4:56 PM | File folder | |
| shoticp.dir | 7/18/2018 2:16 PM | File folder | |
| x64 | 7/18/2018 2:16 PM | File folder | |
| ALL_BUILD.vcxproj | 7/18/2018 2:16 PM | VC++ Project | 110 KB |
| ALL_BUILD.vcxproj.filters | 7/18/2018 2:16 PM | VC++ Project Filte... | 1 KB |
| cmake_install.cmake | 7/18/2018 2:16 PM | CMAKE File | 2 KB |
| CMakeCache.txt | 7/18/2018 2:16 PM | TXT File | 48 KB |
| shoticp.sdf | 7/27/2018 8:46 PM | SQL Server Compa... | 180,928 KB |
| shoticp.sln | 7/18/2018 2:16 PM | Microsoft Visual St... | 4 KB |
| shoticp.vcxproj | 7/18/2018 2:16 PM | VC++ Project | 224 KB |
| shoticp.vcxproj.filters | 7/18/2018 2:16 PM | VC++ Project Filte... | 1 KB |
| ZERO_CHECK.vcxproj | 7/18/2018 2:16 PM | VC++ Project | 87 KB |
| ZERO_CHECK.vcxproj.filters | 7/18/2018 2:16 PM | VC++ Project Filte... | 1 KB |

**Slika 2.9. Konfigurirane datoteke putem CMake-a**

### *2.2.1 Konfiguriranje CMake-a*

Kako bi CMake pravilno radio prvotno ga je potrebno konfigurirati za Point Cloud Library i Visual Studio kako bi novonastale datoteke bile odgovarajuće. Konfiguriranje se provodi tako da korisnik prilagodi i integrira nekoliko datoteka u instalacijski direktorij PCL-a Međutim, zahvaljujući developerima iz zajednice otvoren koda (eng.. *Open Source*) koje promiču korištenje PCL-a i sudjeluju u njegovom razvoju, postoje već unaprijed prilagođene verzije PCL-a čijom se instalacijom integriraju sve potrebne konfiguracijske datoteke kako bi CMake funkcionirao za zadani slučaj bez dodatnih podešavanja. [5]

## 2.3 Visual Studio

Visual Studio je integrirano razvojno okruženje (eng. *integrated development environment – IDE)* distribuirano od Microsofta. Koristi se za razvoj računalnih programa, internet stranica, web aplikacija, i mobilnih aplikacija, te podržava programske jezike kao što su Pynton, JavaScript, C#/VB i C++ koji je ujedno korišten u svrhu rada.

**Slika 2.10. Visual Studio logo [6]**

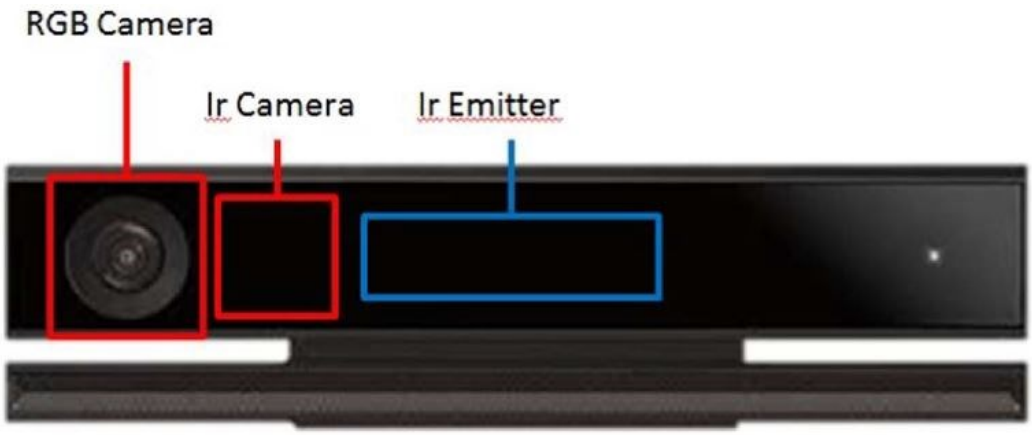
Kako bi se pojednostavilo pisanje koda, posebice velikih i nepreglednih, VS sadrži niz značajki koje olakšavaju snalaženje u takvim situacijama pomoću naprednih pretraživačkih postavki, niz oznaka postavljenih od korisnika i jasno naznačenih oznaka gdje započinju i završavaju programske petlje. Također je prilagođen korisnicima na način da prije izrade .exe datoteke u uređivaču upozorava na pogreške u kodu, koje su ujedno i obrazložene što ubrzava razvoj.

Izvorni kod PCL-a nalazi se u .cpp datoteci i moguće ga je pisati u bilo kojem tekst editoru, no napredne mogućnosti VS uređivača i dobra integracija sa CMake softverom su presudan razlog zašto je odabran Visual Studio.

## 2.4 Microsoft Kinect v2

Kinect v2 je 3D vizijski senzor proizveden od Microsofta, opremljen je RGB kamerom rezolucije 1920x1080 piksela, infracrvenim emiterom i infracrvenom kamerom rezolucije 512x424 piksela koja prikazuje 3D prostor u 217,088 točaka s karakteristikama boje. Od senzora još sadrži mikrofon i LED indikacijsko svjetlo koje služi kao indikator kada kamera radi.

Kao i prethodna verziju Kinect v1, Kinect v2 je također prepoznat kao uređaj velikih potencijalnih mogućnosti s relativnom niskom cijenom pogodnog za izradu niz korisnih aplikacija i aplikacija u svrhu istraživanja.




**Slika 2.11. Kinect v2 – pozicija kamera i IR emitera**

Osnovne specifikacije Kinecta v2 su prikazane u Tablici 1. [8]

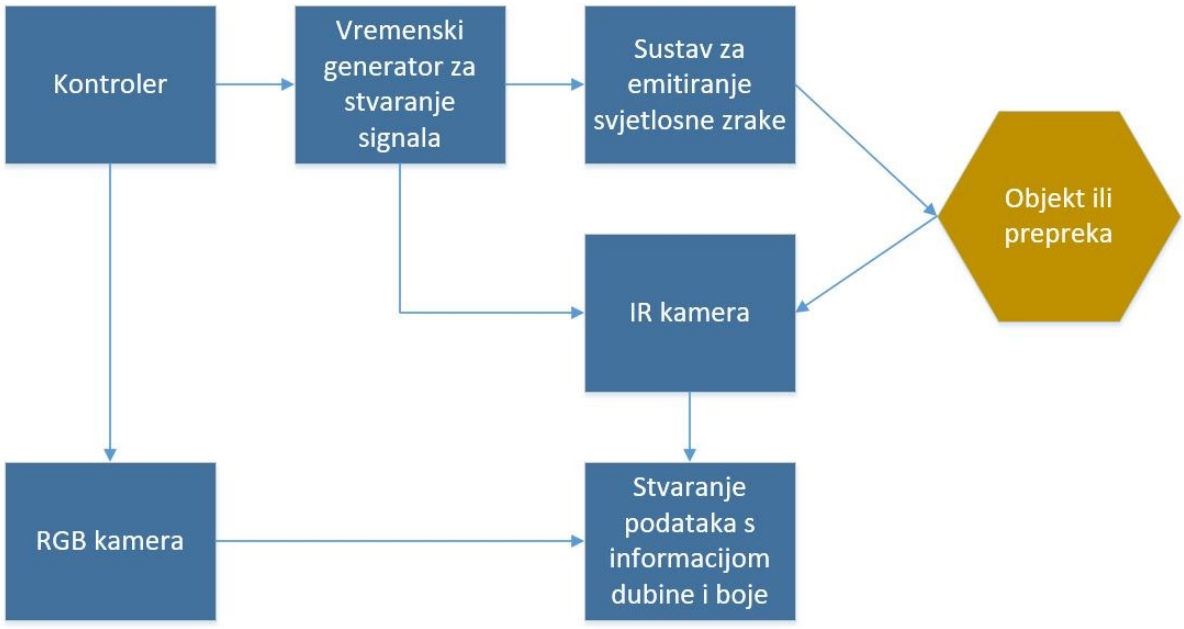
**Tablica 1. Osnovne specifikacije Kinecta v2**

| | |
|---|---|
| Rezolucija IR kamere | 512 x 424 px |
| Vidno polje IR kamere | 70,6° x 60° |
| Radni raspon | 0.5m – 4.5m |
| Rezolucija RGB kamere | 1920 x 1080 px |
| Vidno polje RGB kamere | 84,1° x 53,8° |
| Sličica u sekundi | 30 Hz |
| Dimenzije | 249 x 66 x 67 mm |
| Način spajanja | USB 3 |
| Radni napon | 12 V |
| Snaga trošenja | ~15 W |
| Cijena | ~ 200 USD |

### *2.4.1 Način rada Kinect v2 kamere*

Microsoft Kinect v2 senzor baziran je na ToF (*Time of Flight*) tehnologiji koja ima sposobnost izračuna udaljenosti svake točke iz kadra do senzora kamere. ToF tehnologija temeljena je na matematičkom modelu iz kojeg se udaljenost piksela, odnosno točke, određuje u vremenu $\Delta t$ potrebnom infracrvenoj zraki da se emitira iz senzora, odbije od nekog objekta ili prepreke u kadru, te vrati natrag u senzor kamere.

Infracrvena zraka iz IR odašiljača ispuštena je u obliku kvadratnog vala, a vrijeme potrebno odbijenoj zraki koja putuje natrag u senzor računa se preko faznog pomaka ispuštenog, odnosno primljenog vala, s pretpostavkom da je poznata i valna frekvencija. Za stvaranje takve vrste signala i njegovu interpretaciju zadužen je vremenski generator (eng. *Time generator*), dok ostatak sustava unutar senzora osigurava njegovo pravilno funkcioniranje, odnosno odašiljanje signala i u konačnici stvaranje izračunate točke s poznatim koordinatama. [9]




**Slika 2.12. Stvaranje 3D podatka Kinecta v2 koristeći ToF tehnologiju i senzor boje**

Svjetlost u zraku pređe jedan centimetar u jednoj pikosekundi koja preračunata u SI iznosi $10^{-12}$ sekundi što je dovoljno podataka za definiranje brzine koju svjetlost pređe u zraku, te uz izračunati fazni pomak i poznatu frekvenciju izračunava je konačna dubina točke $d$ preko izraza:

$$2d = \frac{\varphi}{2\pi}\frac{c}{f},$$

gdje su:

$d$ – dubina točke/piksela,

$\varphi$ – fazni pomak između odašiljanog i primljenog signala,

$f$ – frekvencija izlaznog i ulaznog signala,

$c$ – brzina svjetlosti u zraku.

# 3. PROBLEM PREPOZNAVANJA I PROSTORNA LOKALIZACIJA OBJEKATA

U ovom poglavlju razmotrit će se teoretski pristup rješavanju problema prepoznavanja i prostorne lokalizacije objekta na sceni. Pristup rješavanju nije egzaktan, tj. postoji više načina i niz značajki koje definiraju konačan rezultat. Kako bi se što kvalitetnije pristupilo problemu u sklopu rada razvijena su dva pristupa prepoznavanja u stvarnom vremenu (eng. *Real time*), što znači da se algoritam i dohvaćanje scene uzastopno ponavljaju dok korisnik ne odluči samostalno zaustaviti aplikaciju.

Prvi, jednostavniji pristup temelji se na definiranju klastera koji ujedno postaju volumeni od interesa, dok drugi, složeniji pristup prikuplja lokalne informacije oko ključnih točaka u sceni, odnosno ključnih točaka na modelu i na principu numeričkih vrijednosti te usporedbe pronalazi objekt na sceni. Nakon teoretske analize donijeti će se zaključat o prednostima i nedostacima za svaki pristup.

O eksperimentalnom postavu, implementaciji teoretske pozadine u algoritam pomoću Point Cloud Library-a i o dobivenim rezultatima pisati će se u sljedećim poglavljima.

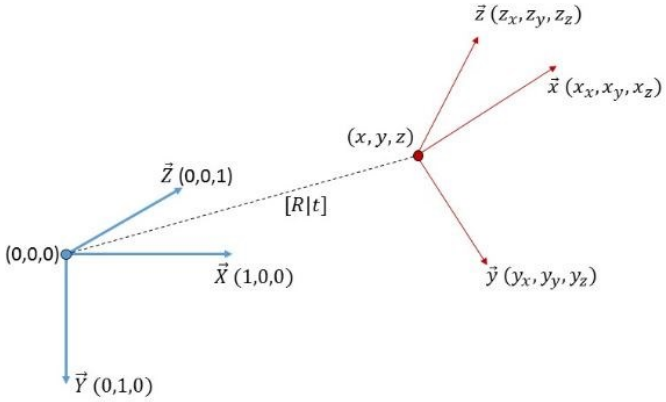
## 3.1 Teoretska podloga

Bit će spomenuti osnovni pojmovi kako bi se olakšalo razumijevanje idejnog cilja, tj. problematike vezane uz zadatak.

### *3.1.1 Matrica homogene transformacije*

Prostorna lokalizacija objekta obuhvaća problem definiranja položaja objekta na sceni što podrazumijeva određivanje njegove translacije i orijentacije u prostoru u odnosnu na fiksan koordinatni sustav. Matrica homogenih transformacija sadrži informacije o orijentaciji i translaciji predmeta, kvadratnog je oblika, veličine 3x3 za 2D problem i 4x4 za 3D. Daljnje će se razmatrati 4x4 matrica pošto se u zadatku upravlja 3D podacima.

Primjerice, ukoliko se na sceni nalazi objekt čiji je koordinatni sustav razmaknut i različito orijentiran od nekog referentnog koordinatnog sustava, potrebno je pronaći odgovarajuću matricu transformacije koja sadrži informacije o odnosu između ta dva koordinatna sustava.



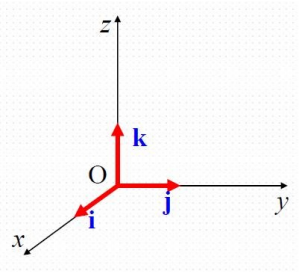

**Slika 3.1. Problematika određivanja matrice tranfsormacije između dva koordinatna sustava**

Matrica homogene transformacije za opisivanje koordinatnog sustava na slici 15. sljedećeg je oblika:

$$A = \begin{bmatrix} i_x & j_x & k_x & p_x \\ i_y & j_y & k_y & p_y \\ i_z & j_z & k_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

i može se raščlaniti na dva dijela – matricu rotacije i vektor translacije:

$$R = \begin{bmatrix} i_x & j_x & k_x \\ i_y & j_y & k_y \\ i_z & j_z & k_z \end{bmatrix}, \qquad T = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}.$$



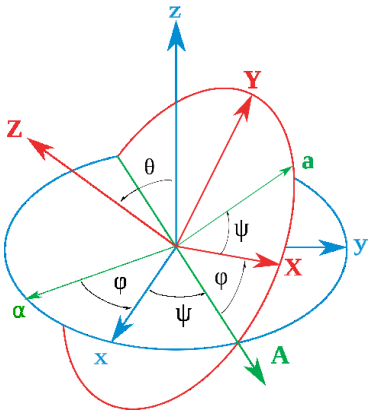

**Slika 3.2. Desnokretni pravokutni koordinatni sustav [10]**

Zadnji red matrice *A* prikazuje standardno proširenje i ne sadrži informacije o translaciji ili orijentaciji objekta. [10]

### *3.1.2 Eulerovi kutovi*

Orijentacija jednog u odnosu na drugi koordinatni sustav može se iskazati trima kutovima, $\emptyset, \theta\ i\ \psi$ ili $\alpha, \beta\ i\ \gamma$, koji odgovaraju trima učinjenim rotacijama.

Matrica rotacije *R* može se zapisati poznajući sva tri Eulerova kuta $\alpha, \beta\ i\ \gamma$ ili $\emptyset, \theta\ i\ \psi$ koji se u literaturi redom nazivaju skretanje, nagib i valjanje. Postoji 12 različitih konvencija koje sugeriraju o redoslijedu izvedenih rotacija i na koje treba obratiti pozornost. Postojeće konvencije su: *z-x-z, x-y-x, y-z-y, z-y-z, x-z-x, y-x-y, x-y-z, y-z-x, z-x-y, x-t-y, z-y-x, y-x-z.*




**Slika 3.3. Eulerovi kutovi**

Tako primjerice prema konvenciji *x-y-z* matrica rotacije *R* je komponirana od množenja triju matrica rotacija $R_x$, $R_y$ i $R_z$, koje su standardnog oblika ovisnog o konvenciji, te prikazuju rotaciju oko navedenih osi.

$$R_x = Rot(x, \emptyset, ) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\emptyset) & -\sin(\emptyset) \\ 0 & \sin(\emptyset) & \cos(\emptyset) \end{bmatrix}$$

$$R_y = Rot(y, \theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$R_x = Rot(z, \psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

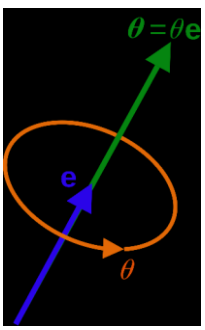
$$R = \begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\varphi)\sin(\psi) + \sin(\varphi)\sin(\theta)\cos(\psi) & \sin(\varphi)\sin(\psi) - \cos(\varphi)\sin(\theta)\cos(\psi) \\ -\cos(\theta)\sin(\psi) & \cos(\varphi)\cos(\psi) - \sin(\varphi)\sin(\theta)\sin(\psi) & \sin(\varphi)cos(\psi) + \cos(\varphi)\sin(\theta)\sin(\psi) \\ \sin(\theta) & -\sin(\varphi)\cos(\theta) & \cos(\varphi)\cos(\theta) \end{bmatrix}$$

Problem može biti zadan i inverzno gdje je potrebno iz izračunate matrice rotacije R odrediti Eulerove kutove, pogodne za analiziranje i kontrolu orijentacije zbog jednostavnog zapisa sačinjenog od tri člana.

### *3.1.3 Axis – angle zapis*

Pored eulerovih kutova, matrica transformacije i kvaterniona postoji i axis – angle zapis za opisivanje rotacije objekata u 3D euklidskom prostoru. Rotacija se prikazuje putem trodimenzionalnog vektora (*axis – angle vektora)* $\boldsymbol{\theta}$ koji nastaje kao skalarni umnožak kuta $\theta$ koji definira iznos rotacije i jediničnog vektora *e* koji ukazuje na smjer vrtnje.

$$\boldsymbol{\theta} = \theta e = [R_x \; R_y \; R_z]$$




**Slika 3.4. Axis angle vektor**

Radi zahtjeva UR5 robotske ruke potrebna je konverzija matrice rotacije u axis – angle što se može učiniti sljedećim izrazima:

$$\theta = \arccos\left(\frac{1}{2}[R_{11} + R_{22} + R_{33} - 1]\right)$$

$$e_x = \frac{R_{32} - R_{23}}{2sin\theta}$$

$$e_y = \frac{R_{13} - R_{31}}{2sin\theta}$$

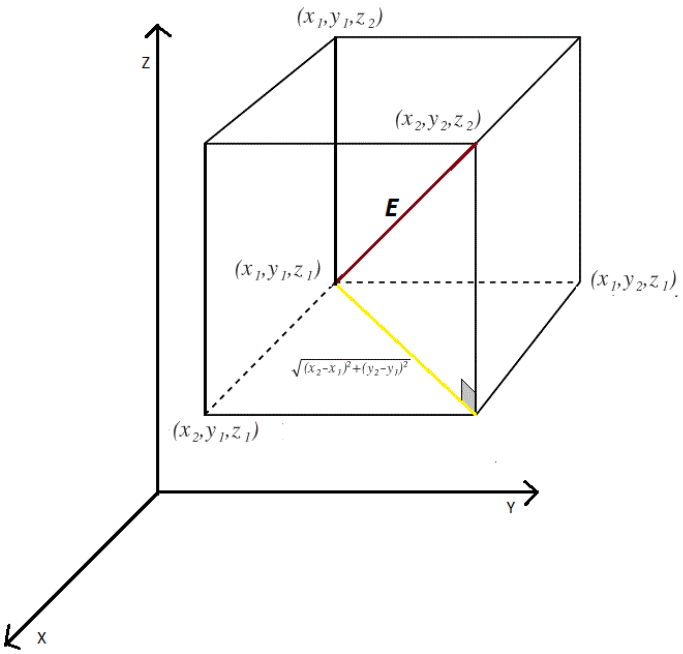
$$e_z = \frac{R_{21} - R_{12}}{2sin\theta}$$

### *3.1.4 Euklidska udaljenost*

Euklidska udaljenost opisuje najkraći razmak između dvije točke u prostoru. Najkraći razmak točke $p_1(x_1, y_1, z_1)$ u odnosu na točku $p_2(x_2, y_2, z_2)$ može se izraziti koristeći Pitagorin poučak:

$$E = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2},$$

gdje je $E$ euklidska udaljenost.

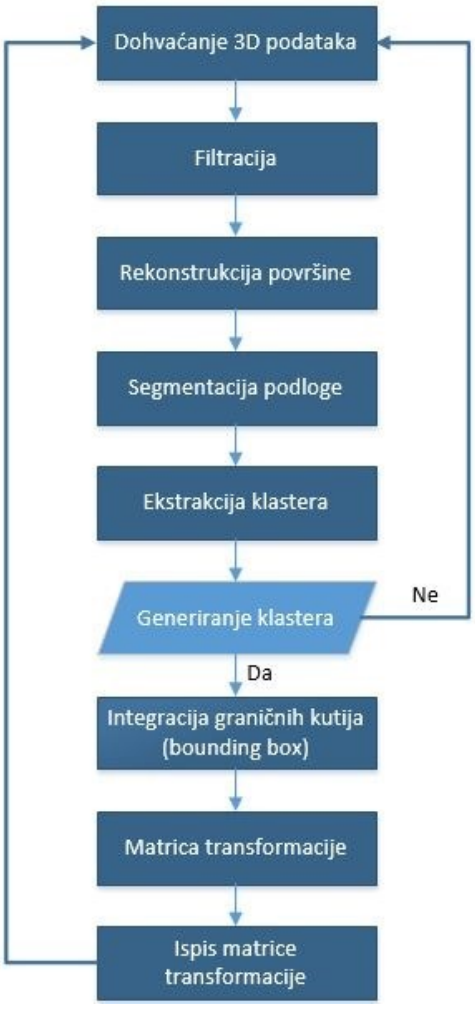



**Slika 3.5. Euklidska udaljenost između dvije točke**

## 3.2 Metoda ekstrakcije klastera

Iako metoda ekstrakcije klastera ne sadrži algoritme prepoznavanja i registracije, integrirane unutar PCL-a, može poslužiti kao pokazni primjer da i dalje postoji mogućnost izolirati objekt sa scene te prostorno ga lokalizirati, a problem prepoznavanja može se promatrati neegzaktno gdje različiti pristupi vode do rješenja.

Klasterizacija je postupak u kojem se oblak točaka $P\{X, Y, Z\}$ dijeli u manje, smislene dijelove, čime svaki objekt na sceni postaje samostalna cjelina točaka $P_i\{X, Y, Z\}$ i stvara se mogućnost lokalizacije jednog objekta, a ne cijele skupine.



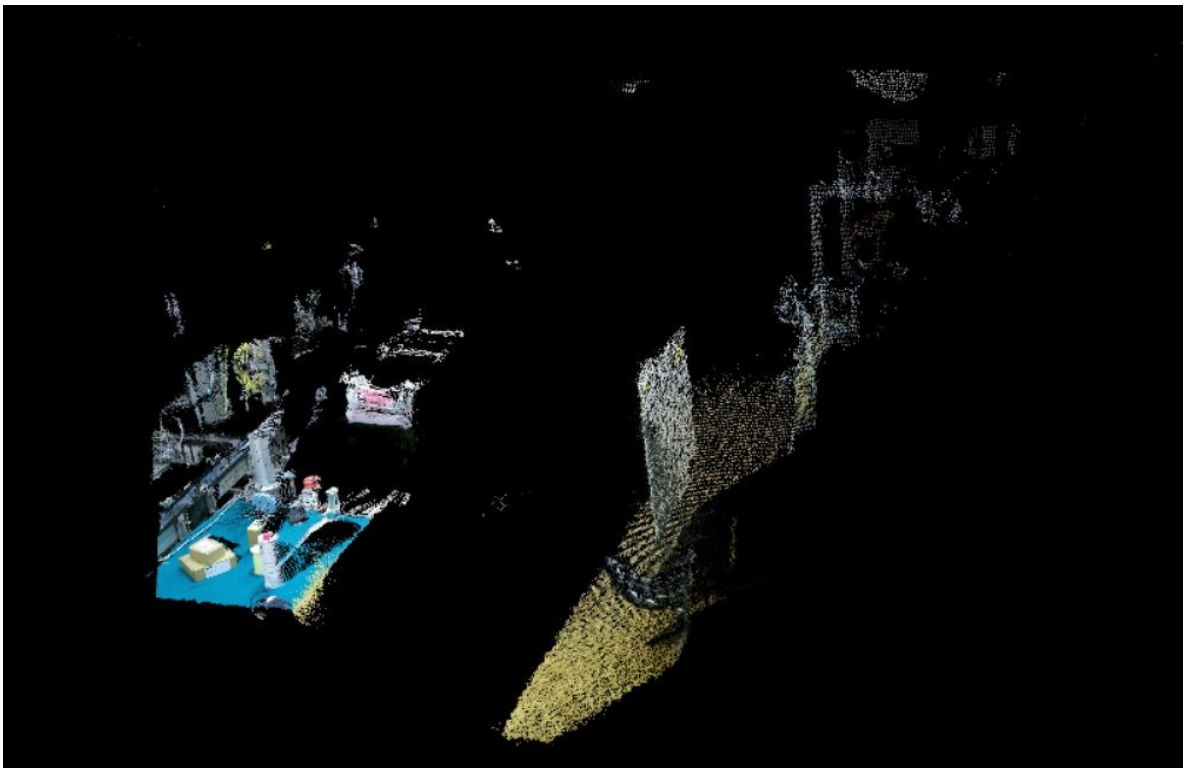

**Slika 3.6. Dijagram toka – metoda ekstrakcija klastera**

U sljedećim poglavljima detaljnije će se razraditi pojedini koraci navedeni u dijagramu toka radi boljeg razumijevanja njihove uloge i načina rada.

### *3.2.1 Filtracija*

Nakon dohvaćanja podataka iz 3D kamere nužan je korak njihovo pred-procesiranje, tj. filtracije u svrhu poboljšanja pojedinih značajki i uklanjanja nepotrebnih informacija kako bi daljnji algoritmi radili brže i efikasnije. Pošto Kinect ne pripada u skupinu preciznih senzora često su prisutne minimalne razlike i rasipanje podataka na konturama manjih objekta, a ideja je da se konačan program izvodi robusno bez prevelikih odstupanja između sličica kamere (eng. *frames*).

Svrha uklanjanja nepotrebnih informacija posebno je bitna i dolazi do izražaja za algoritme koji se odvijaju u stvarnom vremenu jer se značajno skrati vrijeme izvođenja cjelokupnog programa. Također, mnoge funkcije unutar PCL-a djeluju na način da analiziraju svaku točku unutar oblaka čime se povećava važnost pravilne filtracije.



**Slika 3.7. Scena prije primjenjenih filtracija**

#### *3.2.1.1 Passthrough Filter*

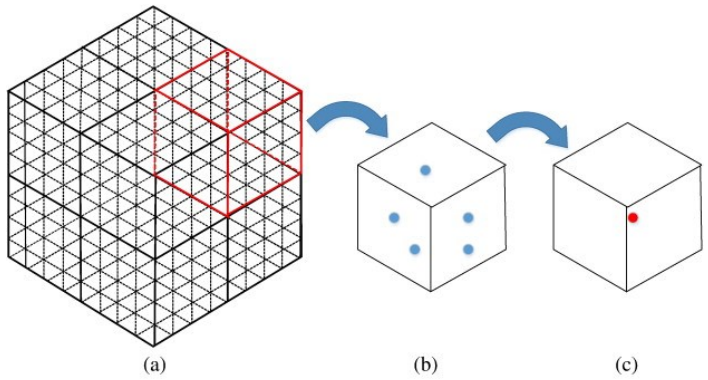
Passthrough filter uklanja dimenziju podataka duž specificirane osi na temelju poznatih koordinata svake točke, odnosno uklanja točke koje se nalaze izvan vrijednosti zadane od korisnika. Interval vrijednosti se zadaje za svaku koordinatnu os posebice, a ovisi o eksperimentalnom postavu, odnosno o položaju kamere naspram scene.

Kako bi se dobila robusnost podataka nužno je učvrstiti kameru jer svakim većim pomakom potrebno je iznova definirati vrijednosti u intervalu. Konačna ideja passthrough filtera je značajno smanjiti broj nepotrebnih podataka u oblaku točaka.

### *3.2.1.2 Voxel Filter*

Reduciranje ukupnog broja točaka u setu podataka predstavlja važnu, ali ne u potpunosti nužnu funkciju za daljnje izvođenje algoritama. Većina operacija unutar Point Cloud Library-a obrađuje svaku točku unutar oblaka stoga će manji broj točaka drastično skratiti daljnje ukupno vrijeme potrebno za obradu.

Voxel filter početni set podataka aproksimira 3D voxelima, odnosno 3D kutijama, u kojoj se nalazi određeni broj točaka. Sve točke prisutne unutar kutije aproksimiraju se sa centroidom – točkom koja opisuje središte ili ravnotežu svih točaka koje su se nalazile unutar kutije.




**Slika 3.8. Voxel filter u koracima, a) oblak točaka se aproksimira graničnim kutijama, b) u graničnoj kutiji se nalazi određeni broj točaka, c) aproksimira se centroid granične kutije dok se ostale uklanjaju**

Koliko će se točaka nalaziti unutar voxela ovisi o vrijednosti zadanoj od korisnika koji definira željenu euklidsku udaljenost između centroida susjednih kutija. Voxel filter osim što se koristi za reduciranje broja točaka može se koristiti i za uspostavljanje uniformne gustoće točaka ukoliko su neki algoritmi to poremetili.

Pseudo kod za voxel filter:

1. Ispuniti ulazni podatak s 3D kutijama (voxelima)
2. U svakoj kutiji izračunati centroid za točke koje se nalaze u njoj
3. Nakon što se izračuna centroid ukloniti preostale točke

Uporaba Voxel filtera svakako može ubrzati daljnje algoritme radi analize manje količine podataka, ali isto tako narušiti točnost registracije jer će objekti biti opisani s manje točaka.

#### *3.2.1.3 Statistical Outlier Removal Filter*

Unatoč greški senzora moguća je pojava nasumično raspoređenih točaka različite gustoće oko rubova objekata. Takve točke ne nose korisne informacije koje bi opisale predmet, dapače, mogu drastično narušiti kvalitetu algoritama zaduženih za računanje lokalnih karakteristika oko skupine točaka, poput normala na površinu.

Nepravilnosti poput šumova mogu se ukloniti uporabom statističke analize susjedstva oko svake točke, na način da one točke koje nisu unutar određenih kriterija budu eliminirane iz skupa. Za svaku točku računa se udaljenost između svih susjeda, a na temelju udaljenosti očekivanje, odnosno prosjek udaljenosti i standardna devijacija. U konačnici računa se granična vrijednost koja se uspoređuje s prosječnom vrijednosti udaljenosti točke od susjeda. [12]

Pseudo kod za statistical outlier removal filter:

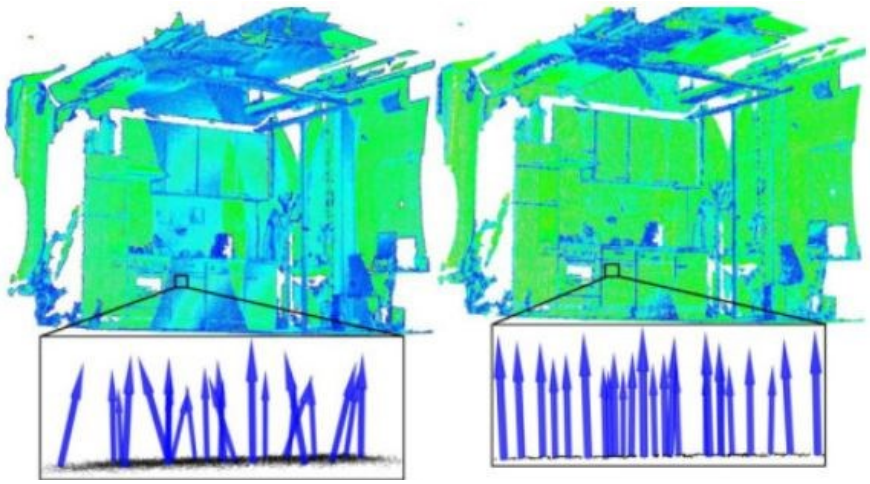
1. Postavljanje k vrijednosti za određivanje susjeda oko svake točke
2. Postavljanje $\alpha$ multiplikatora standardne devijacije
3. Za svaku točku $P_i$ pronaći k najbližih susjeda
4. Izračunati udaljenost $d$ između točke $P_i$ i najbližih susjeda
5. Izračunati prosječnu udaljenost $\mu_d$ između točke $P_i$ i najbližih susjeda
6. Izračunati standardnu devijaciju $\sigma_d$ za udaljenosti $d_i$
7. Izračunavanje dopuštene granice $T_g = \mu_d + \alpha \cdot \sigma_d$
8. Eliminacija točki koje ispunjavaju uvjet $d > T$.

Algoritam zahtjeva dva prolaska kroz cijeli set podataka, u prvom prolazu određuje koje točke ispunjavaju zadani kriterij, da bi na kraju iste bile uklonjene.

### *3.2.2 Rekonstrukcija površine*

Nepravilnosti na površinama uzrokovane manjom greškom senzora teško je ukloniti statističkim analizama, stoga je potrebno koristiti neki od algoritama iz PCL modula za rekonstrukciju površine. Iz prisutnih šumova oko površina i neravnih ploha mogu proizaći pogrešne lokalne karakteristike koje umanjuju kvalitetu prepoznavanja, što posebice dolazi do izražaja u složenijoj metodi prepoznavanja i lokalizacije koja će biti opisana kasnije.

Rješenje problema proizlazi iz traženja odgovarajućeg polinoma višeg stupnja, analizom okolnih točaka u podacima, nakon čega slijedi kreacija točaka koje nedostaju prateći izračunatu funkciju. Tako površina postaje pogodnija za uporabu nadolazećih funkcija kao što je primjerice izračun normala (Slika 3.9).



**Slika 3.9. Položaj normala na površini prije i nakon korištenja rekonstrukcije površine**

Također je potrebno pripaziti da definirani radijus filtera ne bude premali, odnosno preveliki, jer može doći do estimacije pogrešnog polinoma, a površina izgubiti svoj prvobitni, karakteristični, oblik.

### *3.2.3 Uklanjanje podloge*

Postoji više razloga zašto je poželjno ukloniti podlogu iz seta podataka. Jedan od razloga je što podloga ponekad zauzima popriličan udio u ukupnom broju točaka što može usporiti naredne algoritme. U takvoj situaciji nužno je osigurati uvjet da se funkcija uklanjanja odvije u manje vremena nego što bi se odvio ostatak algoritma bez uklonjene podloge, stoga željena filtracija ne bi došla u pitanje zbog manjka efikasnosti.

Međutim, uklanjanje podloge je često neophodna ili vrlo korisna radnja za pojedine funkcije. Tako primjerice ne bi bilo moguće stvoriti klastere bez njenog odstranjivanja, dok kod izračuna lokalnih značajki i registracije značajno se pospješuje konačan rezultat u smislu točnosti pozicioniranja, postotku prepoznavanja i brzine izvođenja. U okviru rada korištena su dva načina odstranjivanja podloge, putem RANSAC algoritma i RGB filtera.

#### *3.2.3.1 RANSAC segmentacija*

Segmentacija je proces u kojem se 2D ili 3D podaci raščlanjuju na segmente, odnosno više dijelova, u svrhu kvalitetnije analize objekata od interesa unutar podataka. Ideja je iz oblaka segmentirati podlogu te je zatim ukloniti od ostatka točaka.

RANSAC (Random Sample Consensus) je iterativna metoda/algoritam koja ima mogućnost detektirati parametre koji pripadaju određenom matematičkom modelu kao što su primjerice ravnine, cilindri, sfere, kružni presjeci, paralelne površine ili linije i slično. Prilikom djelovanja RANSAC algoritma podaci se dijele na tipične (eng. *inlier*) i netipične (eng. *outlier*) vrijednosti, tako podaci, tj. točke, koje zadovoljavaju zadani matematički model bit će proglašene kao *inliers* dok ostatak se odbacuje i definira kao *outliers.*

Pseudo kod za RANSAC:

1. Iz ulaznog oblaka točaka nasumično se odabiru točke iz kojih se postavlja matematički model koji određuje kriterij po kojima će ostale točke biti proglašene kao *inliers*, odnosno *outliers*.
2. Nasumične točke u *n* iteracija uspoređuju se sa postavljenim matematičkim modelom, ukoliko se uklapaju postavlja se hipoteza da pripadaju *inlersima*.
3. Procijenjeni model se klasificira kao dobar ukoliko je popriličan broj točaka iz ulaznog oblaka definiran kao *inliers.*
4. Zatim se iz svih *inliersa* ponovno računa osnovni model jer je inicijalni model postavljen od samo inicijalnih točaka

5. U konačnici točnost modela je procijenjena na temelju pogreške *inliersa* u odnosu na izračunati osnovni model.

Prednost RANSAC algoritma je robusnost, što implicira da će u svakoj sličici kamere vrlo vjerojatno rezultat bit zadovoljavajući. Negativno je što je potrebno dobro postaviti parametre koji ovise o broju objekata na sceni, odnosno o udjelu broja točaka podloge u usporedbi s cijelim setom podataka. [13]

#### *3.2.3.2 RGB filtracija*

Filtracija boje je jedan od primjera gdje prisustvo RGB senzora u Kinectu v2 dolazi do izražaja, na robustan i efektivan način mogu se odstraniti točke određene boje definiranjem ispravnog RGB intervala

Međutim, potrebno je pripaziti da objekti od interesa ne budu jednake boje kao i podloga jer će doći do filtracije istih. Preporuka je da podloga bude crne ili bijele boje jer se isključuju samo točke uskog RGB intervala što ostavlja veći RGB prostor za objekte od interesa.

U sklopu rada odlučeno je da podloga bude odstranjena RGB filterom radi bržeg procesiranja, stabilnog djelovanja u svakoj sličici, te bez utjecaja o količini predmeta na podlozi.

### *3.2.4 Ekstrakcija klastera*

Nakon što su načinjene potrebne filtracije potrebno je od preostalih točaka na sceni, koje predstavljaju objekte, izraditi klastere, tj. zasebne cjeline kako bi se analizirao jedan predmet, a ne cijeli skup preostalih i neorganiziranih točaka. Klasterizaciju je moguće napraviti korištenjem klase euklidske ekstrakcije u kombinaciji s kd-stablom za traženje najbližeg susjeda.

Radi lakšeg razumijevanja spomenute metode sastavit će se pseudo kod:

1. Generiranje kd-stabla za ulazni oblak točaka *P*.
2. Sastavljanje prazne liste klastera *C* te definiranje seta podataka *Q* u kojem se nalaze točke koje nisu provjerene unutar kd-stabla.
3. Za sve točke $p_i$ kao elemente oblaka *P* učiniti sljedeće korake:
    - pridodati red točaka $p_i$ u skup *Q*
    - zatim za sve točke $p_i$ unutar skupa *Q:*
        - ♦ pretražiti i sastaviti set $P_k^i$ susjednih točaka od točaka $p_i$ u sferi s definiranim radijusom pretraživanja $r < d_{th}$
        - ♦ za susjedne točke $p_k^i$ unutar seta $P_k^i$ provjeriti da li su sve točke procesuirane, ako nisu dodati ih u skup *Q*
    - nakon što je cijela lista *Q* obrađena, dodati *Q* u listu klastera *C, Q* resetirati kako bi bio prazan
4. Algoritam završava kada su obrađene sve točke $p_i$ kao elementi oblaka *P* i kategorizirane u klastere *C*. [14]

Dva skupa točaka $P_1^i$ i $P_2^j$ su različita ukoliko je ispunjen uvjet minimalne udaljenosti između krajnje točke iz svakog skupa:

$$\min \left|\left|p_i - p_j\right|\right| \geq d_{th},$$

gdje je $d_{th}$ maksimalna/dopuštena granica udaljenosti.

Ukoliko nije pronađen niti jedan klaster, odnosno objekt na sceni, program se vraća na početak gdje dohvaća scenu, obrađuje je opisanim filtracijama, rekonstrukcijom površine, odstranjuje podlogu, te u konačnici opet dolazi do navedenog koraka.

### *3.2.5 Granične kutije*

Izdvojeni klasteri, ukoliko su pronađeni na sceni, predstavljaju objekte koji su prikazani u 3D prostoru stotinama ili tisućama točkama. Kako bi sve točke pojedinog objekta u konačnici bile povezane i zajedno sačinjavale veću cjelinu integrirat će se ideja graničnih kutija (*engl. bounding box).*

Granične kutije sačinjavaju najmanji zatvoreni volumen, odnosno najmanje gabaritne mjere, u kojem su sadržane sve točke koje sačinjavaju jedan klaster, a osnovna zadaća im je odrediti prostornu transformaciju predmeta na sceni koja uključuje translaciju i orijentaciju u odnosnu na referentan koordinatni sustav.

Izuzev prostorne transformacije, granične kutije omogućuju jednostavno definiranje mjesta prihvata objekta zbog poznatih minimalnih i maksimalnih vrijednosti točaka od centroida, tj. sjecišta glavnih dijagonala, za svaku os u kartezijevom koordinatnom sustavu. Također za predmet omeđen graničnim kutijama jednostavnim algebarskim izrazima moguće je redom odrediti duljinu, širinu i visinu:

$$d_{gk} = x.\max - x.min$$
$$b_{gk} = y.\max - y.min$$
$$h_{gk} = z.\max - z.min.$$

Veza između minimalnih i maksimalnih vrijednosti i centroida granične kutije može se prikazati poznatom točkom centroida ($\frac{x.\max - x.min}{2}$, $\frac{y.\max - y.min}{2}$, $\frac{z.\max - z.min}{2}$).

### *3.2.6 Prednosti i nedostaci metode ekstrakcije klastera*

Kao što je već napomenuto u metodi ekstrakcije klastera nisu upotrebljavani PCL algoritmi prepoznavanja i registracije, međutim, pravilnim korištenjem funkcija iz ostalih modula dospjelo se do izolacije objekta sa scene i njegove prostorne lokalizacije opisane matricom transformacije.

Algoritam ne podržava usporedbu modela sa scenom, što dovodi do nemogućnosti referenciranja na određeni objekt, što je ujedno najveće ograničenje i nedostatak. Primjerice, ukoliko se na podlozi nalaze tri različita predmeta odabir se vrši na principu najbolje odabranog klastera, odnosno ne postoji mogućnost vlastitog odabira određenog objekta. Ukoliko se dva objekta na sceni dodiruju ili su suviše blizu (~ 1 cm), bit će proglašeni kao jedan klaster, što također narušava funkcionalnost programa, no problem se donekle može regulirati maksimalnim, odnosno minimalnim brojem točaka koji klaster može sadržavati.

Da bi metoda ekstrakcije klastera pravilno funkcionirala potrebno je osigurati da u vidno područje ulaze samo objekti koje se želi izuzeti, i to u razmacima većim od ~1cm, kako se ne bi dogodio iznad navedeni problem. Ukoliko su postignuti poželjni uvjeti metoda ekstrakcije klastera predstavlja robusno rješenje s visokom ponovljivošću i relativno dobrom preciznošću.

Granične kutije, osim što ispostavljaju informacije o prostornoj orijentaciji, donose i dodatne funkcionalnosti kao što su izračun dimenzije objekata i mogućnost točnog definiranja mjesta prihvata.

Prednosti:

- Brzina rada programa
- Podešavanje malog broja parametara
- Visoka ponovljivost
- Definiranje mjesta prihvata pomoću graničnih kutija
- Izračun dimenzije objekata kao dodatna funkcija

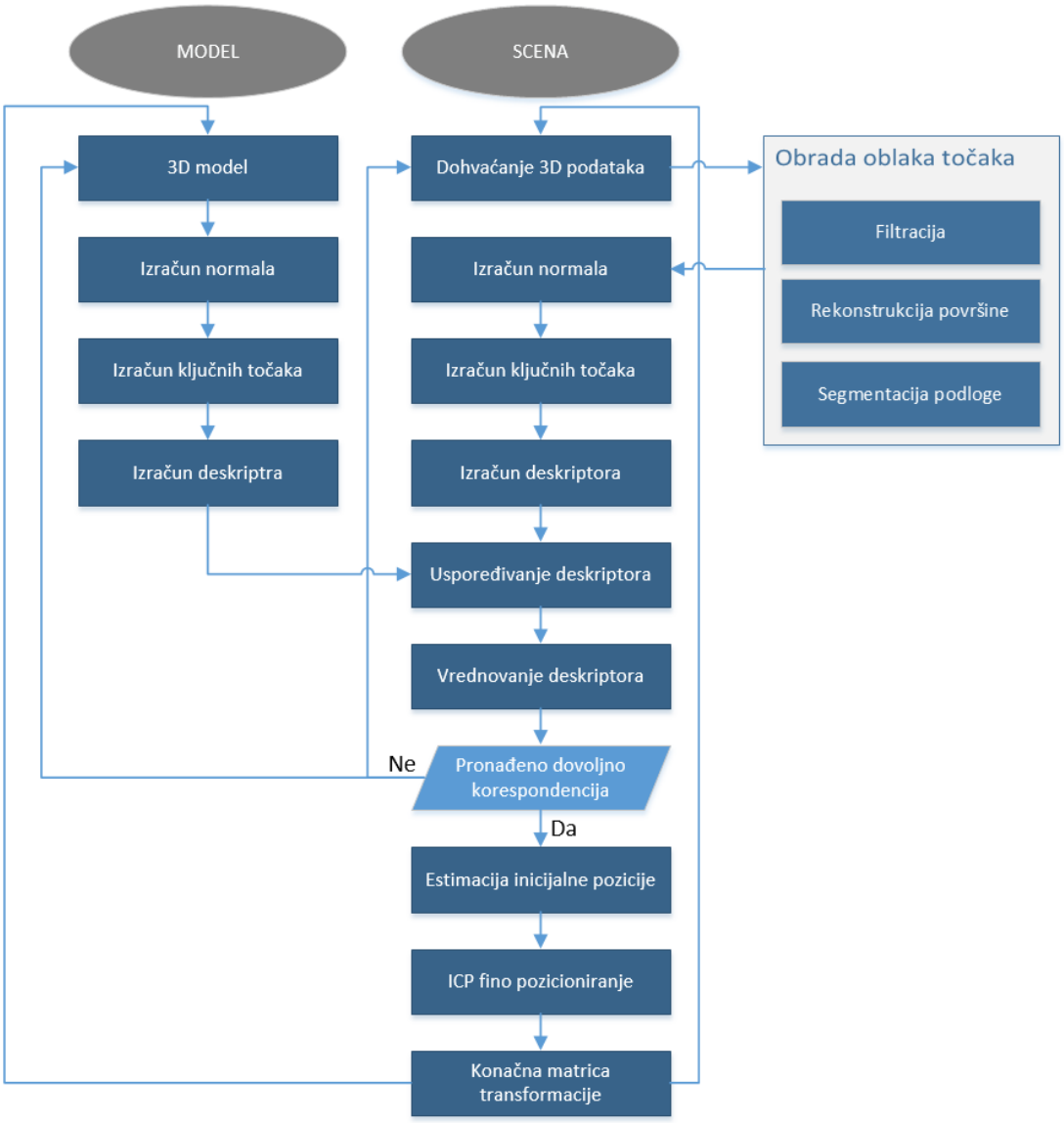
Nedostaci:

- Nije moguće definirati objekt koji se želi izuzimati
- Ukoliko se objekti dodiruju ili su suviše blizu definirani su kao jedan klaster
- Nužna segmentacija, odnosno odstranjivanje podloge

Uslijed nedostataka koji uvelike ograničavaju funkcionalnost i smisao izuzimanja željenog predmeta u skupu, razvit će se metoda koja prikuplja lokalne informacije o sceni, tj. objektima, te uključuje algoritme prepoznavanja i registracije.

## 3.3 Metoda lokalnih značajki

Potrebno je pronaći složenije rješenje koje će biti robusno te konzistentno, a ujedno može eliminirati ograničenja, odnosno nedostatke, prethodne metode. Problemu će se pristupiti prikupljajući korisne informacije o lokalnoj geometriji objekata na sceni, a zatim na principu usporedbe modela i scene detektirati željeni predmet i odrediti njegovu prostornu poziciju i orijentaciju. Metoda lokalnih značajki zahtjeva filtraciju, izračun ključnih točaka, normala na površinu, deskriptora, ekstrakciju predmeta sa scene koji će predstavljati model te algoritme za usporedbu i vrednovanje deskriptora. Problematika se sastoji u pravilnom izboru između navedenih zahtjeva te pravilnom postavljanju niza parametara. Jednu od važnijih uloga u metodi lokalnih značajki imaju deskriptori koji računaju i pohranjuju informacije o geometriji objekta oko ključne točke. Na temelju izračunatih informacija vrši se usporedba modela i scene, te u konačnici određuju korespondencije. Korespondencije označavaju iste, tj. vrlo slične, točke između modela i scene.




**Slika 3.10. Dijagram algoritma za metodu lokalnih značajki**

### *3.3.1 Obrada oblaka točaka*

Filtracija, rekonstrukcija površine i uklanjanje podloge, opisani u poglavljima 3.2.1 do 3.2.3, i dalje su prisutni i imaju važnu ulogu u metodi lokalnih značajki.

Pošto deskriptori, u metodi lokalnih značajki, prikupljaju informacije o geometriji, ideja je da scena ima što manje šumova i nepravilnosti kako bi prikupljene informacije bile vjerodostojne, tj. iste/slične u svakoj sličici (*engl. frame)* za istu scenu.

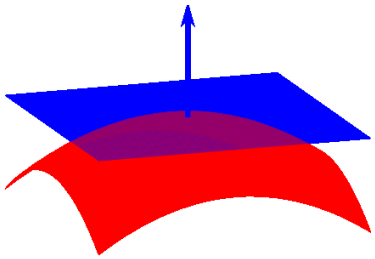
Osim djelovanja *Passthrough i Voxel* filtera posebno dolazi do izražaja *Statiscial Outlier Removal* filter jer uklanja zaostale točke nastale greškom senzora, odnosno točke koje nemaju ulogu opisivanja objekta na sceni, a mogu negativno utjecati na stabilnost, ponovljivost, smanjenju broja korespondencija i pogrešnu konačnu estimaciju pozicije.

Normale, kao jedne od osnovnih nosioca informacija o geometriji lokalnog područja, često mogu biti pogrešno orijentirane radi distorzije površine uslijed greške senzora. Kako bi se stvar ispravila koriste se algoritmi iz rekonstrukcije površine čije je djelovanje prikazano na slici 3.9.

Uklanjanje podloge nije nužna obrada kao kod metode ekstrakcije klastera, međutim, deskriptori su procesorski zahtjevan proces i prisutnost podloge osjetno može usporiti rad programa radi većeg broja izračunatih značajki. Izostanak podloge znači da je potreban manji izračun broja ključnih točaka, normala i deskriptora.

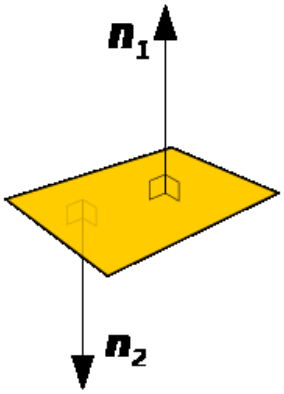
### *3.3.2 Normale*

U 3D prostoru normale su vektori okomiti na tangentnu ravninu sačinjenu od skupa točaka na lokalnom području, a služe za matematičko određivanje orijentacije površine koja omeđuje promatranu cjelinu, u ovom slučaju to su traženi objekti na sceni.



**Slika 3.11. Normala na površinu, crveno područe – površina objekta, plavo područje – tangnetna ravnina**

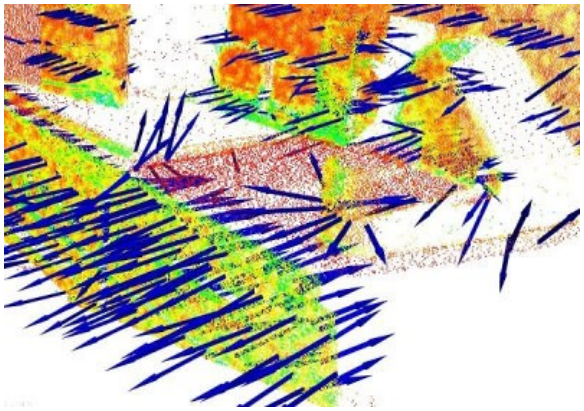
Problematika izračuna normala pojavljuje se jer postoji mogućnost dvostruke orijentacije normala na istu tangentnu ravninu kao što je prikazano na sljedećoj slici.




**Slika 3.12. Dvostruka orijentacija normala**

Za izračun normala unutar PCL-a koristi se analiza glavnih komponenata (*eng. Principal Component Analysis – PCA)*, metoda koja smanjuje dimenzionalnost, odnosno broj varijabli, u cilju preglednosti i pojednostavljenja velikog broja podataka.

Uslijed nemogućnosti nasumičnog matematičkog izračuna konstantne orijentacije normala na površinu pojavljuju se normale različite orijentacije prikazane na slici 3.13.

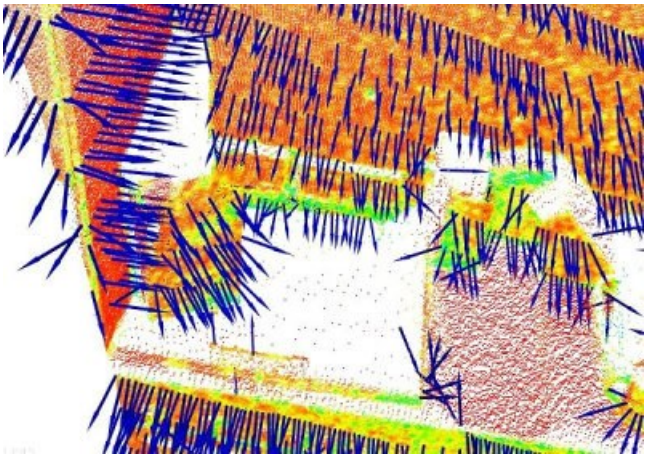


**Slika 3.13. Različita orijentacija normala**

Kako bi se problem ispravio potrebno je definirani točku gledišta (*engl. viewpoint*) prema kojoj će biti usmjerene normale kako bi se postignula ujednačena orijentacija. Na taj način zadovoljava se jednadžba:

$$\overrightarrow{n_i} \cdot (v_p - p_i) > 0,$$

gdje su $\overrightarrow{n_i}$ vektori normale, $v_p$ definirana točka gledišta i $p_i$ središnja točka unutar definiranog radijusa ili k susjednih točaka. [13]



**Slika 3.14. Ujednačena orijentacija normala**

Prema inicijalnim postavkama točka gledišta je definirana prema senzoru kamere što također ponekad može uzrokovati nekonzistentnu orijentaciju normala.

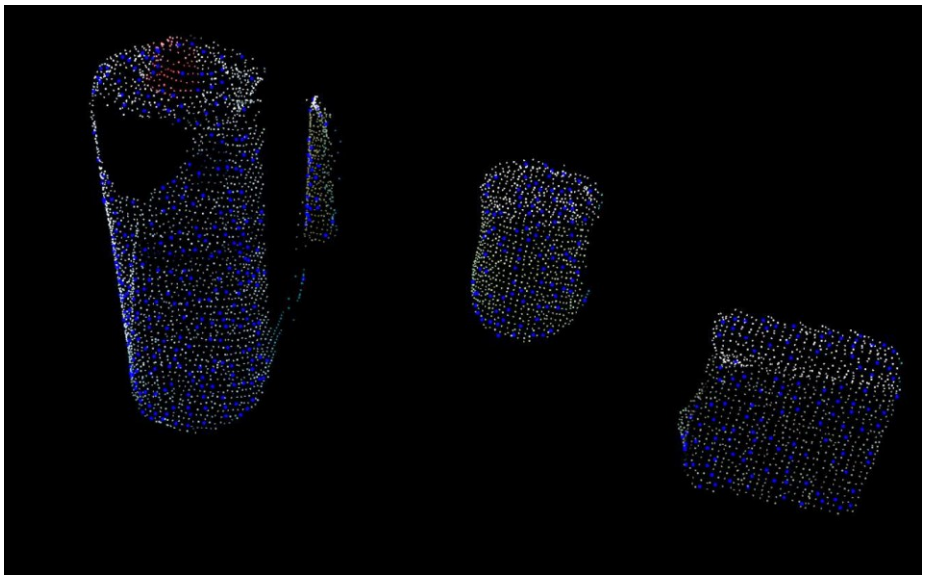
Pseudo kod za izračun normala:

1. Za sve točke unutar oblaka naći najbližeg susjeda pomoću definiranog radijusa
2. Izračunati normalu *n* za točku *p*
3. Orijentirati normale prema točki gledišta

### *3.3.3 Ključne točke*

Dohvaćena scena sa 3D senzorom nakon filtracije još uvijek može sadržavati nekoliko desetaka tisuća točaka, ovisno o broju objekata na sceni, što je poprilično puno kao ulazni parametar za deskriptore koji su procesorski zahtjevni za računanje, stoga je potrebno integrirati točke od interesa.

Ključne točke ili točke od interesa imaju svrhu prezentirati scenu/objekt u manjem broju točaka tako da se izračunavaju na mjestima gdje su prisutne određene lokalne karakteristike, primjerice promjena geometrije ili boje. Kao takve trebale bi ispunjavati svojstvo ponovljivosti, tj. da se bez obzira na translaciju ili rotaciju objekta nalaze na istoj geometriji u više iteracija programa. [14]



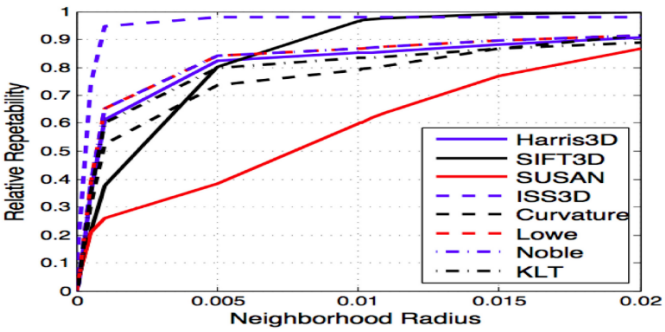
**Slika 3.15. Plavom bojom označene ključne točke na predmetima**

Unutar PCL-a prisutno je više detektora ključnih točaka poput Noble, Curvature, ISS3D, Harris 3D, SUSAN, Lowe, KLT, Voxel, SIFT, NARF i još nekolicine njih. Odabir je određen na temelju eksperimentalnih istraživanja napravljenih u literaturi [15] i [16] gdje je promatran postotak ponovljivosti ključnih točaka s obzirom na višestruke rotacije, odnosno translacije objekata, u odnosu na definiran radijus za klasifikaciju najbližih susjeda.

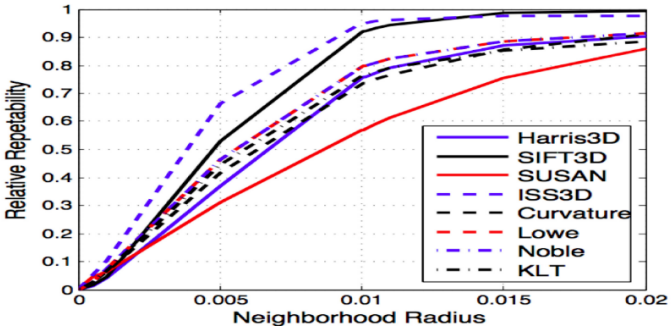
U literaturi [16] eksperiment je rađen na RGB – D setu podataka koji se sastoji od 300 različitih objekata podijeljenih u 51 kategoriju. Na promatranim predmetima rotacija i translacija nije napravljena pomicenjem kamere niti objekata na sceni već softverski, tj. definiranjem matrice transformacije.

**Slika 3.16. Primjer objekata promatranih u testu**

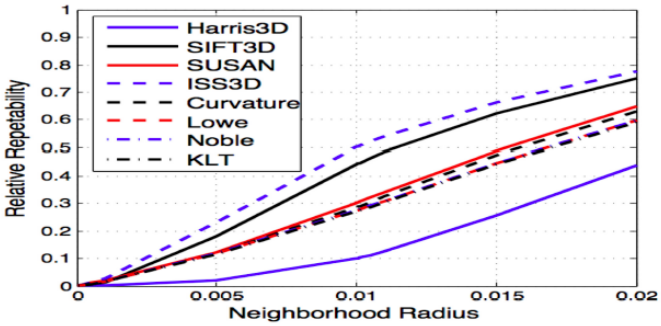
Rotacija je učinjena koracima od 10° počevši s rotacijom od 5° sve do 45°, dok je translacija napravljena pomakom u sve tri osi kartezijevog koordinatnog sustava.



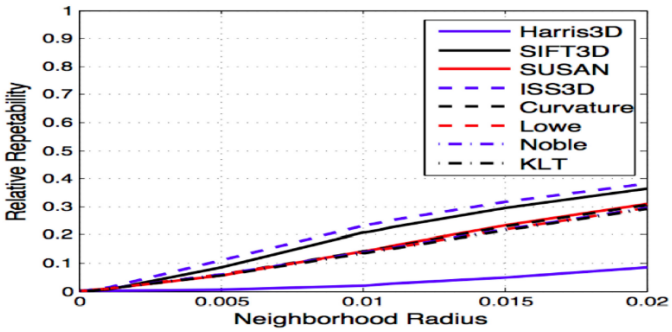

**Slika 3.17. Ponovljivost za rotaciju od 5°**



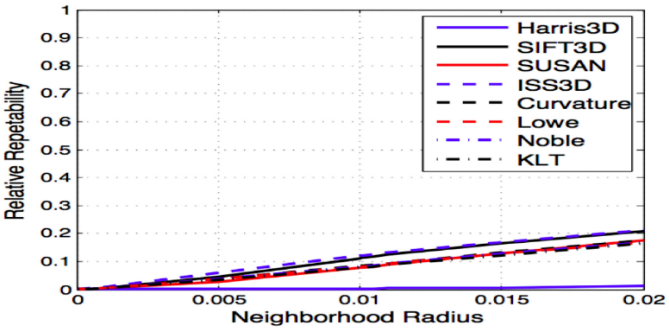

**Slika 3.18. Ponovljivost za rotaciju od 15°**



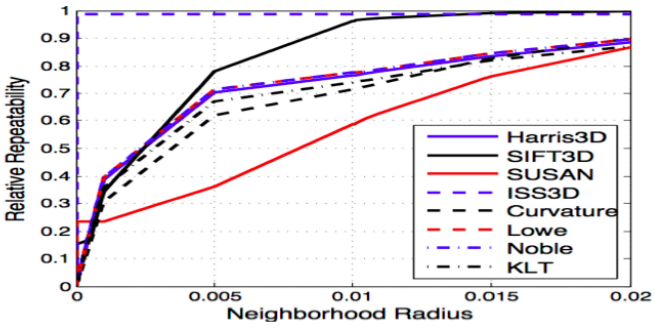

**Slika 3.19. Ponovljivost za rotaciju od 25°**




**Slika 3.20. Ponovljivost za rotaciju od 35°**




**Slika 3.21. Ponovljivost za rotaciju od 45°**

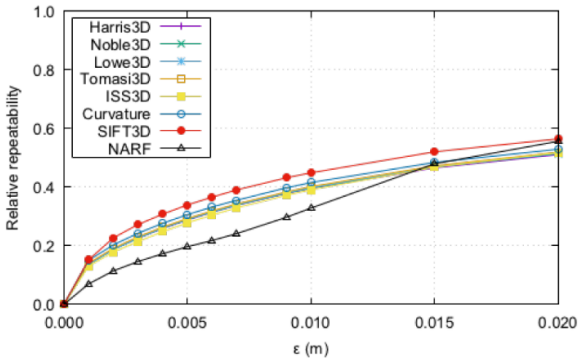



**Slika 3.22. Ponovljivost za translaciju**

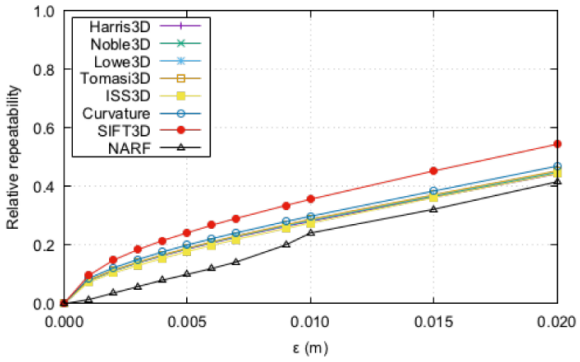
Iz priloženog može se uočiti da je najveću invarijantnost na rotaciju i translaciju pokazao ISS3D detektor, također se može spaziti trend da uslijed porasta rotacije pada ponovljivost kod svih ključnih točaka.

Postavljanjem većeg radijusa za definiranje najbližih susjeda povećava se i ponovljivost, međutim tada opada ukupan broj ključnih točaka što može uzrokovati da manji objekti na sceni ne budu zadovoljavajuće opisani.

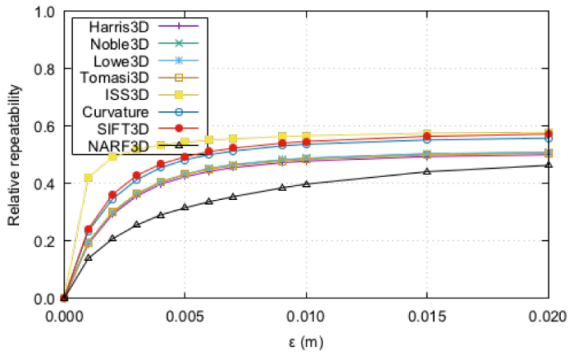
U literaturi [17] dohvaćen je set podataka pomoću SR-4k (SwissRangers 4000) kamere bazirane na ToF tehnologiji kao i Kinect v2. U ovome eksperimentu, za razliku od prošloga, nije napravljen test na translaciju, dok rotacija nije napravljena apliciranjem matrice transformacije već zakretanjem kamere oko objekta. Ponovljivost ključnih točaka mjerena je na nekoliko različitih objekata s obzirom na rotacije od 10 i 20 stupnjeva.



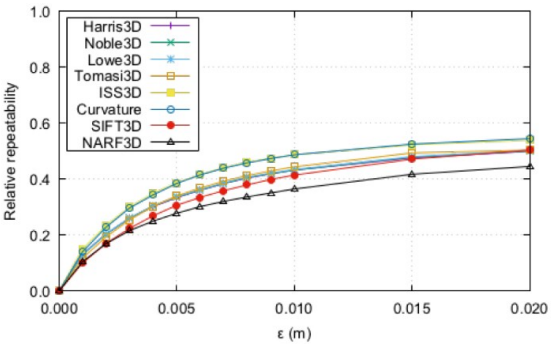

**Slika 3.23. Ponovljivost za rotaciju od 10° - tipkovnica**



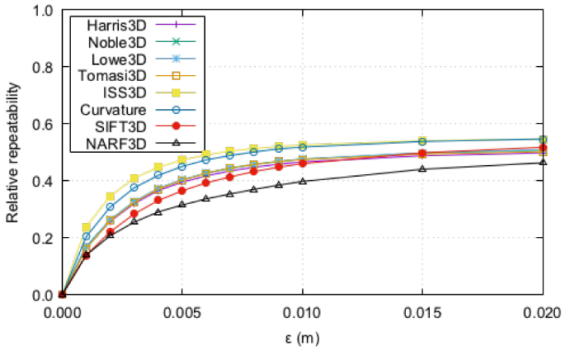

**Slika 3.24. Ponovljivost za rotaciju od 20° - tipkovnica**



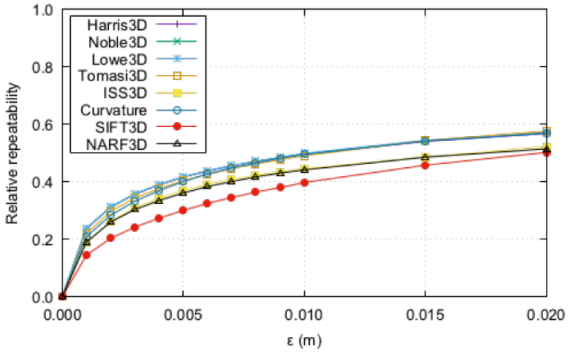

**Slika 3.25. Ponovljivost za rotaciju od 10° - dječja kolica**




**Slika 3.26. Ponovljivost za rotaciju od 20° - dječja kolica**




**Slika 3.27. Ponovljivost za rotaciju od 10° - vrč**




**Slika 3.28. Ponovljivost za rotaciju od 20° - vrč**

Može se ustanoviti da ponovljivost ključnih točaka ovisi i o samoj geometriji objekta, tako da nije moguće u potpunosti utvrditi detektor ključnih točaka koji bi bio najbolji za sve situacije.

Kao i u prošlom testu ponovljivost raste povećanjem radijusa za klasifikaciju najbližih susjeda, što uzrokuje pad ukupnog broja točaka od interesa.

Uzevši u obzir presjek oba testa ISS3D detektor se pokazao kao najbolje rješenje promatrajući rotaciju i translaciju. Kao najveći nedostatak svih detektora je kalkulacija kod velikih rotacija gdje drastično opada broj ponovljenih ključnih točaka što može dovesti do problema prilikom usporedbe deskriptora modela i scene.

Unutar rada dodatno su testirane Harris3D ključne točke jer kao i ISS3D uzimaju svojstvo boje prilikom izračuna, te voxel ključne točke koje se postavljaju nasumično u svakoj iteraciji jer ne uzimaju svojstvo geometrije niti boje, ali su zanimljive zbog vrlo brze kalkulacije i jednostavnog podešavanja njihova broja.

SIFT3D detektor također je pokazao pozitivne rezultate, međutim njegov izračun ključnih točaka traje višestruko dulje nego primjerice kod ISS3D-a i Harris3D-a kao što je prikazano u tablici 2.

**Tablica 2. Prosječno vrijeme potrebno detektorima za izračun ključnih točaka izračunato na temelju više objekata [15]**

| Detektor ključnih točaka | Srednji broj izračunatih ključnih točaka | Srednje vrijeme[s] |
|---|---|---|
| ISS3D | 86,24 | 1,07 |
| Harris3D | 85,63 | 1,05 |
| SIFT3D | 87,46 | 9,54 |

Voxel ne pripada modulu ključnih točaka nego je to prenamijenjen Voxel filter opisan u poglavlju 3.2.1.2.

### *3.3.4 Deskriptori*

Nakon izvršavanja potrebnih filtracija i prethodno navedenih izračunatih značajki, poput normala i ključnih točaka, odlučujuću ulogu za prepoznavanje objekata imaju deskriptori. Deskriptori se dijele na globalne i lokalne, ovisno o veličini podrške oko točke koju opisuju. Lokalni deskriptori opisuju predmet promatrajući susjedstvo oko svake izračunate ključne točke što ih čini deskriptivnim i robusnim na buku i okluzije. Okluzija u viziji označava pojavu kada objekt djelomično izlazi iz kadra ili je prekriven nekim drugim objektom na sceni.

Za razliku od lokalnih, koji opisuju dijelove objekta, globalni deskriptori promatraju objekt kao kompletnu cjelinu što ih čini kompaktnima i vremenski efikasnijima, međutim nemaju mogućnost analizirati specifičan detalj na predmetu, nisu toliko robusni na okluziju i buku, te nužno zahtijevaju segmentaciju podloge [17].

Unutar rada promatrani su lokalni deskriptori s pretpostavkom da je moguća prisutnost okluzija i da nije nužno staviti težište na brzinu izvođenja procesa već na robusnost i uspješnost prepoznavanja.

Kako bi deskriptori, bilo globalni ili lokalni, davali zadovoljavajuće rezultate trebali bi biti sljedećih karakteristika[18]:

1. Robusni na transformacije – transformacije kao što su translacija i promjena orijentacije ne bi smjele utjecati na konačan rezultat, tj. na broj nađenih korespondencija.
2. Robusni na buku – blage varijacije buke ne bi smjele drastično ugrozili vrijednost izračunatog deskriptora
3. Robusni na rezoluciju – očekivanje je da bi za različite gustoće oblaka točaka deskriptori trebali biti slični/isti ukoliko je geometrija oko promatrane točke ista.

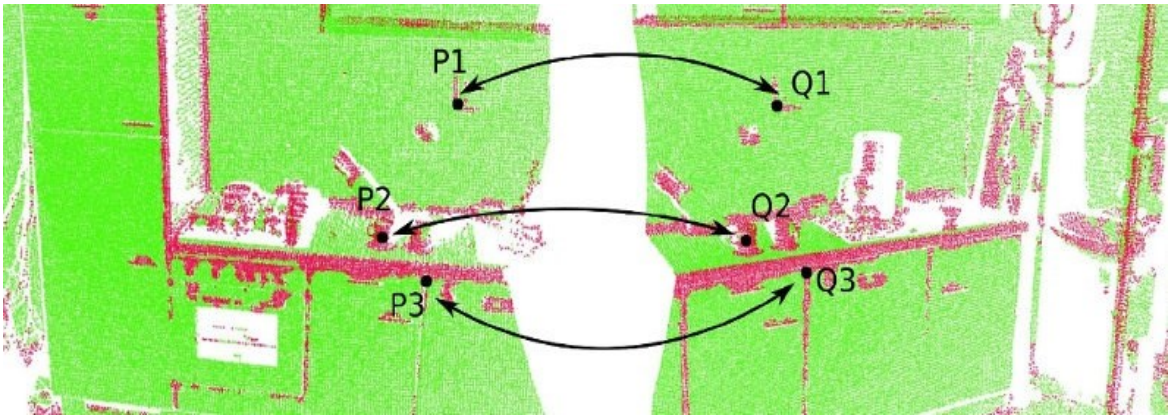
Nadalje, lokalni deskriptori se mogu podijeliti u tri kategorije s obzirom na način rada koji ih karakterizira: funkcija potpisa (*engl. signatures*), histogrami i hibridi.

Deskriptori s funkcijom potpisa za izračun zahtijevaju invarijantni lokalni referentni okvir (eng. *Local Reference Frame*), odnosno koordinatni sustav koji će djelovati samo u ograničenom dijelu prostora, te kalkulaciju ključnih točaka na lokalnom području. Karakteristika takvih deskriptora je što su vrlo opisni, no vrlo osjetljivi na buku.

Deskriptori u kategoriji histograma koriste lokalne geometrijske značajke, ponajviše odnos između kutova normala koje zatvaraju u odnosnu na lokani koordinatni sustav (*LRF*). Takvi deskriptori su manje opisni, ali zato vrlo robusni na buku.

Hibridni lokalni deskriptori koriste svojstva potpisa i histograma što ih čini deskriptivnima, ali ujedno i robusnim. Negativno je što prethodno zahtijevaju izračun normala i ključnih točaka što može usporiti program.

Za bolje razumijevanje deskriptori se mogu interpretirati kao numerička vrijednost pripisana utjecajem geometrije oko promatrane točke. Primjerice, ukoliko geometrija modela odgovara geometriji na sceni, izračunati deskriptori će imati vrlo slične ili iste numeričke vrijednosti što označava korespondenciju, odnosno sličnost između modela i scene ili bilo koja dva oblaka točaka. [Slika 3.29]



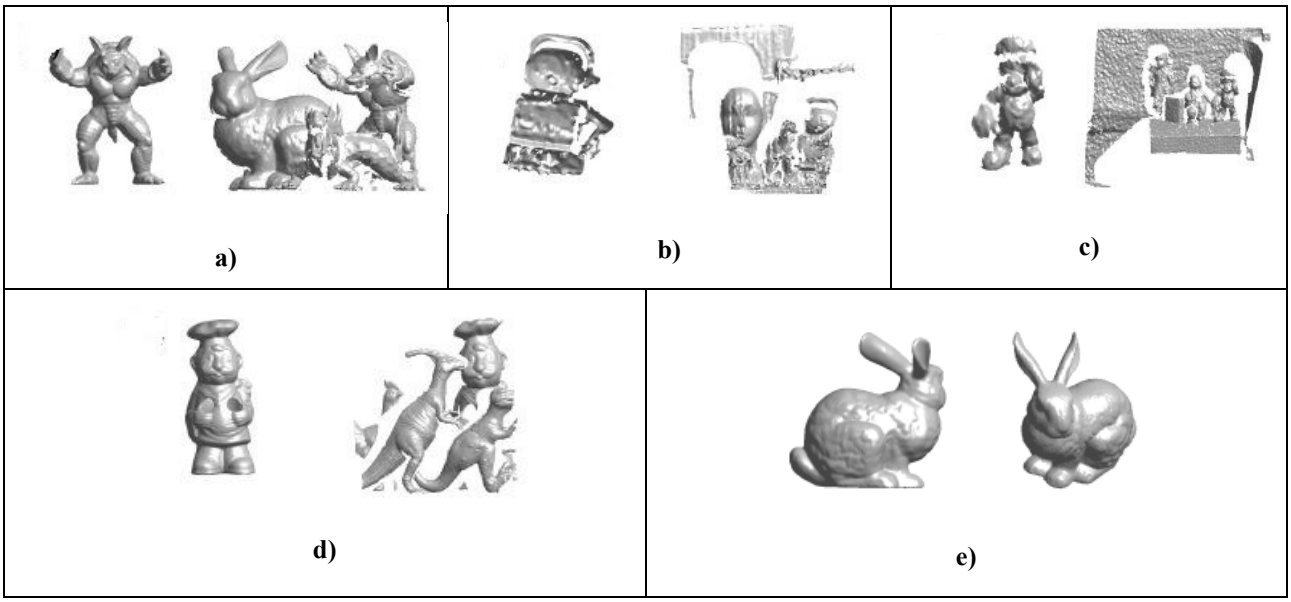

**Slika 3.29. Usporedba deskriptora s dvije identične scene, , točka $P_1 \cong Q_1, P_2 \cong Q_2, P_3 \cong Q_3$**

U tablici 3. prikazani su neki od lokalnih deskriptora dostupnih unutar PCL-a i njihova osnovna svojstva.

**Tablica 3. Neki od prisutnih lokalnih deskriptora unutar PCL-a**

| Deskriptor | Kategorija | Uporaba boje |
|---|---|---|
| **SPIN** | Histogram | Ne |
| **3DSC** | Histogram | Ne |
| **FPFH** | Histogram | Ne |
| **PFH** | Histogram | Ne |
| **PFHRGB** | Histogram | Da |
| **ISS** | Histogram | Ne |
| **KPQ** | Signature | Ne |
| **3D SURF** | Signature | Ne |
| **MeshHoG** | Signature | Da |
| **SHOT** | Hibrid | Ne |
| **CSHOT** | Hibrid | Da |

U literaturi [19] napravljeno je istraživanje o uspješnosti deskriptora ovisno o kvaliteti seta podataka. Tako primjerice podaci dobiveni iz laserski skenova ili CAD modela su kategorizirani kao visoko kvalitetni dok podaci dobiveni iz Kinecta nisko ili srednje. Svi setovi podataka sadrže okluzije, te su u nesređenom stanju.




**Slika 3.30. Različiti setovi podataka, a) miješani set podataka različite kvalitete, b) spacetime stereo set – srednja kvaliteta c) kinect – niska kvaliteta, d) laser skener – visoka kvaliteta, e) skeniranje objekata – visoka kvaliteta**

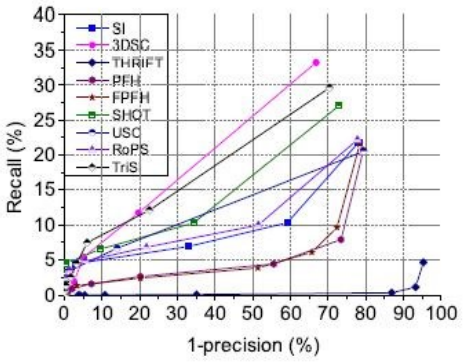
Redom su prikazani rezultati deskriptora dobiveni za setove podataka a), b), c), d) i e) u ovisnosti opoziva (*engl. Recall*) na postignutu preciznost. Opoziv se definira kao omjer točnih korespondencija i ukupnog broja pronađenih korespondencija između modela i scene. Točne korespondencije proizlaze iz algoritama za vrednovanje deskriptora dok ukupan broj korespondencija iz početnog grupiranja sličnih deskriptora putem kd – stabla.

$$opoziv = \frac{točne\ korespondencije}{ukupan\ broj\ korespondencija}.$$

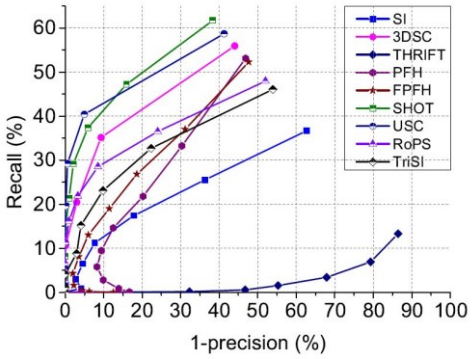
Preciznost (eng. *Precision*) se definira kao omjer lažnih korespondencija i zbroja točnih i lažnih korespondencija.

$$1 - preciznost = \frac{lažne\ korespondencije}{točne\ korespondencije + lažne\ korespondencije},$$

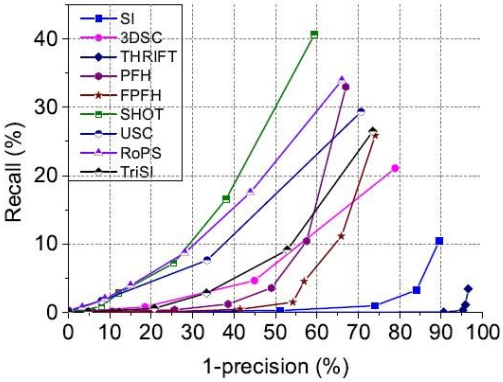
Iz navedenih izraza slijedi da se deskriptor može klasificirati kao bolji ukoliko ima veći opoziv i veću preciznost koja je izrazom prikazana recipročno, tj. manji postotak - veća preciznost.



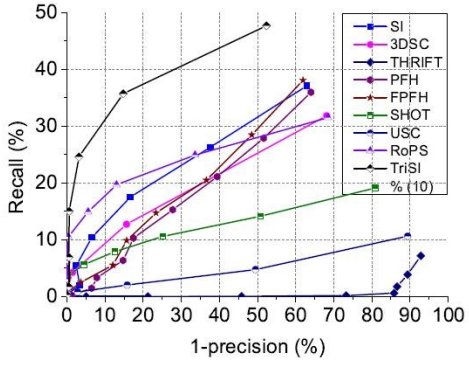

**Slika 3.31. Rezultati za set podataka a)**



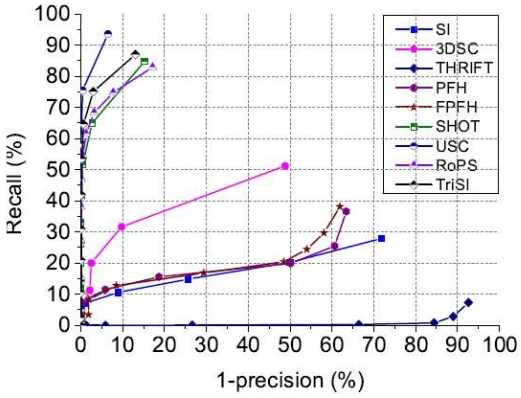

**Slika 3.32. Rezultati za set podataka b)**




**Slika 3.33. Rezultati za set podataka c)**




**Slika 3.34. Rezultati za set podataka d)**



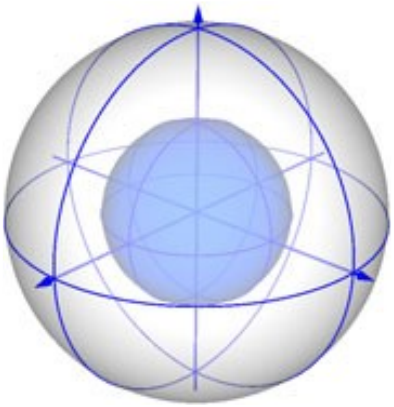

**Slika 3.35. Rezultati za set podataka e)**

Iz priloženog se može ustanoviti da izbor deskriptora uvelike ovisi o kvaliteti korištenih podataka. Za potrebe ovoga rada, gdje se koristi Kinect v2 kamera koja daje podatke niske do srednje kvalitete, odabrati će se hibridni SHOT lokalni deskriptor. Kako bi se iskoristila prednost RGB senzora razmotrit će se i unaprijeđena verzija SHOT deskriptora, CSHOT (*color – SHOT).*

### *3.3.4.1 Shot deskriptor*

SHOT (*Signature of Histograms of Orientations*) je hibridni lokalni deskriptor kojem su za estimaciju potrebne ključne točke te normale što ga čini dovoljno deskriptivnim, a ujedno i robusnim na buku i okluzije. Kao i većinu deskriptora karakterizira ga robusnost na translaciju i rotaciju.

Sadrži sferičnu strukturu oko središnje točke (ključne točke) za koju je definiran lokalni koordinatni sustav. Sferična struktura je podijeljena u 32 volumena u smjeru azimuta, visine i radijalne osi, zatim za svaki volumen se računa 3D histogram, odnosno niz numeričkih vrijednosti na temelju geometrijskih značajka unutar volumena. U konačnici svi izračunati histogrami se povezuju kako bi se dobila konačna vrijednost deskriptora.



**Slika 3.36. Struktura SHOT deskriptora oko središnje točke**

### *3.3.4.2 CSHOT deskriptor*

Color – SHOT je unaprijeđena verzija SHOT deskriptora gdje se za svaki volumen računa dodatni histogram na temelju RGB informacija unutar scene. Time se otvara mogućnost za ostvarivanje više pozitivnih korespondencija i veća robusnost na buku i okluzije. Iako mnoga istraživanja ukazuju na bolje rezultate CSHOT-a u odnosu na SHOT [20][21], izuzev vremenskog zahtjeva za izračun, eksperimentom unutar rada utvrđeni su lošiji rezultati.

Lošiji rezultati mogu se pripisati izostanku dodatnog osvjetljenja čime nije postignuta konzistentnost boje u svim dijelovima scene uslijed utjecaja sjene i različitog osvjetljenja.

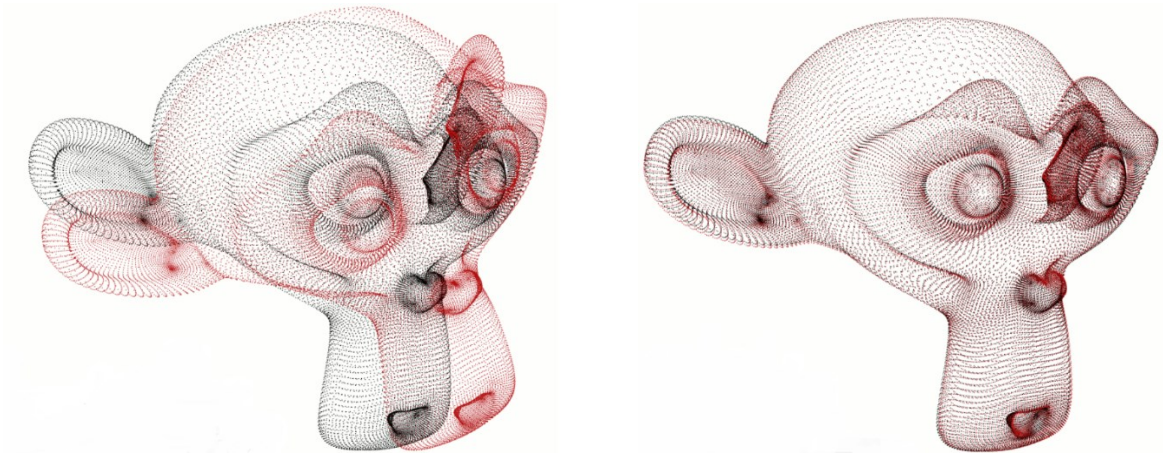
### *3.3.6 ICP registracija*

Pojam registracije u 3D viziji odnosi se na poravnanje dvaju istih oblaka točaka nazvanima izvor (*eng. source*) i cilj (*eng. target*) od kojih je cilj fiksan, a izvor mobilan. Ideja je minimizirati udaljenost modela s jednakim objektom na sceni te u konačnici dobiti njihovu matricu transformacije kao razliku u translaciji i rotaciji. [22] Da bi ICP (*eng. Iterative closest points*) registracija izvršila zadatak prethodno zahtjeva kalkulaciju inicijalne pozicije objekta na sceni što je upravo bio zadatak deskriptora i njihove usporedbe, tako da se ICP registracija može smatrati finim pozicioniranjem koje će ispraviti grešku inicijalne transformacije.

Algoritam radi na način da računa grešku pozicije svakog para točaka između dva oblaka točaka, tj. izvora i cilja. Konačna transformacija se određuje na temelju greške svih izračunatih parova točaka. Iz samog naziva očitava se da algoritam djeluje iterativno gdje u svakom koraku smanjuje razliku između dva oblaka. Algoritam sadrži nekoliko osnovnih kriterija:

1. Broj iteracija je dosegao maksimum definiran od korisnika
2. Izračunata greška između dva oblaka je manja od one koju je definirao korisnik
3. Suma kvadrata euklidske udaljenosti je manja od one koju je definirao korisnik

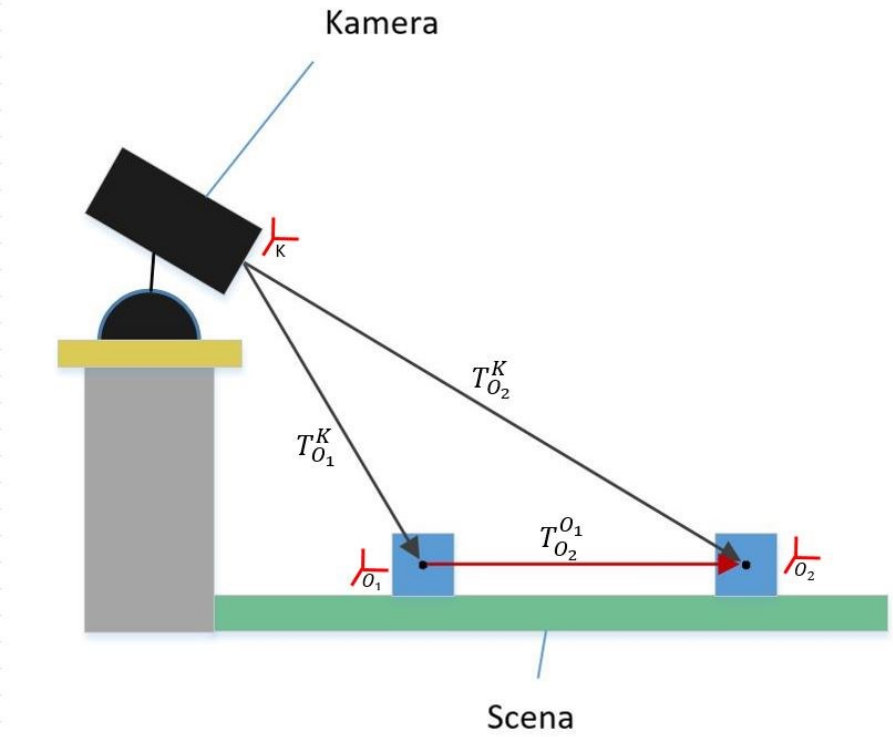
Ukoliko u maksimalnom broju iteracija nije postignuta željena vrijednost fitnes funkcije algoritam odbacuje rezultat, a program kreće ispočetka. Za razliku od izračuna i usporedbe deskriptora ICP je veoma brz algoritam što u konačnici nema velikog utjecaj na vremensko trajanje cijelog programa, ali uvelike optimizira konačnu matricu transformacije. Algoritam ICP registracije je već samostalno dostatan za izradu korisnih aplikacija, primjerice poput istraživanja u literaturi [23]



**Slika 3.37. Poravnanje dvaju oblaka u *n* iteracija ICP algoritmom [24]**

### *3.3.7 Konačna matrica transformacije*

Za određivanje razlike u poziciji referentnog modela i objekta na sceni potrebno je znati njihovu transformaciju u odnosu na ishodište kamere. Poznavanjem tih dviju transformacija moguće je odrediti razliku u pomaku i rotaciji između referentnog i prepoznatog predmeta. Udaljenost predmeta od ishodišta ili međusobna udaljenost modela i objekta promatra se iz centroida, odnosno težišta mase. Na sljedećoj slici prikazane su potrebne transformacije za dobivanje informacije o položaju objekta na sceni.




**Slika 3.38. Transformacije između oblaka točaka**

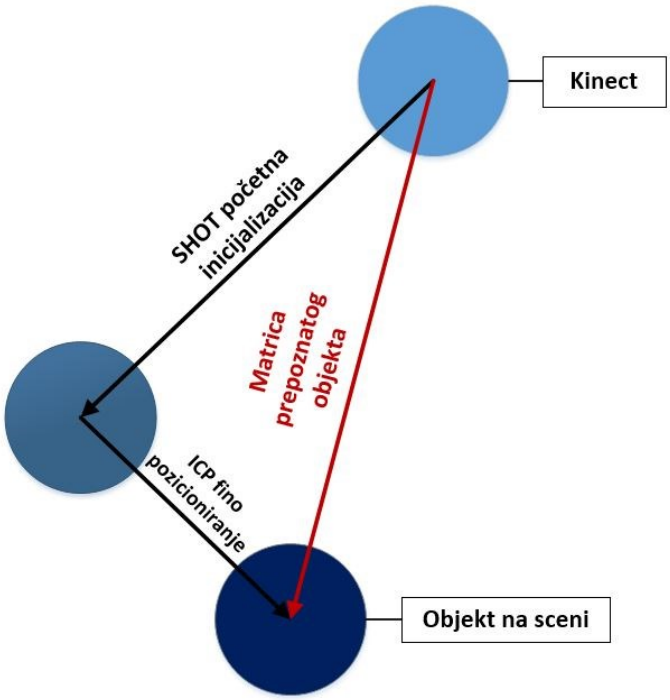
Oznaka $T^K_{O_1}$ označava transformaciju referentnog koordinatnog sustava (koordinatni sustav modela) $O_1$ u odnosu na Kinect, $T^K_{O_2}$ transformaciju prepoznatog objekta u odnosu na Kinect, dok $T^{O_1}_{O_2}$ transformaciju između dva objekta na sceni.

Transformacija $T^{O_1}_{O_2}$ se šalje robotu putem TCP konekcije kao konačan rezultat, a može se izračunati iz izraza:

$$T^{O_1}_{O_2} = (T^K_{O_1})^{-1} \cdot T^K_{O_2}.$$

Transformacija referentnog objekta u odnosu na ishodište kamere je poznata varijabla. Matrica transformacije prepoznatog objekta dobivena usporedbom deskriptora, a zatim fino podešena pomoću ICP registracije. U konačnici $T_{O_2}^K$ transformacija se dobiva iz izraza:

$$T_{O_2}^K = T_{ICP} \cdot T_{SHOT.}$$




**Slika 3.39. Fino podešavanje matrice transformacije prepoznatog objekta**

### *3.3.8 Izrada modela*

Ukratko će se predstaviti dva načina izrade modela u obliku oblaka točaka koji služi za usporedbu s objektima na scene.

Prvi postupak se sastoji od uzimanja scene s Kinect kamerom, zatim potrebnim filtracijama i ekstrakcijom klastera izolirati model, tj. iz svih točki scene izolirati samo one koje predstavljaju model. Prilikom postupka model mora biti jasno vidljiv na sceni, bez ikakvih okluzija, a filtracijama je potrebno očistiti buku kako ne bi došlo do krive estimacije pozicije, tj. korespondencija, prilikom usporedbe.

Program za izradu modela sastoji se od filtracija i rekonstrukcije površine opisanih u poglavljima 3.2.1 i 3.2.2, te algoritma za ekstrakciju klastera opisanog u poglavlju 3.2.4.

Nakon što algoritam pronađe klastere (ili samo jedan klaster) zasebno se spremaju u .pcd datoteku čime je napravljen model. Pošto kamera nije u mogućnosti iz jednog kadra prikazati potpun model od 360° ponekad je potrebno ICP registracijom opisanom u poglavlju 3.3.6 spojiti više kadrova u jedan. Registracija nije potrebna ukoliko model simetrijom može prikazati sve svoje značajke iz jednog kadra.

Drugi način izrade .pcd datoteke je pomoću CAD softvera. Napravljeni 3D model se pohranjuje u .stl datoteku, zatim pomoću softvera MeshLab radi se konverzija iz .stl u .obj, te u konačnici unutar PCL-a postoji instaliran program koji .obj pretvara u željeni .pcd format. Model napravljen u CAD-u predstavlja idealan model u potpunom prikazu od 360° bez šumova, deformiranih površina i ostalih nedostataka.

Trodimenzionalni podaci dobiveni iz Kinecta v2 mogu se klasificirati kao podaci niske kvalitete, posebice kod prikaza dimenzijski manjih objekata, te kao takvi utvrđeno je manjak korespondencija prilikom usporedbe s idealnim podacima dobivenih iz CAD-a, stoga za izradu modela korišten je prvi način.

### *3.3.9 Prednosti i nedostaci metode lokalnih značajki*

Složeniji pristup problemu uspješno je otklonio neka osnovna ograničenja prethodne metode, poput nemogućnosti točnog definiranja objekta koji se želi prepoznati u nestrukturiranoj okolini te problem spajanja dvaju predmeta koja stoje blizu u jedan klaster.

Iako su deskriptori, vodeće veličine koje opisuju lokalnu geometriju, invarijantni na rotaciju i translaciju testiranjem se ustanovilo da veće rotacije mogu drastično utjecati na manji konačni broj korespondencija. Takva pojava može proizaći radi nekonzistentnosti normala objašnjenih u poglavlju 3.3.2 i/ili ključnih točaka objašnjenih u poglavlju 3.3.3 između scene i modela. Problem se može riješiti višestrukom rotacijom modela koji su pohranjeni u bazi znanja i paralelno se uspoređuju sa scenom, ili dohvaćanjem podataka bolje kvalitete, međutim to je potrebno ispitati eksperimentom.

Uslijed varijacije ponovljivosti ključnih točaka moguća je pojava neprecizne estimacije inicijalne pozicije dobivene iz korespondencija što opet utječe na lošije pozicioniranje ICP registracijom, takve pojave su češće prisutnije kod simetričnih objekata što će se pokazati kasnije testiranjem.

Prednosti:

- Mogućnost prepoznavanja i lokalizacije željenog objekta u nestrukturiranoj okolini
- Višestruko prepoznavanje istih objekata u jednoj iteraciji programa
- Izgradnja baze znanja sačinjene od različitih objekta
- Veliki izbor deskriptora i metoda za izračun ključnih točaka
- Više pristupa vrednovanju korespondencija za dobivanje robusnosti

Nedostaci:

- Procesorski zahtjevna metoda
- Velik broj međusobno ovisnih parametara koje je potrebno uskladiti
- Velike rotacije smanjuju broj korespondencija

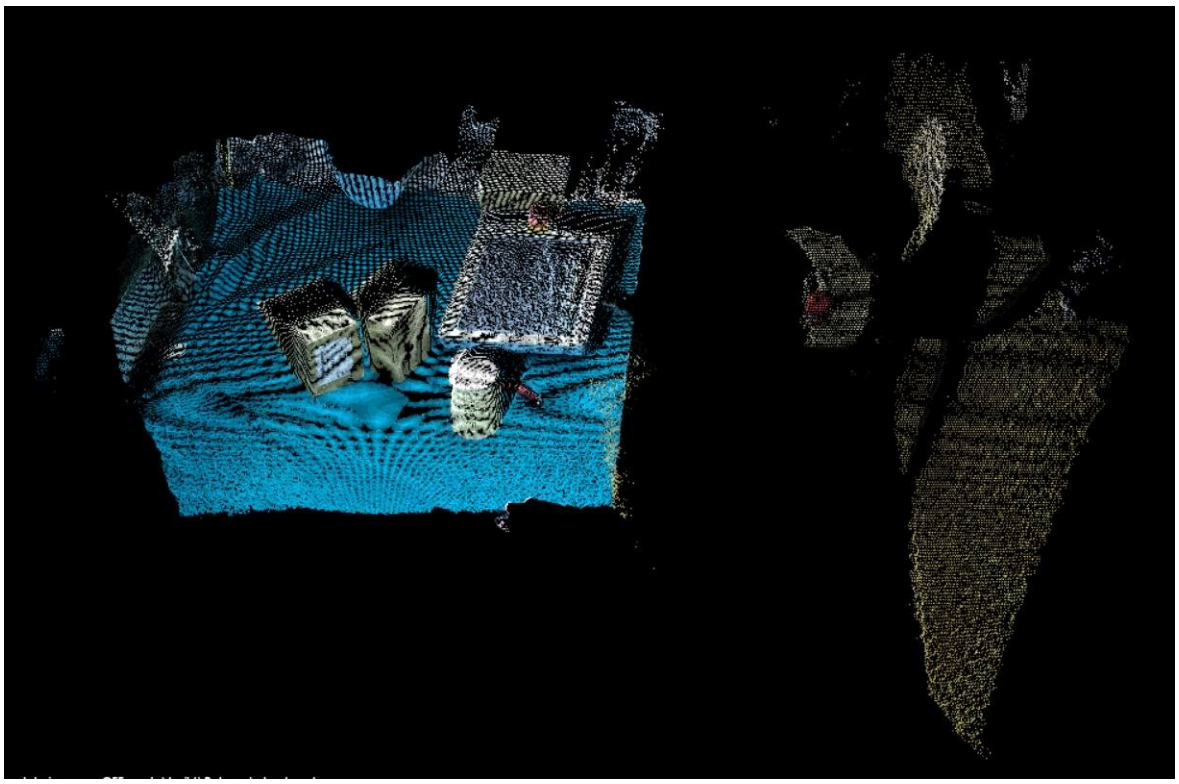
# 4. IMPLEMENTACIJA METODA PREPOZNAVANJA POMOĆU PCL

Iza teoretske razrade obaju metoda slijedi njihova implementaciju pomoću Point Cloud Library-a gdje će se navesti korištene klase, glavne funkcije unutar klasa te korišteni parametri. Radi boljeg razumijevanja pojedinih algoritama, ukoliko je moguće, prikazat će se na koji način djeluju na oblak točaka.

Usklađivanje parametra jedna je od osnovnih zadaća za pravilno funkcioniranje cjelokupnog programa. Važno je napomenuti da veličine koje označavaju neku dimenziju poput radijusa, pozicije točke, udaljenosti i slično su definirane u metrima.

## 4.1 Metoda ekstrakcije klastera

Klase i funkcije prikazane su redoslijedom kako su postavljenje u algoritmu. Početni oblak točaka bez ikakvih filtracija prikazan je na slici 4.1.



**Slika 4.1. Početni nefiltrirani oblak točaka**

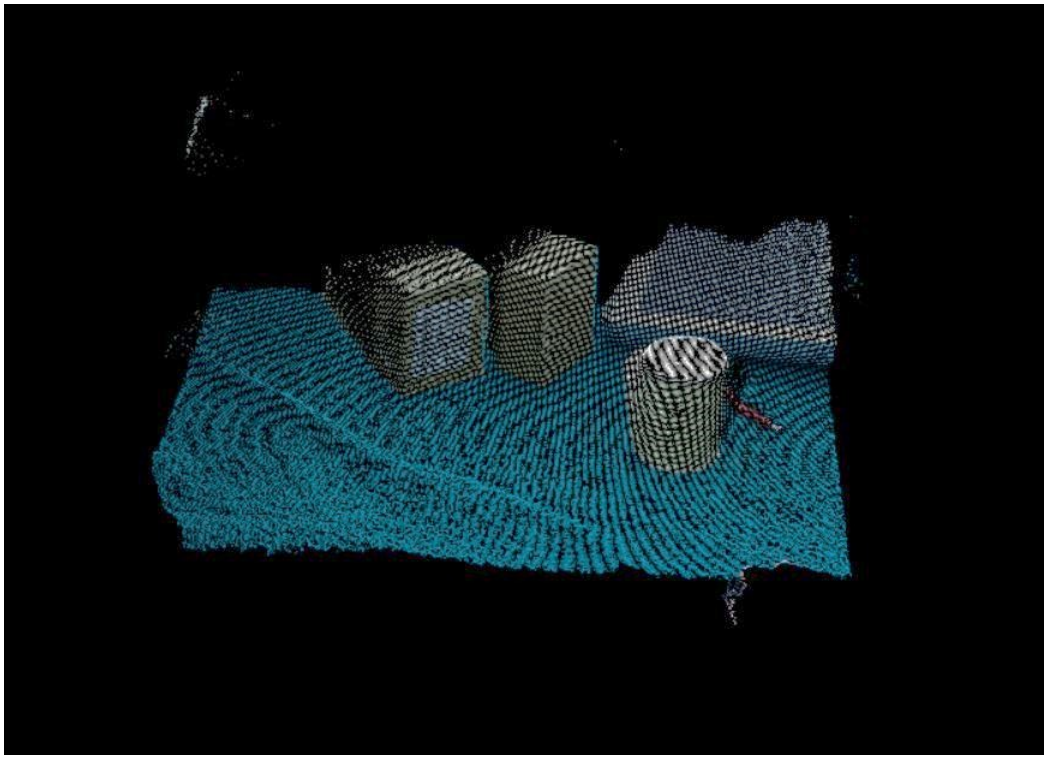
### *4.1.1 Passthrough filter*

Za pozivanje passthrough filtera opisanog u poglavlju 3.2.1.1 potrebno je uključiti zaglavlje (*eng. header*) `#include <pcl/filters/passthrough.h>`, a zatim pozvati klasu `pcl::PassThrough<PointType>`.

**Tablica 4. Passthrough filter**

| Naziv funkcije unutar klase | Opis funkcije | Vrijednost parametra |
|---|---|---|
| `.setInputCloud(naziv_oblaka)` | Ulazni podatak za obradu | - |
| `.setFilterFieldName("z")` | Defininiranje na os z | z |
| `.pass.setFilterLimits(limit_min, limit_max)` | Postavljanje ograničenja dimenzije duž osi z | `(0.0, 0.8)` |
| `.setFilterFieldName("x")` | Definiranje na os x | x |
| `.pass.setFilterLimits(limit_min, limit_max)` | Postavljanje ograničenja dimenzije duž osi x | `(-0.20,0.8)` |
| `pass.filter(naziv_oblaka)` | Izlazni podatak za daljnju obradu | - |

Vrijednosti parametara passthrough filtera striktno ovise o eksperimentalnom postavu kamere, tj. o odnosu položaja kamere naspram scene.



**Slika 4.2. Scena nakon passthrough filtera**

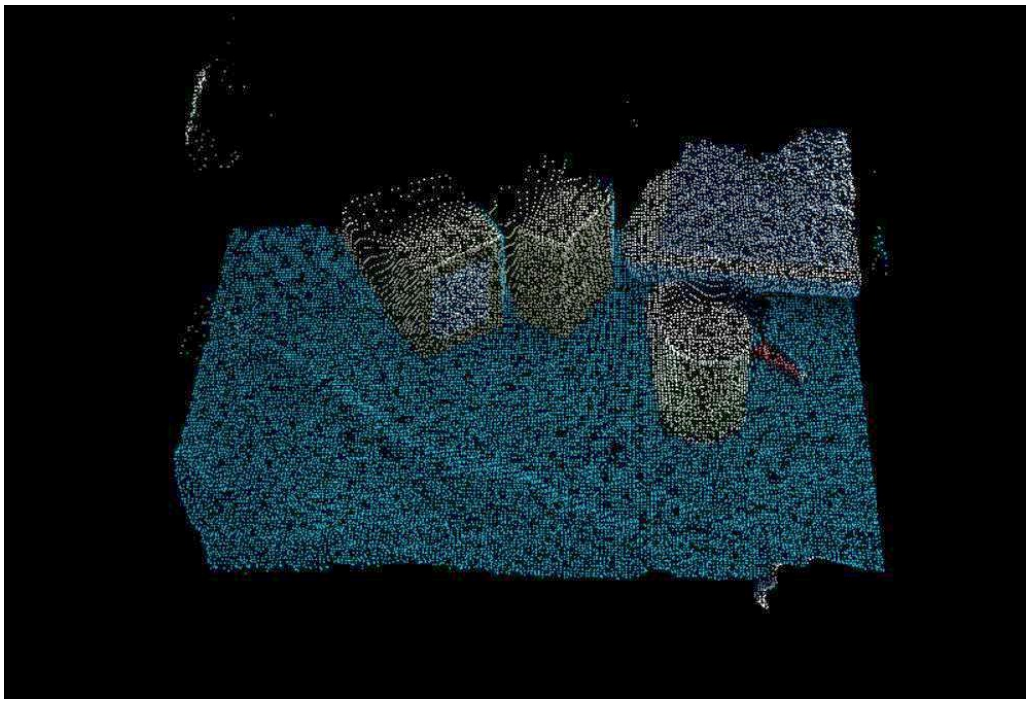
### *4.1.2 Voxel filter*

Za implementaciju voxel filtera opisanog u poglavlju 3.2.1.2 potrebno je uključiti zaglavlje `#include <pcl/filters/voxel_grid.h>`, a zatim pozvati klasu `pcl::VoxelGrid<PointType>`.

**Tablica 5. Voxel filter**

| Naziv funkcije unutar klase | Opis funkcije | Vrijednost parametra |
|---|---|---|
| `.setInputCloud(naziv_oblaka)` | Ulazni podatak za obradu | - |
| `.setLeafSize(x_os, y_os, z_os)` | Međusobna euklidska udaljenost točaka za svaku os - rezolucija | `(0.030f,0.030f, 0.030f)` |
| `.filter(naziv_oblaka)` | Izlazni podatak za daljnju obradu | - |

Zadani parametri u konačnici odgovaraju euklidskoj udaljenosti između dvije susjedne točke, u podacima oblaka točaka ta udaljenost se naziva i rezolucija.



**Slika 4.3. Scena nakon voxel filtera**

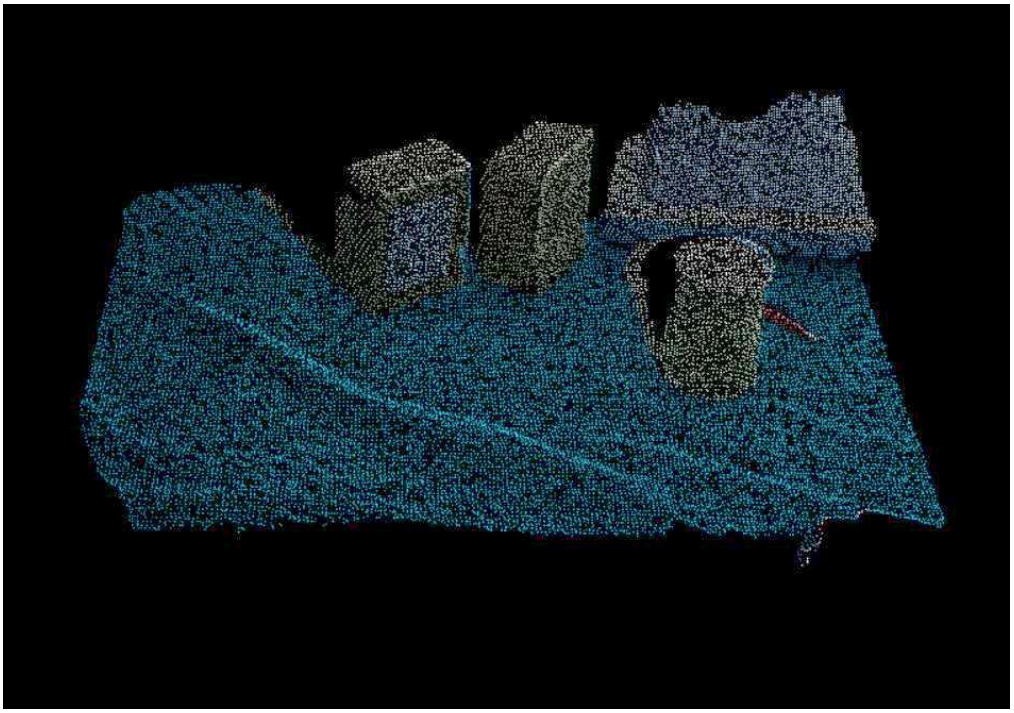
### *4.1.3 Statistical outlier removal filter*

Za pozivanje statistical outlier removal filtera opisanog u poglavlju 3.2.1.3 potrebno je uključiti zaglavlje `#include <pcl/filters/statistical_outlier_removal.h>`, a zatim pozvati klasu `pcl::StatisticalOutlierRemoval<PointType>`.

**Tablica 6. Statistical outlier removal filter**

| Naziv funkcije unutar klase | Opis funkcije | Vrijednost parametra |
|---|---|---|
| `.setInputCloud(naziv_oblaka)` | Ulazni podatak za obradu | - |
| `.setMeanK(broj_točaka)` | Broj susjednih točaka za analizu | `100` |
| `.setStddevMulThresh(stand_dev)` | Standardna devijacija | `0.8` |
| `.filter(naziv_oblaka)` | Izlazni podatak za daljnju obradu | - |

Što je vrijednost funkcije `.setStddevMulThresh(stand_dev)` veća filter će biti blaži što znači da mogu zaostat točke koje stvaraju šum, u suprotnome, ako je parametar premali, može se dogoditi da filter ukloni korisne točke koje sačinjavaju objekt.



**Slika 4.4. Scena nakon statistical outlier removal filtera**

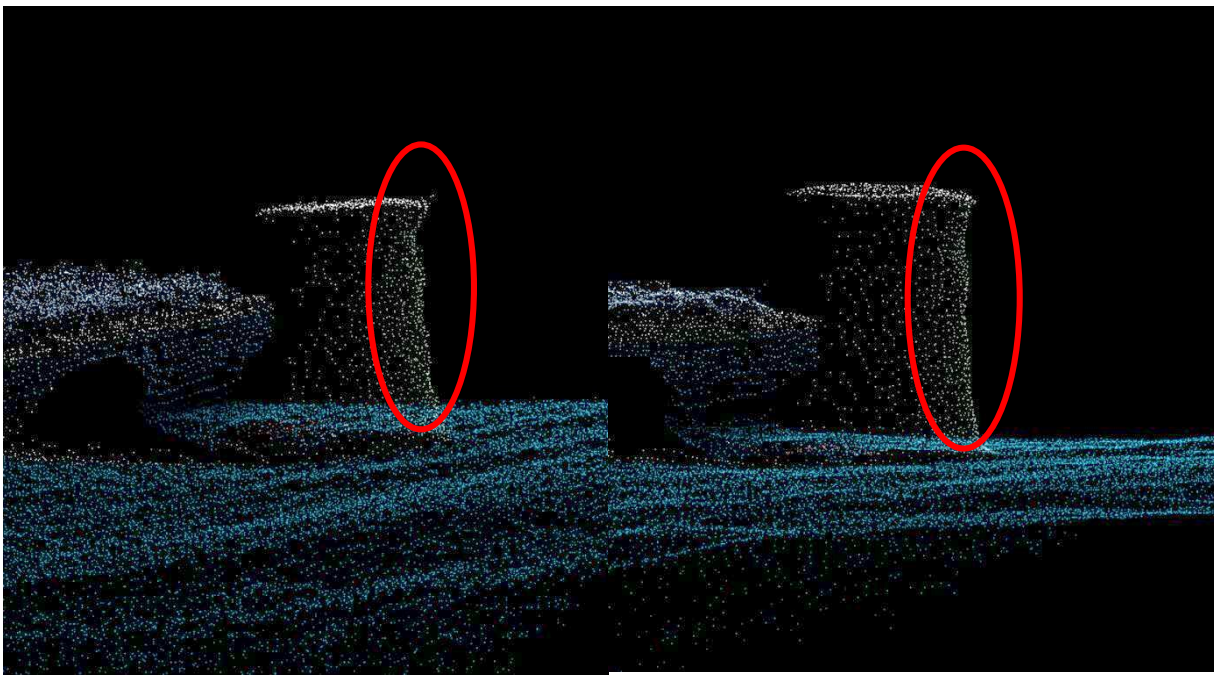
### *4.1.4 Rekonstrukcija površine*

Za implementaciju rekonstrukcije površine opisane u poglavlju 3.2.2 potrebno je uključiti zaglavlje `#include <pcl/surface/mls.h>`, a zatim pozvati klasu `pcl::MovingLeastSquares<PointType, pcl::PointXYZRGBNormal>.`

**Tablica 7. Rekonstrukcija površine**

| Naziv funkcije unutar klase | Opis funkcije | Vrijednost parametra |
|---|---|---|
| `.setInputCloud(naziv_oblaka)` | Ulazni podatak za obradu | - |
| `.setComputeNormals(boolean)` | MLS zahtjeva prethodno izračunate normale | `true` |
| `.setPolynomialFit(boolean)` | Korištenje aproksimacije polinoma | `true` |
| `.setSearchMethod(tree)` | Definiranje kd-stabla za pretragu najbližih susjeda | - |
| `.setSearchRadius(rad)` | Radijus za definiranje najbližih susjeda | `0.02` |
| `.process(naziv_oblaka)` | Izlazni podatak za daljnju obradu | - |

Ako je vrijednost funkcije `setSearchRadius(rad)` prevelika može značajno usporiti algoritam, a objekte na sceni prikazati deformirano čime se gube detalji i elementarne konture, stoga je potrebno pronaći srednju vrijednost koja će rekonstruirati površinu bez dodatnih deformacija.



**Slika 4.5. Prije (lijevo) i nakon (desno) MLS rekonstrukcije površine**

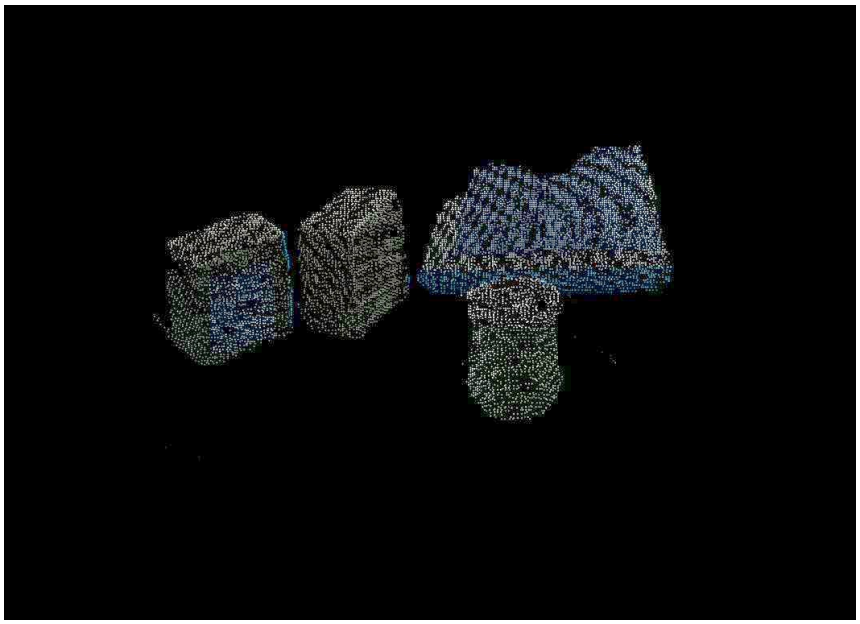
### *4.1.5 Uklanjanje podloge – RANSAC segmentacija*

Za implementaciju RANSAC segmentacije opisane u poglavlju 3.2.3.1 potrebno je uključiti zaglavlje `#include <pcl/segmentation/sac_segmentation.h>`, a zatim pozvati klasu `pcl::SACSegmentation<PointType>`.

**Tablica 8. RANSAC segmentacija**

| Naziv funkcije unutar klase | Opis funkcije | Vrijednost parametra |
|---|---|---|
| `.setModelType(pcl::SACMODEL_PLANE)` | Postavljanje modela koji se pretražuje | - |
| `.setMethodType(pcl::SAC_RANSAC)` | Metoda kojom se pretražuje | - |
| `setMaxIterations(integer)` | Brj RANSAC iteracija | `150` |
| `.setDistanceThreshold(double)` | Tolerancija s obzirom na zadani model pretrage | `0.011` |
| `.setInputCloud(naziv_oblaka)` | Ulazni podatak za obradu | - |
| `.segment(*inliers)` | Izuzimanje podataka koji su zadovoljili | - |
| `.filter(*naziv_oblaka)` | Izlazni podatak za daljnju obradu | - |

Parametar `.setDistanceThreshold(double)` potrebno je odabrati što preciznije, ukoliko je vrijednost premala postoji mogućnost da dijelovi podloge ne budu uklonjeni, dok u suprotnom, ako vrijednost bude prevelika, dio objekata koji stoje na podlozi bit će nepotrebno uklonjen.



**Slika 4.6. Scena bez poodloge**

### *4.1.6 Uklanjanje podloge – RGB filtracija*

Za implementaciju RGB filtracije opisane u poglavlju 3.2.3.2 potrebno je uključiti zaglavlje `#include <pcl/filters/conditional_removal.h>`, a zatim pozvati klasu `pcl::ConditionAnd<PointType>.`

**Tablica 9. RGB filtracija**

| Naziv funkcije unutar klase | Opis funkcije | Vrijednost parametra |
|---|---|---|
| `.setInputCloud(naziv_oblaka)` | Ulazni podatak za obradu | - |
| `.addComparison("r", integer_Max)` | Maksimalna vrijednost crvene boje koja se prikazuje | 255 |
| `.addComparison("r", integer_Min)` | Minimalna vrijednost crvene boje koja se prikazuje | 100 |
| `.addComparison("b", integer_Max)` | Maksimalna vrijednost plave boje koja se prikazuje | 255 |
| `.addComparison("b", integer_Min)` | Minimalna vrijednost plave boje koja se prikazuje | 100 |
| `.addComparison("g", integer_Max)` | Maksimalna vrijednost zelene boje koja se prikazuje | 255 |
| `.addComparison("g", integer_Min)` | Minimalna vrijednost zelene boje koja se prikazuje | 100 |
| `.filter(*naziv_oblaka)` | Izlazni podatak za daljnju obradu | - |

Poželjno je da podloga ili posuda u kojoj se nalaze objekti bude crne ili bijele boje, na taj način je jednostavnije odrediti područje boje koje treba ukloniti jer se eliminira moguća razlika u kalibraciji boje kamere i ekrana, tj. traženje točnog područja.

Scena nakon djelovanja RGB filtracije izgleda identično/slično kao na slici 4.6.

### *4.1.7 Ekstrakcija klastera*

Za ekstrakciju klastera opisanu u poglavlju 3.2.4 potrebno je uključiti zaglavlje `#include <pcl/segmentation/extract_clusters.h>`, a zatim pozvati klasu `pcl::EuclideanClusterExtraction<PointType>.`

**Tablica 10. Ekstrakcija klastera**

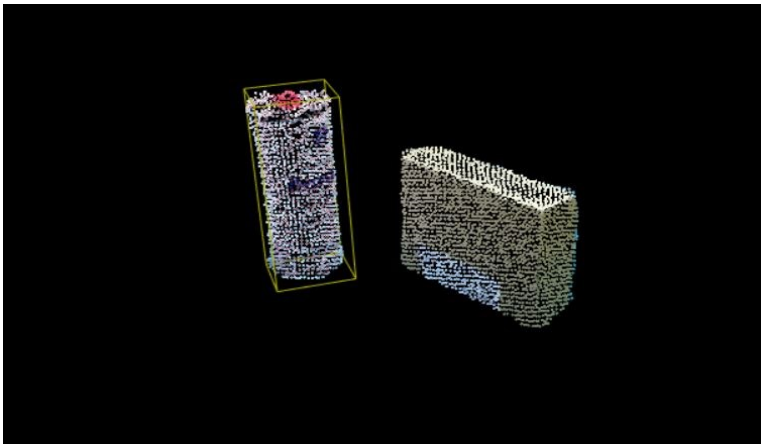
| Naziv funkcije unutar klase | Opis funkcije | Vrijednost parametra |
|---|---|---|
| `.setInputCloud(naziv_oblaka)` | Ulazni podatak za obradu | - |
| `.setClusterTolerance(double)` | Potrebna udaljenost između dva objekta da budu proglašeni kao odvojeni klasteri | `0.02` |
| `.setMinClusterSize(integer)` | Minimalni broj točaka u klasteru | `550` |
| `setMaxClusterSize(integer)` | Maksimalni broj točaka u klasteru | `10000` |
| `.setInputCloud(naziv_oblaka)` | Ulazni podatak za obradu | - |
| `.segment(*inliers)` | Izuzimanje podataka koji su zadovoljili | - |
| `.filter(*naziv_oblaka)` | Izlazni podatak za daljnju obradu | - |

Parametar `.setClusterTolerance(double)` bi svakako trebao težiti nuli, tada bi algoritam mogao klasificirati dva objekta koja su stoje vrlo blizu kao dva odvojena klastera. Međutim, u radu s realnim podacima uspostavljeno je da ukoliko navedeni parametar bude premali jedan objekt može biti podijeljen na dva klastera što drastično umanjuje funkcionalnost programa.

### *4.1.8 Granične kutije*

Za integraciju ideje graničnih kutija opisanih u poglavlju 3.2.5 potrebno je uključiti zaglavlja #include <pcl/common/pca.h>, #include <pcl/common/centroid.h> i #include <pcl/common/transforms.h>.

Pošto implementacija ne zahtjeva podešavanje parametara neće se tablično prikazati, a cjelokupni kod dostupan je pri kraju diplomskog rada u datoteci metoda_ekstrakcije_klastera.cpp.



**Slika 4.7. Prikaz objekta omeđenog graničnom kutijom**

## 4.2 Metoda lokalnih značajki

Scena i model kod metode lokalnih značajki sadrže jednake filtere već opisane u poglavljima 4.1.1 do 4.1.4, s isto tako jednakim parametrima.

Normale, ključne točke te u konačnici deskriptori računaju se za model i za scenu.

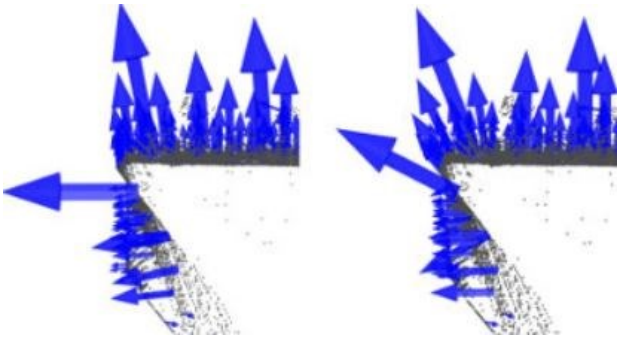
### *4.2.1 Izračun normala*

Za izračun normala opisanih u poglavlju 3.3.2 potrebno je uključiti zaglavlje `#include <pcl/features/normal_3d_omp.h>` , a zatim pozvati klasu `pcl::NormalEstimationOMP<PointType, NormalType>`.

**Tablica 11. Izračun normala**

| Naziv funkcije unutar klase | Opis funkcije | Vrijednost parametra |
|---|---|---|
| `.setKSearch(int)` | Broj k susjeda koji stvaraju tangentnu ravninu | `10` |
| `.setInputCloud(naziv_oblaka)` | Ulazni podatak za obradu | - |
| `.setViewPoint(int_x, int_y, int_z)` | Postavljanje točke gledišta | `(15, 100, 15)` |
| `.compute(*naziv_normala)` | Izlazni podatak normala | - |

Ukoliko se odabere preveliki broj k susjeda moguće je krivo definiranje tangentne ravnine, a samim time i kriva estimacija normala kao što je prikazano na slici ispod.



**Slika 4.8. Dobro odabran k broj susjeda (lijevo) i loše odabran k broj susjeda (desno) [25]**

Prema zadanim postavkama točka gledišta se nalazi u centru senzora kamere što može dovesti do problema opisanog na slici (def), stoga se definira novo gledište funkcijom .setViewPoint podalje od scene.

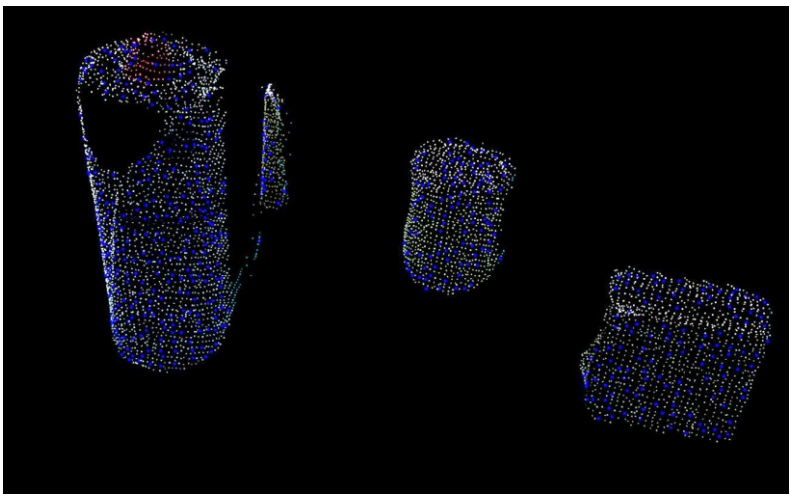
### *4.2.2 Izračun ključnih točaka*

Od korištenih detektora ključnih točaka tablično će se prikazati samo ISS detektor koji je ujedno i najčešće korišten dok će ostatak biti dostupan u cjelokupnom kodu na kraju rada. Za izračun ISS ključnih točaka opisanih u poglavlju 3.3.3 potrebno je uključiti zaglavlje `#include <pcl/keypoints/iss_3d.h>`, a zatim pozvati klasu `pcl::ISSKeypoint3D<PointType, PointType>.`

**Tablica 12 ISS3D ključne točke**

| Naziv funkcije unutar klase | Opis funkcije | Vrijednost parametra |
|---|---|---|
| `.setSearchMethod(tree)` | Definiranje kd-stabla za pretragu najbližih susjeda | - |
| `.setThreshold21(double)` | Omjer svojstvenih vrijednosti matrice kovarijance | `0.65` |
| `.setThreshold32(double)` | Omjer svojstvenih vrijednosti matrice kovarijance | `0.1` |
| `.setMinNeighbors(int)` | Minimalan broj susjednih točaka | `4` |
| `.setNumberOfThreads(int)` | Broj jezgri prilikom izračuna | `4` |
| `.setInputCloud(naziv_oblaka)` | Ulazni podatak za obradu | - |
| `.setSalientRadius(double)` | Postavljanje istaknutog radijusa | `0.007f` |
| `.setNormalRadius(double)` | Postavljanje normalnog radijusa | `0.02f` |
| `.compute(naziv_oblaka)` | Izlazni podatak za daljnju obradu | - |

ISS detektor zahtjeva niz parametara koje je potrebno uskladiti, stoga su početne vrijednosti parametara uzete iz literature [26], a zatim prilagođene dok nije postignuto željeno stanje koje obuhvaća broj i razmještaj ključnih točaka.



**Slika 4.9. ISS3D ključne točke**

### *4.2.3 Izračun deskriptora*

Od deskriptora tablično će se prikazati SHOT deskriptor pošto je ponajviše korišten radi najbolje dobivenih performansi. Za izračun SHOT deskriptora opisanog u poglavlju 3.3.4 potrebno je uključiti zaglavlje `#include <pcl/features/shot_omp.h>`, a zatim pozvati klasu `pcl::SHOTEstimationOMP<PointType, NormalType, DescriptorType>`.

**Tablica 13. Izračun deskriptora**

| Naziv funkcije unutar klase | Opis funkcije | Vrijednost parametra |
|---|---|---|
| `.setRadiusSearch(float)` | Podrška (radijus) oko ključne točke za izračun deskriptora | `0.08f` |
| `.setInputCloud(naziv_oblaka)` | Ulazni podatak prethodno izračunatih ključnih točaka | - |
| `.setInputNormals(naziv_normala)` | Ulazni podatak prethodno izračunatih normala | - |
| `.setSearchSurface(naziv_oblaka)` | Ulazni podatak cjelokupne scene i modela | - |
| `.compute(naziv_deskriptora)` | Izračunati deskriptori | - |

### *4.2.4 Usporedba i vrednovanje deskriptora*

Za usporedbu deskriptora opisanu u poglavlju 3.3.5 potrebno je uključiti zaglavlje `#include <pcl/kdtree/kdtree_flann.h>`, a zatim uključiti klasu `pcl::KdTreeFLANN<DescriptorType>`.

Pomoću kd-stabla potrebno je usporediti deskriptore scene i modela. Interval kvadratne udaljenost za SHOT deskriptore je od 0 do 1, gdje nula predstavlja idealan slučaj i sparivanje samo onih deskriptora koji su potpuno jednaki. Kako bi se postigao veći broj korespondencija, a da ujedno budu sastavljene od što sličnijih deskriptora, postavljena je kvadratna udaljenost 0.335f.

Uslijed mogućnosti lažnih korespondencija potrebno je ih dodatno vrednovati Hough3D algoritmom, ali prije toga je potrebno definirati lokalni referentni koordinatni sustav za ključne točke modela i scene.

Za implementaciju lokalnog koordinatnog referentnog sustava potrebno je uključiti zaglavlje `#include <pcl/features/board.h>`, a zatim pozvati klasu `pcl::BOARDLocalReferenceFrameEstimation<PointType, NormalType, RFType>`.

**Tablica 14. Lokalni referentni koordinatni sustav**

| Naziv funkcije unutar klase | Opis funkcije | Vrijednost parametra |
|---|---|---|
| `.setRadiusSearch(float)` | Podrška (radijus) oko ključne točke za definiranje lokalnog koordinatnog sustava | `0.065f` |
| `.setInputCloud(naziv_oblaka)` | Ulazni podatak prethodno izračunatih ključnih točaka | - |
| `.setInputNormals(naziv_normala)` | Ulazni podatak prethodno izračunatih normala | - |
| `.setSearchSurface(naziv_oblaka)` | Ulazni podatak cjelokupne scene i modela | - |
| `.compute(naziv_RS)` | Izračunati deskriptori | - |

Radijus za definiranje lokalnog koordinatnog sustava mora biti sličan radijusu za izračun deskriptora.

Nakon definiranja lokalnog koordinatnog sustava slijedi vrednovanje korespondencija Hough3D algoritmom.

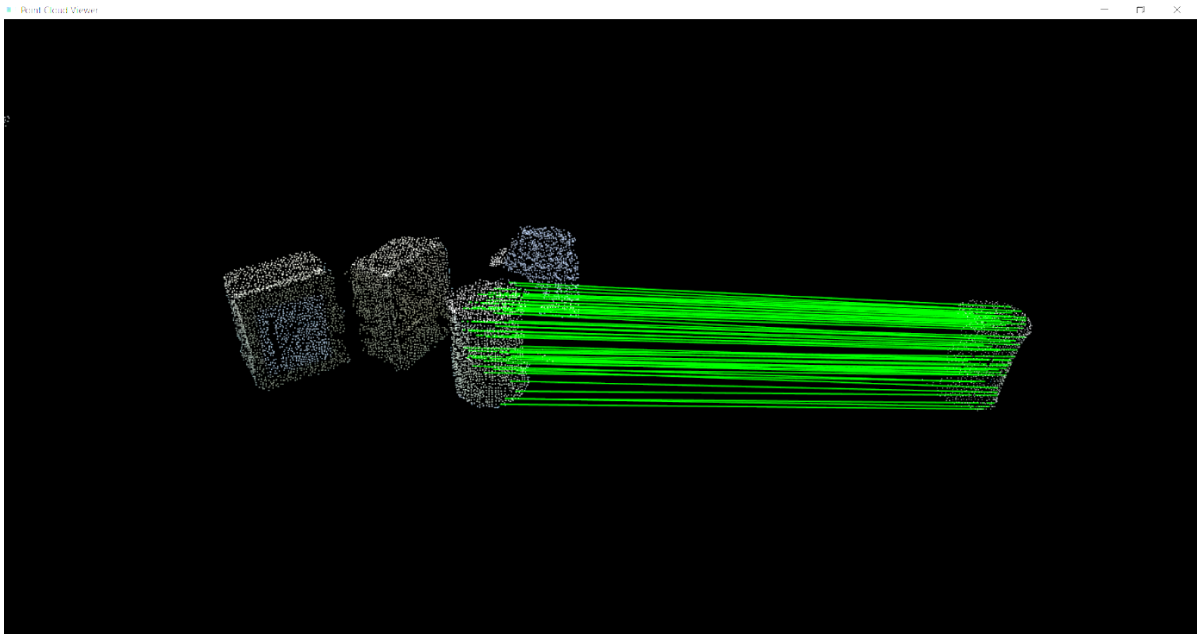
Za implementaciju Hough3D algoritma potrebno je uključiti zaglavlje `#include <pcl/recognition/cg/hough_3d.h>`, a zatim pozvati klasu `pcl::Hough3DGrouping<PointType, PointType, RFType, RFType>`.

**Tablica 15. Hough3D vrednovanje korespondencija**

| Naziv funkcije unutar klase | Opis funkcije | Vrijednost parametra |
|---|---|---|
| `.setHoughBinSize(float)` | Veličina granične kutije oko ključne točke | `0.065f` |
| `.setHoughThreshold(float)` | Definiranje granice potrebnih pozitivnih glasova za točne korespondencije | `7.5f` |
| `.setInputCloud(naziv_oblaka)` | Ulazni podatak prethodno izračunatih ključnih točaka | - |
| `.setInputRf(naziv_RF)` | Ulazni podatak izračunatog referentnog koordinatnog sustava | - |
| `.setModelSceneCorrespondences(vektor)` | Vektor korespondencija | - |

Potrebno je pažljivo odabrati navedene parametre za vrednovanje deskriptora. Što je vrijednost funkcije `.setHoughBinSize` manja bit će i manji broj korespondencija, ali one će biti točnije. Ukoliko se vrijednost postavi prevelikom moguć je veći broj lažnih korespondencija što rezultira prepoznavanju krivog objekta na sceni. Što je parametar `.setHoughThreshold` veće vrijednosti mogu se očekivati točnije korespondencije, te suprotno, ukoliko je manji više korespondencija, ali mogu se pojaviti one netočne.

U konačnici iz dobivenog vektora korespondencije scene i modela računa se matrica inicijalne pozicije objekta na sceni. Ukoliko se na sceni nalazi istovremeno više traženih objekata izlaz će se sastojati i od više matrica transformacija što je pogodno za rad s više robota.



**Slika 4.10. Pronađene korespondencije između modela i scene**

### *4.2.5 ICP registracija*

Za implementaciju ICP registracije opisane u poglavlju 3.3.6 potrebno je uključiti zaglavlje `#include <pcl/registration/icp.h>`, a zatim pozvati klasu `pcl::IterativeClosestPoint<PointType, PointType>`.

**Tablica 16. ICP registracija**

| Naziv funkcije unutar klase | Opis funkcije | Vrijednost parametra |
|---|---|---|
| `.setMaximumIterations(int)` | Maksimalni broj ICP iteracija | `100` |
| `.setInputSource(naziv_oblaka)` | Rotirani model SHOT transformacijom | - |
| `.setInputTarget(naziv_oblaka)` | Ulazni podatak scene | - |
| `.setMaxCorrespondenceDistance(float);` | Ukoliko je udaljenost između točaka veća od postavljenje granice neće ulaziti u proces registracije | `0.025` |
| `.setTransformationEpsilon(double)` | Maksimalna razlika između dvije korespondencije | `1e-9` |
| `.setEuclideanFitnessEpsilon(double)` | Razlika euklidske udaljenosti | `1e-8` |
| `.getFinalTransformation(float)` | Dobivena matrica transformacije | - |

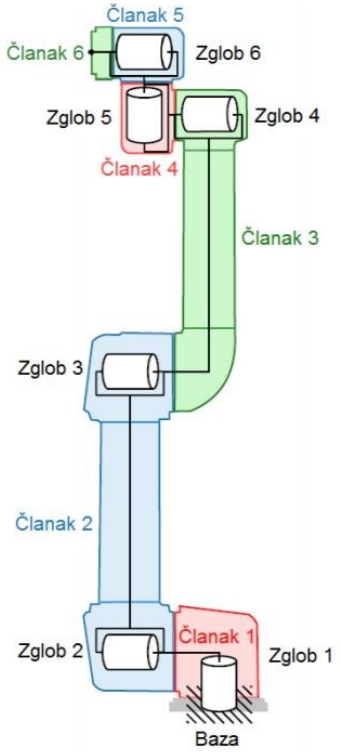
ICP registracijom je dobivena druga matrica transformacije, za fino pozicioniranje, koja u umnošku s prvom matricom daje konačnu matrice transformacije.

# 5. POVEZIVANJE S ROBOTOM

Konačnu matricu transformacije objekta potrebno je podijeliti s UR5 robotskom rukom uz prethodno ostvarenu komunikaciju s računalom.
UR5 je reprogramabilni višenamjenski manipulator danskog proizvođača Universal Robots čije su osnovne specifikacije dane u tablici 17.

**Tablica 17. Specifikacije UR5 robotske ruke [27]**

| | |
|---|---|
| Masa: | 18,4 kg |
| Nosivost: | 5 kg |
| Doseg: | 850 mm |
| Opseg zglobova: | +/- 360° |
| Brzina svih zglobova: | 180 °/s |
| Brzina alata: | 1 m/s |
| Ponovljivost: | +/- 1 mm |
| Promjer baze: | 149 mm |
| Broj stupnjeva slobode: | 6 |
| Veličina upravljačke jedinice: | 475 x 423 x 268 mm |
| Programiranje: | Polyscop grafičko sučelje |
| Potrošnja energije: | ~200W |
| Materijali: | Aluminij i plastika |
| Raspon radne temperature: | 0 – 50 °C |




**Slika 5.1. Zglobovi i segmenti UR5 robotske ruke [28]**

## 5.1 Ostvarivanje komunikacije

Komunikacija između računala i robotske ruke ostvarena je putem TCP-a (eng. *Transmission Control Protocol*) koji osigurava pouzdanu isporuku podataka kontroliranim redoslijedom. Kako bi stvorili preduvjeti za uspostavljanje veze potrebno je osigurati da se server i klijent nalaze u istoj podmreži (eng. *subnet*).

Unutar rada ostvarena je komunikacija gdje je PC u službi servera dok robotska ruka u ulozi klijenta. Server (PC) na robota šalje tekstualnu poruku sastavljenu od numeričkih vrijednosti pozicije i orijentacije oblika „(X, Y, Z, $R_x$, $R_y$, $R_z$)“, prva tri člana ukazuju na prostornu poziciju objekta, dok ostala tri njegovu orijentaciju u axis – angle zapisu. Ovakav način komunikacije je kontinuiran, odnosno server robotu konstantno, bez zaustavljanja, šalje podatke o lokaciji objekta.

Kako bi robot mogao pravilno primiti informaciju potrebno je poštivati određena pravila zapisivanja pozicije i orijentacije:

1. Podaci o poziciji i orijentaciji moraju biti zapisani u zagradi.
2. Svaki podatak mora biti razdvojen zarezom.
3. Na kraju izraza koji se šalje potrebno je dodati oznaku '\n' kako bi se svako slanje izvršilo u novom redu.

Uslijed stalnog slanja robot neće izvršavati novodolazeće naredbe o poziciji sve dok prvotno dobivenu naredbu ne izvrši do kraja, međutim preporuka je da server pošalje novu naredbu tek nakon što dobije povratni signal od klijenta da je izvršena prethodna operacija kako bi se povećala stabilnost sustava.

Ostatak programiranja vrši se na privjesku za učenje (eng. *teach pendant*) gdje se definiraju inicijalne postavke robota poput točke vrha alata i mrežne postavke, zatim redoslijed izvođenja programa te preuzete numeričke vrijednosti sa servera se pretvaraju u naredbe gibanja.

Dakako, povezivanje računala i robota također je moguće i obratnim putem kada je robot server, a PC klijent. UR sadrži karakteristične servere na portovima 30001, 30002 i 30003 koji se redom nazivaju primarni, sekundarni i *real time* server. Takav način povezivanja zahtjeva znanje URScript programskog jezika za upravljanje robotom.

**Tablica 18. Dostupni serveri na UR kontroleru**

| | Primarni server | Sekundarni server | Real –Time server |
|---|---|---|---|
| Port | 30001 | 30002 | 30003 |
| Frekvencija [Hz] | 10 | 10 | 500 |
| Primanje | URScript naredbe | URScript naredbe | URScript naredbe |
| Slanje | Stanje robota i dodatne poruke | Stanje robota | Stanje robota |

## 5.2 Program na robotu

Na robotskom privjesku za učenje potrebno je definirat točku vrha alata, prilagođeni koordinatni sustav te varijable koje se koriste za naredbe gibanja, primanje podataka sa servera, kao i brojači u *loop* petljama.

Naredbne gibanja su oblika *var_1= p*[x, y, z, Rx, Ry, Rz], gdje oznaka *p* definira da se radi o gibanju, i upravo tim redoslijedom, (x, y, z, Rx, Ry, Rz), su infromacije poslane sa servera. Također je potrebno uskladiti koordinatne osi kamere i robota dodavanjem odgovarajućeg predznaka, primjerice da smjer translacije +X na kameri bude i +X na robotu.

Pošto je ideja robotsko rukovanje predmetima rada potrebno je upravljanje vakumskom hvataljkom i organizirano odlaganje objekata nakon prihvata.

**Slika 5.2. Odlaganje objekata nakon robotskog izuzimanja**

# 6. EKSPERIMENTALNI POSTAV I REZULTATI

U ovom poglavlju prikazat će se međusobni odnos položaja kamere, scene i robotske ruke, također će se analizirat učinkovitost metode lokalnih značajki što podrazumijeva ispitivanje preciznosti ovisno o geometriji traženog objekta. Objekti će se geometrijski klasificirati s obzirom na broj ravnina simetrije, promotrit će se sljedeća 4 slučaja:

a) objekt s više od tri ravnine simetrije
b) objekt s tri ravnine simetrije
c) objekt s jednom ravninom simetrije,
d) objekt bez ravnine simetrije.

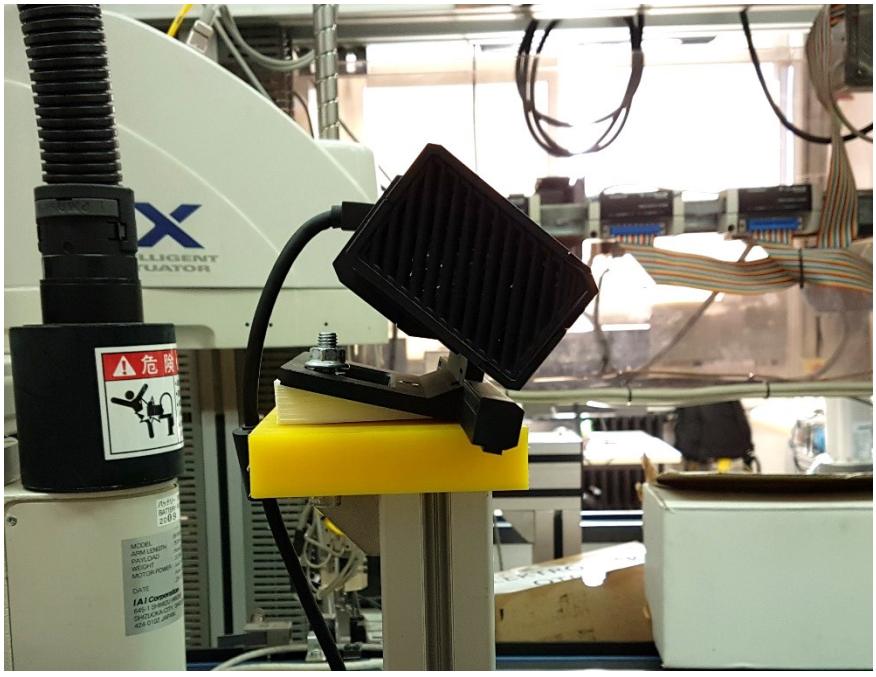
Kako bi rezultati bili što mjerodavniji za svaki učinjeni test navest će se osnovna pravila po kojima je izveden:

- za svaki test zapisat će se 10 uzastopnih matrica transformacija dobivenih u programu, zasebno inicijalna matrica i konačna matrica koja je ispravljena ICP registracijom,
- svaki objekt bit će ispitan po standardnim postavkama navedenim u prethodnim poglavljima,
- očekivana pozicija objekta u testu je (0, 0, 0) duž osi x, y, z i to bez učinjenih rotacija, odnosno idealna matrica transformacije je jedinična, dok eulerovi kutevi iznose 0°,
- svi rezultati translacije bit će zapisani u milietrima, dok rotacije u stupnjevima,
- polja pozicija koja duž osi x, y i z imaju pogrešku veću od |5| mm s obzirom na referentnu nulu bit će označena crvenom bojom, u suprotnome zelenom, jednaka stvar će se učiniti i za grešku kutova veću od |2°| izraženu eulerovim vrijednostima.

Za olakšan uvid u rezultate testiranja u konačnici će se navesti postotak točnih, odnosno netočnih, pozicija finalne matrice transformacije te donijeti zaključak o utjecaju geometrije na preciznost.

Predmeti korišteni u eksperimentu ne smiju biti prozirni ili previše reflektirajući jer u suprutnome neće biti pravilno prikazani radi pogrešnog odbijanja infracrvenih zraka iz senzora.

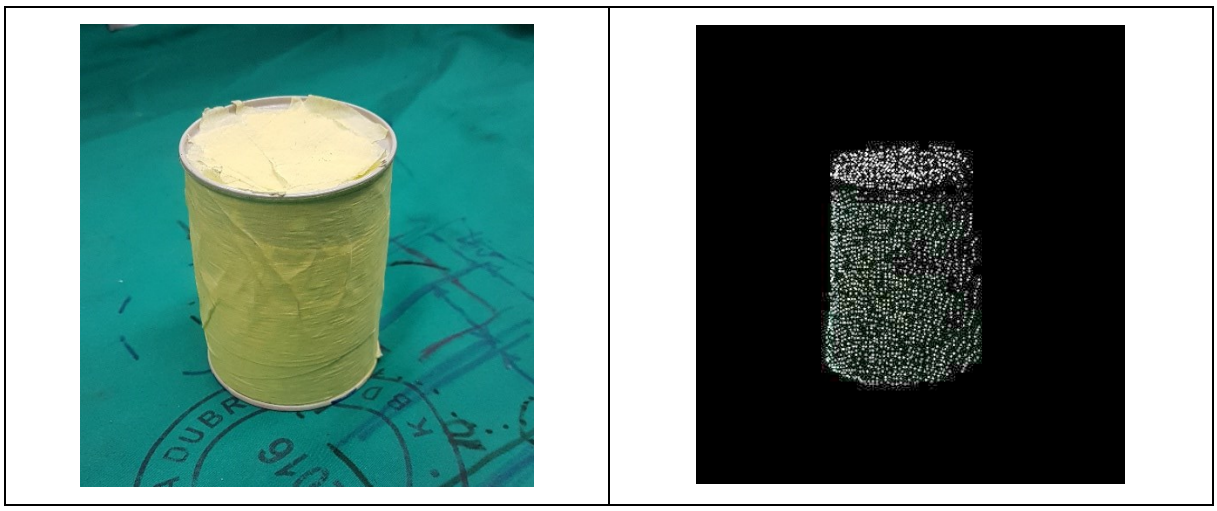
**Slika 6.1. Eksperimentalni postav**



**Slika 6.2. Fiksiranje kamere**

## 6.1 Objekt s više od tri ravnine simetrije

Objekt za ispitivanje koji ima više od tri ravnine simetrije prikazan je na donjim slikama.



**Slika 6.3. Fotografija i .pcd datoteka testiranog objekta**

**Tablica 19. Rezultati translacije za objekt s neograničenim brojem ravnine simetrije**

| Translacija (u mm) | | | | | | |
|---|---|---|---|---|---|---|
| | Inicijalna translacija (SHOT) | | | Konačna translacija (ICP ispravak) | | |
| Redni br. rezultata | X | Y | Z | X | Y | Z |
| 1 | 6 | 0 | 2 | 1 | 0 | 0 |
| 2 | 2 | -1 | 1 | 0 | 0 | 0 |
| 3 | 0 | 0 | 4 | 0 | 1 | 0 |
| 4 | -2 | -1 | 8 | 0 | 0 | 0 |
| 5 | 3 | 1 | 8 | 1 | 0 | 0 |
| 6 | -2 | 6 | -1 | 0 | 0 | 1 |
| 7 | 1 | 0 | 2 | 0 | 0 | 1 |
| 8 | -3 | -3 | 3 | 0 | 0 | 0 |
| 9 | 1 | 2 | 1 | 0 | 0 | 0 |
| 10 | 3 | 3 | 6 | 0 | 0 | 0 |
| | | | | | | |
| Pozitivni rezultati inicijalne translacije: | | | | 83,33% | | |
| Pozitivni rezultati konačne translacije: | | | | 100% | | |
| Negativni rezultati inicijalne translacije: | | | | 16,67% | | |
| Negativni rezultati konačne translacije: | | | | 0 | | |

U inicijalnoj translaciji dobiveni su poprilično precizni i konstantni rezultati što je pozitivna značajka za ICP registraciju koja je još dodatno ispravila rezultate. Pojavljivanje greške od 1 mm u konačnoj translaciji je zanemarivo, odnosno neće utjecati na konačnu funkcionalnost.

**Tablica 20. Rezultati rotacije za objekt s neograničenim brojem ravnine simetrije**

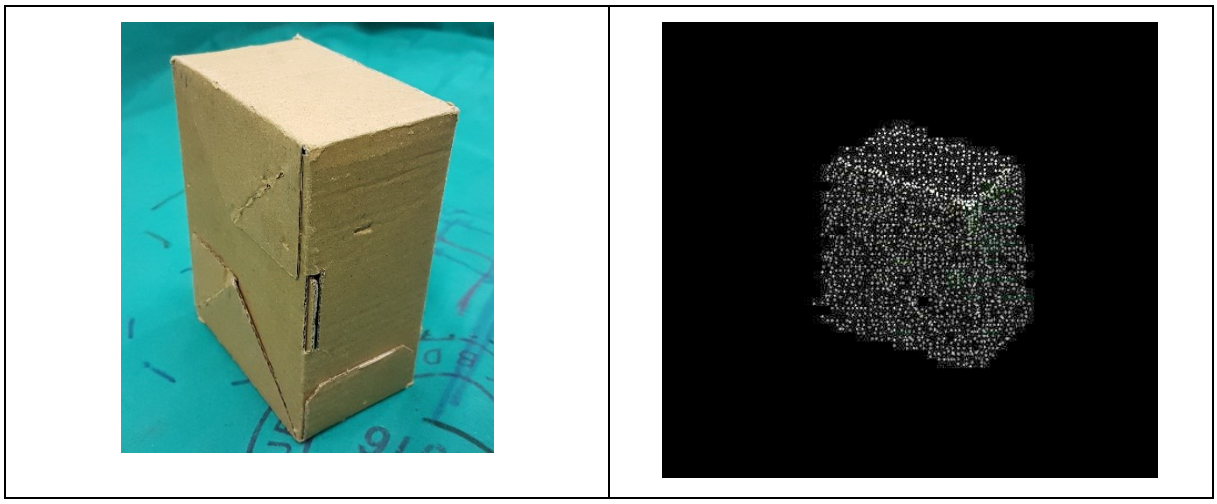
| Rotacija (u stupnjevima) | | | | | | |
|---|---|---|---|---|---|---|
| | Inicijalna Eulerova rotacija (SHOT) | | | Konačna Eulerova translacija (ICP ispravak) | | |
| Redni br. rezultata | X | Y | Z | X | Y | Z |
| 1 | 1,81 | -3,71 | 15,52 | 0,83 | -0,060 | 3,44 |
| 2 | 0,79 | -1,73 | -4,84 | 0,34 | -0,03 | -0,63 |
| 3 | 0,01 | -2,81 | 3,95 | 0,52 | 0,05 | -0,51 |
| 4 | -2,61 | 12,71 | -1,89 | 0,34 | 0,23 | -0,57 |
| 5 | 1,95 | 0,06 | 2,23 | 0,8 | 0 | -0,28 |
| 6 | -7,91 | 1,25 | 4,81 | 0,63 | 0,17 | -0,52 |
| 7 | 3,8 | 4,29 | -3,27 | 0,57 | 0 | -0,57 |
| 8 | 4,26 | -2,16 | 6,09 | 0,2 | 0 | -0,27 |
| 9 | -0,33 | 0,32 | 6,83 | 0,41 | -0,12 | 7,18 |
| 10 | 1,44 | 7,29 | -3,63 | 0,63 | 0,17 | -0,75 |
| | | | | | | |
| Pozitivni rezultati inicijalne rotacije: | | | 33% | | | |
| Pozitivni rezultati konačne rotacije: | | | 93,33% | | | |
| Negativni rezultati inicijalne rotacije: | | | 67% | | | |
| Negativni rezultati konačne rotacije: | | | 6,67% | | | |

Greška rotacije svakako je utjecajnija od greške translacije, međutim ICP algoritam gotovo u potpunosti ispravlja navedene greške. Pozitivno djelovanje ICP registracije moguće je zahvaljujući dobrim rezultatima inicijalne translacije. Iako su i dalje prisutne greške oko Z osi, tj. osi rotacije, one neće utjecati na funkcionalnost pošto se radi o cilindričnom objektu.

Lošija estimacija inicijalne rotacije prisutna je zbog relativno jednostavne geometrije promatranog objekta, za koju su duž oblaka izračunati deskriptori sličnih vrijednosti, čime se povećava mogućnost krivih korespondencija i kriva estimacija rotacije. Problem se može kontrolirati povećanjem kriterija vrednovanja deskriptora što bi doprinijelo točnijim korespodencijama, ali bi negativno utjecalo na uspješnost prepoznavanja zbog manjka istih.

## 6.2 Objekt s tri ravnine simetrije

Za razliku od prethodnog, sljedeće ispitani objekt je ograničen s tri ravnine simetrije i prikazan je na donjim slikama.



**Slika 6.4. Fotografija i .pcd datoteka ispitanog objeka**

**Tablica 21. Rezultati translacije za objekt s tri ravnine simetrije**

| Translacija (u mm) | | | | | | |
|---|---|---|---|---|---|---|
| | Inicijalna translacija (SHOT) | | | Konačna translacija (ICP ispravak) | | |
| Redni br. rezultata | X | Y | Z | X | Y | Z |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | -2 | 5 | 0 | 0 | 0 |
| 3 | 1 | -1 | 2 | 0 | 0 | 0 |
| 4 | 1 | -1 | 3 | 1 | 0 | 0 |
| 5 | 1 | -6 | -1 | 0 | 0 | 0 |
| 6 | 0 | -3 | 7 | 0 | 1 | 0 |
| 7 | 0 | 0 | 3 | 0 | 0 | 0 |
| 8 | 0 | 3 | 5 | 0 | 0 | 0 |
| 9 | 1 | -3 | 1 | 1 | 0 | 0 |
| 10 | 0 | -4 | 3 | 0 | 0 | 0 |
| | | | | | | |
| Pozitivni rezultati inicijalne translacije: | | | 86.7% | | | |
| Pozitivni rezultati konačne translacije: | | | 100% | | | |
| Negativni rezultati inicijalne translacije: | | | 13.3% | | | |
| Negativni rezultati konačne translacije: | | | 0% | | | |

Ponovno su postignuti vrlo dobri rezultati inicijalne translacije što rezultira u potpunosti ispravljenim rezultatima u konačnoj translaciji.

**Tablica 22. Rezultati rotacije za objekt s tri ravnine simetrije**

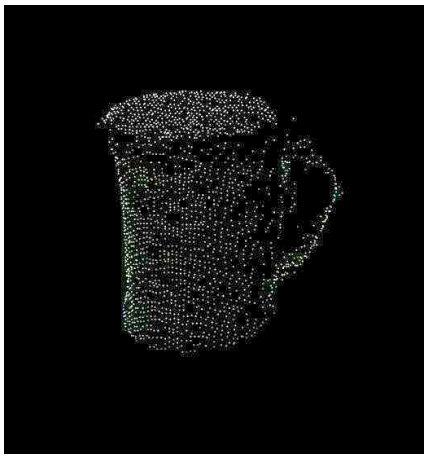
| Rotacija (u stupnjevima) | | | | | | |
|---|---|---|---|---|---|---|
| | Inicijalna Eulerova rotacija (SHOT) | | | Konačna Eulerova rotacija (ICP ispravak) | | |
| Redni br. rezultata | X | Y | Z | X | Y | Z |
| 1 | -2,22 | -3,95 | 1,19 | 0,23 | -0,63 | 0,23 |
| 2 | -2,46 | -4,91 | 2,82 | -1 | -0,11 | 0 |
| 3 | -3,54 | -3 | 2,98 | 0,057 | -0,46 | -0,11 |
| 4 | -0,055 | -0,8 | -3,86 | 0 | 0 | 0,17 |
| 5 | -3,4 | -1,50 | 6,07 | -0,11 | -0,63 | 0,34 |
| 6 | -1,71 | 2,57 | 2,27 | -0,17 | 0 | 0,23 |
| 7 | 2,43 | -2,04 | -1,55 | -0,06 | -0,46 | 0,27 |
| 8 | -0,11 | -3,33 | -3,76 | 0,23 | -0,46 | 0,23 |
| 9 | -0,74 | -0,05 | -0,69 | 0 | 0 | 0 |
| 10 | -2,075 | 1,64 | 2,74 | 0,11 | -0,52 | 0,06 |

| | |
|---|---|
| Pozitivni rezultati inicijalne rotacije: | 50% |
| Pozitivni rezultati konačne rotacije: | 100% |
| Negativni rezultati inicijalne rotacije: | 50% |
| Negativni rezultati konačne rotacije: | 0% |

Također i u ovome ispitivanju prisutnije su greške rotacije, no ipak u nešto manjem postotku nego kod prethodnog ispitivanja. Zahvaljujući boljim rezultatima ICP registracija je u potpunosti otklonila grešku.

## 6.3 Objekt s jednom ravnine simetrije

Sljedeće ispitani objekt sadrži jednu ravninu simetrije i prikazan je na donjim slikama.



**Slika 6.5. Fotografija i .pcd datoteka ispitanog objekta**

**Tablica 23. Rezultati translacije za objekt s jednom ravninom simetrije**

| Translacija (u mm) | | | | | | |
|---|---|---|---|---|---|---|
| | Inicijalna translacija (SHOT) | | | Konačna translacija (ICP ispravak) | | |
| Redni br. rezultata | X | Y | Z | X | Y | Z |
| 1 | -2 | -2 | 4 | 0 | 0 | 0 |
| 2 | -1 | -1 | 0 | 0 | 1 | 0 |
| 3 | 3 | -2 | 5 | 0 | 0 | 0 |
| 4 | 1 | -2 | 0 | 0 | 0 | 0 |
| 5 | 0 | -2 | 0 | 0 | 0 | 0 |
| 6 | -1 | -1 | -1 | 0 | 1 | 0 |
| 7 | 3 | -4 | -4 | 0 | 0 | 0 |
| 8 | -2 | 4 | 0 | 0 | 0 | 0 |
| 9 | -4 | 0 | -1 | 0 | 0 | 0 |
| 10 | -1 | 0 | 1 | 0 | 0 | 0 |
| | | | | | | |
| Pozitivni rezultati inicijalne translacije: | | | 96,67% | | | |
| Pozitivni rezultati konačne translacije: | | | 100% | | | |
| Negativni rezultati inicijalne translacije: | | | 3,33% | | | |
| Negativni rezultati konačne translacije: | | | 0% | | | |

Za navedeni objekt već u inicijalnoj translaciji postignuti su vrlo precizni rezultati tijekom svih 10 iteracija, a ICP djelovanjem greška je gotovo u potpunosti eliminirana.

**Tablica 24. Rezultati rotacije za objekt s jednom ravninom simetrije**

| Rotacija (u stupnjevima) | | | | | | |
|---|---|---|---|---|---|---|
| | Inicijalna Eulerova rotacija (SHOT) | | | Konačna Eulerova translacija (ICP ispravak) | | |
| Redni br. rezultata | X | Y | Z | X | Y | Z |
| 1 | 3,25 | -1,83 | 1,77 | 0 | -0,11 | 0,29 |
| 2 | 0,69 | -0,23 | 0,46 | -0,23 | -0,12 | 0,23 |
| 3 | -0,79 | -0,59 | -2,41 | -0,17 | -0,06 | -0,11 |
| 4 | -1,71 | -2,03 | 1,82 | 0,34 | -0,06 | -0,17 |
| 5 | -12,45 | 2,63 | -1,57 | 0,11 | 0 | -0,29 |
| 6 | -1,54 | 1,21 | 8,06 | -0,11 | 0,057 | -0,11 |
| 7 | -3,96 | 1,56 | 0 | 0 | 0 | 0 |
| 8 | -5,42 | 0,24 | 7,50 | -0,06 | -0,17 | 0,11 |
| 9 | 1,89 | 1,6 | -0,46 | 0 | 0 | 0 |
| 10 | -2,7 | -3,71 | 6,29 | 0 | 0 | 0 |

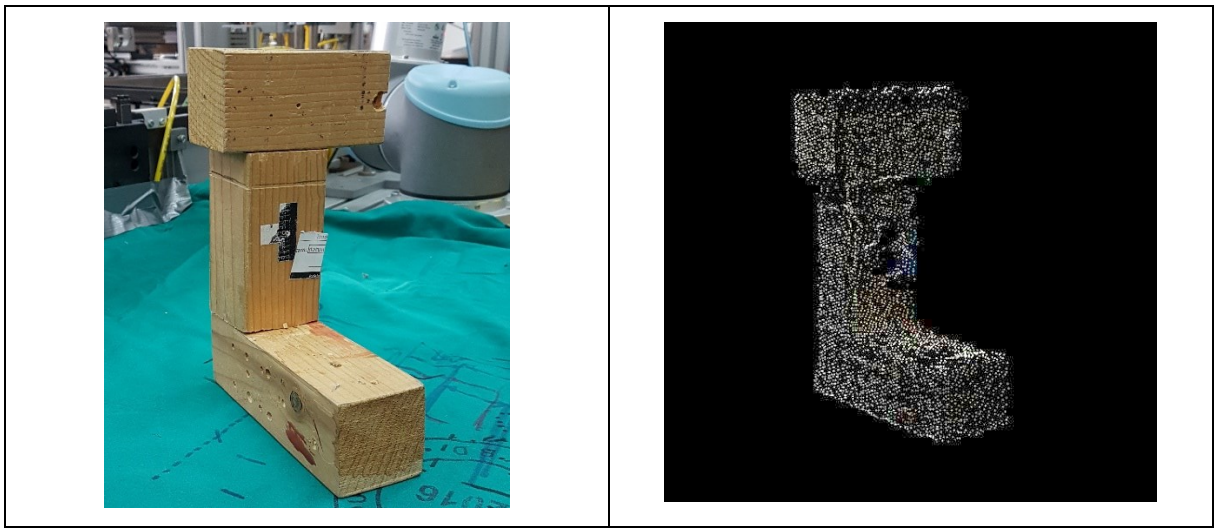
| | |
|---|---|
| Pozitivni rezultati inicijalne rotacije: | 60% |
| Pozitivni rezultati konačne rotacije: | 100% |
| Negativni rezultati inicijalne rotacije: | 40% |
| Negativni rezultati konačne rotacije: | 0% |

U ovome testu dobiven je veći postotak pozitivnih inicijalnih rotacija nego u prethodnim slučajevima, međutim mogu se uočiti pojedine ekstremne vrijednosti, poput 12.45° ili 8.06°, koje bi narušile funkcionalnost praktične primjene kada bi se informacije poslale robotu. U ovakvim situacija može se primjetiti važnost ICP finog pozicioniranja nakon čijeg djelovanja je greška svedena u svim ispitivanjima za sve osi ispod jednog stupnja.

Pretpostavka je da su veća odstupanja rotacije nastale zbog specifičnog detalja na ispitanom objektu poput drške za prihvat (slika 6.5). Uočeno je da je dolazilo do vidljive nekonzistentnosti točaka oko navedenog područja što možebitno djeluje na pogrešnu estimaciju rotacije.

## 6.4 Objekt bez ravnine simetrije

U konačnici ispitat će se objekt koji nema niti jednu ravninu simetrije prikazan na slikama ispod.



**Slika 6.6. Fotografija i .pcd datoteka ispitanog objekta**

**Tablica 25. Rezultati translacije za objekt bez ravnine simetrije**

| Translacija (u mm) | | | | | | |
|---|---|---|---|---|---|---|
| | Inicijalna translacija (SHOT) | | | Konačna translacija (ICP ispravak) | | |
| Redni br. rezultata | X | Y | Z | X | Y | Z |
| 1 | 1 | 0 | -3 | 0 | 0 | 0 |
| 2 | -3 | -3 | -1 | 0 | 0 | 0 |
| 3 | 1 | -4 | 2 | 0 | 0 | 0 |
| 4 | 4 | 6 | -1 | 0 | 1 | 0 |
| 5 | -1 | 1 | 4 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 3 | -3 | 0 | 0 | 0 | 0 |
| 8 | 0 | 1 | -1 | 0 | 0 | 0 |
| 9 | 0 | -3 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | | | | |
| Pozitivni rezultati inicijalne translacije: | | | 96,67% | | | |
| Pozitivni rezultati konačne translacije: | | | 100% | | | |
| Negativni rezultati inicijalne translacije: | | | 3,33% | | | |
| Negativni rezultati konačne translacije: | | | 0% | | | |

Kao i kod objekta s jednom osi simetrije opet su postignuti vrlo precizni rezultati inicijalne translacije koji su u potpunosti ispravljeni ICP registracijom

**Tablica 26. Rezultati rotacije za objekt bez ravnine simetrije**

| Rotacija (u stupnjevima) | | | | | | |
|---|---|---|---|---|---|---|
| | Inicijalna Eulerova rotacija (SHOT) | | | Konačna Eulerova translacija (ICP ispravak) | | |
| Redni br. rezultata | X | Y | Z | X | Y | Z |
| 1 | -0,05 | 2,29 | 1,23 | 0 | 0 | 0 |
| 2 | 0,16 | 0,87 | 5,16 | 0,11 | 0,11 | 0,11 |
| 3 | -1,27 | -4,18 | -1,77 | 0,29 | -0,29 | 0,06 |
| 4 | -2,94 | 0,45 | 1,82 | 0 | 0 | 0 |
| 5 | 3,73 | -0,11 | -5,28 | 0,4 | -0,17 | -0,23 |
| 6 | 2,44 | -1,64 | -1,19 | 0,53 | -0,23 | -0,15 |
| 7 | -0,80 | 0,69 | -0,02 | 0 | 0 | 0 |
| 8 | -0,55 | -0,34 | -1,83 | -0,17 | 0,06 | -0,11 |
| 9 | 1,32 | 0,06 | -0,12 | 0 | 0 | 0 |
| 10 | 0,76 | -1,48 | 1,18 | 0,29 | -0,17 | -0,17 |
| | | | | | | |
| Pozitivni rezultati inicijalne rotacije: | | | 76,67% | | | |
| Pozitivni rezultati konačne rotacije: | | | 100% | | | |
| Negativni rezultati inicijalne rotacije: | | | 23,33% | | | |
| Negativni rezultati konačne rotacije: | | | 0% | | | |

Također kao i kod prethodno ispitanih objekata više grešaka je prisutno prilikom estimacije rotacije, međutim ona je u potpunosti ispravljena ICP poravnanjem.

U suprotnosti s prvim primjerom gdje su rezultati rotacije dobiveni sparivanjem deskriptora prilično netočni, u ovom primjeru je postignuto najviše pozitivnih rezultata inicijalne rotacije. Izostanak ravnina simetrije rezultiralo je većem broju karakterističnih deskriptora i njihovom točnijem sparivanju, a samim time i točnijom estimacijom translacije i rotacije.

# 7. ZAKLJUČAK

U okviru diplomskog zadatka primjenjena su dva pristupa prepoznavanja i prostorne lokalizacije objekata na sceni koristeći knjižnicu otvorenog koda Point Cloud Library te 3D senzor Kinect v2. Prvi i jednostavniji pristup temelji se na ekstrakciji klastera koji postaju volumeni od interesa, dok drugi, složeniji način, prikuplja informacije o lokalnoj geometriji modela i scene, a zatim na temelju usporedbe traži korespondencije, odnosno sličnosti između dva oblaka točaka. Rezultati prostorne pozicije prilagođavaju se u zapis namijenjen UR5 robotskoj ruci te se šalju putem TCP/IP konekcije. Ostatak programiranja izvršava se na privjesku za učenje gdje se definiraju inicijalne postavke robota, redoslijed izvođenja programa i preuzete numeričke vrijednosti sa servera pretvaraju se u naredbe gibanja.

Unatoč prevelikim ograničenjima metode ekstrakcije klastera ona nije daljnje razmatrana u eksperimentalnom niti praktičnom dijelu. Bez obzira na vrlo brzo izvođenje i stabilne rezultate nedostaci poput klasifikacije dva objekta kao jedan, ukoliko se predmeti nalaze suviše blizu, te nemogućnost referenciranja na određeni objekt bili su dovoljno veliki da naruše funkcionalnost praktične primjene na robotu.

Kako bi se navedena ograničenja primjenjena je metoda lokalnih značajki koja se pokazala pouzdanom i konzistentnom s vrlo preciznim rezultatima, posebice nakon djelovanja ICP registracije. Međutim, za pravilan rad potrebno je točno postavljanje niza parametara što zahtjeva poznavanje teorije prije uporabe.

Rezultati ispitivanja na više različtih objekata pokazali su točnije rezultate prostorne pozicije ukoliko je promatrani predmet kompleksnije geometrije. Takva pojava uzrokovana je unikatnijim vrijednostima deskriptora oko specifičnih detalja iz čega proizlaze točnije korespondencije, a samim time i točnija estimacija pozicije. Unatoč tome, dobiveni rezultati su dovoljno precizni, s vrlo malom greškom, i kod jednostavnijih predmeta čime se ne ograničava funkcionalna uporaba na robotu.Point Cloud Library pokazao se kao vrlo složen alat, s velikom količinom organiziranih i kompleksnih algoritama za obradu i analizu 3D podataka, čija uporaba zahtjeva poznavanje C++ programskog jezika.

Pošto Kinect v2 snima podatke niže do srednje kvalitete, gdje se nerijetko može uočiti rasipanje podataka, daljnje ispitivanje bi zahtjevalo isprobavanje algoritma na preciznijem senzoru,

također analizu ostalih lokalnih dekskriptora i ključnih točki te implementaciju globalnih deskriptora.

Ovakvi algoritmi osim za industrijskog robota mogu biti primjenjivi i na drugim vrstama robota, poput humanoidnih ili uslužnih, također i na autonomnim vozilima za analizu okoline, no takve aplikacije zahtjevaju integraciju veće baze znanja i optimiranje brzine izvođenja procesa.

# LITERATURA

[1] *Point Cloud Library,* http://pointclouds.org/, zadnje pristupljeno: Listopad 2018.

[2] R. Bogdan Rusu and S. Cousins, *3D is here: Point Cloud Library (PCL)*, Menlo Park, SAD, 2011.

[3] F. Alsadat Estiri, *3D Object Detection and Tracking Based Based On Point Cloud Library*, Tampere university of technology, Tampere, Finland, 2014.

[4] *CMake*, www.cmake.com, Zadnje pristupljeno: Listopad 2018.

[5] *Unanancyowen*, http://unanancyowen.com/en/pcl-kinectv2-with-grabber/, zadnje pristupljeno: Listopad 2018.

[6] *Visual Studio*, https://code.visualstudio.com/, zadnje pristupljeno: Listopad 2018.

[7] L. Caruso, R. Russo, S. Savino, *Microsoft Kinect V2 vision system in manufacturing application*, University of Naples Federico II, Naples, Italy, 2017.

[8] P. Fankhauser, M. Bloesch, D. Rodriguez, R. Kaestner, M. Hutter, R. Siegwart, Kinect v2 for Mobile Robot Navigation: Evaluation and Modeling, Techische Universitat Munchen, Germany, 2014.

[9] J. Sell and P. O'Connor, *The Xbox One System on a Chip and Kinect Sensor*, Microsoft, USA, 2014.

[10] M. Crneković, A. Jokić, *Robotika – prezentacije*, Fakultet strojarstva i brodogradnje, Zagreb, 2018.

[11] S. Briot, W. Khalil, *Dynamics of Parallel Robots*, Springer International Publisher, 2015.

[12] *Stackexchange*, https://stats.stackexchange.com/questions/288669/the-algorithm-behind pclstatisticaloutlierremoval, zadnje Pristupljeno: Listopad 2018.

[13] *Point Cloud Library,* http://pointclouds.org/documentation/tutorials/normal_estimation.php, zadnje pristupljeno: Listopad 2018.

[14] B. Steder, R. R. Bogdan, K. Konolige, W. Burgard, *Point feature extraction on 3D range scans taking into account object boundaries*, IEEE International Conference on Robotics and Automation, 2011.

[15] Silvio Filipe and Luis A. Alexandre, *A Comparative Evaluation of 3D Keypoint Detectors in a RGB-D Object Dataset*, Instituto de Telecomunicacoes, Portugal, 2014.

[16] V. Ghorpade, L. Trassoudaine, P. Checchin, *Performance Evaluation od 3D Keypoint Detectors For Time-Of-Flight Depth Data*, Institut Pascal, France, 2016.

[17] K. Berker Logoglu, *Spatial 3D local descriptors for object recognition in RGB-D images*, Informatic Institute of Middle east techical university, 2016.

[18] *Robotica*, http://robotica.unileon.es/index.php/PCL/OpenNI_tutorial_4:_3D_object_recognition_(descriptors), zadnje pristupljeno: Listopad 2018.

[19] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, J. Zhang, *Performance Ecaluation of 3D Local Feature Descriptors*, The University of Western Australia, 2014.

[20] Q. Zhang, L. Kong, J. Zhao, *Real-Time General Object Recognition for Indoor Robot based on PCL*, IEEE International Conference on Robotics and Biomimetics, China, 2013.

[21] R. Hansch, T. Weber, O. Hellwich, *Comparison od 3D Interest Point Detectors and Descriptors for Point Cloud Fusion,* 2016.

[22] *Robotica*, Zadnje pristupljeno: Listopad 2018.

[23] B. Jerbić, F. Šuligoj, M. Švaco, B. Šekoranja, *Robot Assisted 3D Point Cloud Object Registration*, 2014.

[24] http://pointclouds.org/documentation/tutorials/pairwise_incremental_registration.php Zadnje pristupljeno: Listopad 2018.

[25] *Point Cloud Library*, http://pointclouds.org/documentation/tutorials/normal_estimation.php, zadnje pristupljeno: Listopad 2018.

[26] https://sites.google.com/site/3dkeypoints/results---set-5 Zadnje pristupljeno: Listopad 2018.

[27] https://www.universal-robots.com/media/50588/ur5_en.pdf Zadnje pristupljeno: Listopad 2018.

[28] Nikola Kirić, *Primjena 3D vizijskog sustava za praćenje objekata robotom u realnom vremenu*, Fakultet strojarstva i brodogradnje, Zagreb, 2015.

# PRILOZI

I. CD-R disc