SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 694

Detekcija i prepoznavanje objekata u svrhu izlučivanja točaka hvata i generiranja podražaja za sučelje mozak-računalo

Denis Štogl

Zagreb, lipanj 2013.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Zahvaljujem menotoru prof. dr. sc. Ivanu Petroviću na vođenju i potpori tijekom izrade rada.

Posebna zahvala dipl. ing. Srećku Jurić-Kavelju na izdvojenom vremenu, strpljivosti i svom prenešenom znanju koje je omogućilo realizaciju ovog rada.

SADRŽAJ

Popis slika **vi**

1. Uvod **1**

2. Pregled dosadašnjih istraživanja **3**

2.1. ROS - Operacijski sustav za robote 6

3. Metode za detekciju i prepoznavanje objekata u programskom sustavu *ROS* **8**

3.1. Tabletop Object Detector 8

3.2. Object Recognition Kitchen (ORK) 9

3.2.1. ORK: *Line-mod* 11

3.2.2. ORK: *Tabletop Object Recognition* 12

3.2.3. ORK: *Textured Object Detection* 13

3.2.4. ORK: *Transparent objects* 13

4. Metode za izlučivanje točaka hvata i manipulacija objektima u programskom sustavu *ROS* **14**

4.1. Metode određivanja hvata 14

4.2. Manipulacija objektima 16

5. Upotreba programskog sustava *ROS* za detekciju, prepoznavanje i manipulaciju objektima **17**

5.1. Oprema korištena prilikom izvođenja eksperimenata 17

5.1.1. Senzor Microsoft Xbox 360 Kinect 17

5.1.2. Ruka Schunk Powerball i hvataljka WSG-50 20

5.2. Manipulacijski cjevovod u simuliranom okruženju 22

5.2.1. Segmentacija prostora, detekcija i prepoznavanje objekata . . 25

5.2.2. Izlučivanje točaka hvata i manipulacija objektima 25

5.3. Segmentiranje i prepoznavanje objekata u stvarnom okruženju 28
5.3.1. Tabletop Object Detector 28
5.3.2. Object Recognition Kitchen (ORK) 30

6. Generiranje podražaja za sučelje mozak-računalo **37**
6.1. ROS čvor: generator SSVEP podražaja nad predmetima u sceni 39
6.2. ROS čvor: prikaz detektiranih objekata 39

7. Manipulacija objektima pomoću sučelja mozak-računalo **42**

8. Zaključak **44**

Literatura **45**

A. Popis repozitorija **53**

POPIS SLIKA

3.1. Tabletop Object Detector cjevovod 10
3.2. Line-mod primjer, izvor: Hinterstoisser et al. (2011) 12

4.1. Arhitektura cjevovoda za manipulaciju objektima, izvor: Jerbić et al. (2013) 16

5.1. Xbox 360 Kinect uređaj, izvor: internet 18
5.2. Povezivanje podataka RGB kamere i senzora dubine 19
5.3. Problem detekcije prozirnih objekata 21
5.4. Ruka Schunk Powerball s hvataljkom WSG-50, izvor: internet 22
5.5. Object Manipulation cjevovod, izvor: Jerbić et al. (2013) 24
5.6. Simulacijsko okruženje 24
5.7. Prikaz podataka Kinect senzora tijekom simulacije 26
5.8. Rezultat segmentacije scene 27
5.9. Hvatanje i podizanje predmeta u simuliranom okruženju 28
5.10. Prikaz podataka Kinect senzora sa stvarne scene 29
5.11. Rezultat segmentacije scene 30
5.12. Točkasti uzorak za snimanje podataka 31
5.13. Prozori pri snimanju objekata ORK sustavom 33
5.14. Objekti snimljeni ORK sustavom 35

6.1. SSVEP stimulacija sa šahovnicom, izvor: internet 37
6.2. P300 matrica slova, izvor: internet 38
6.3. Generator SSVEP podražaja nad predmetima 40
6.4. Sučelje za prikaz izlučenih objekata 41

7.1. Prijedlog upotrebe BCI sustava za pomoć pri manipulaciji osobama s invaliditetom, izvor: Jerbić et al. (2013) 43

1. Uvod

Detekcija i prepoznavanje objekta u svrhu izlučivanja točaka hvata danas je istraživački vrlo aktivno područje. Neka od vrlo zanimljivih područja primjene prepoznavanja objekata u robotici su roboti za čovječanstvo (engl. *Robots for humanity*) te osobni roboti (engl. *Personal robots*). U oba slučaja primjena robota je vrlo slična jer je glavni cilj razviti robota koji može zamijeniti čovjeka u ulozi osobnog asistenta odnosno razviti robota koji može potpuno autonomno djelovati u prostoru prilagođenom ljudima i surađivati s ljudima. Velikom broju istraživanja pridonio je i razvoj povoljnih senzora za 3D detekciju prostora, kao što je *Microsoft Xbox 360 Kinect*, te rast računske moći računala. Uz veliki broj istraživanja razvijen je i velik broj metoda od kojih se neke baziraju samo na 2D (Willow Garage, 2013h) ili 3D podacima (Muja i Ciocarlie, 2013) dok druge kombiniraju te podatke (Hinterstoisser et al., 2011).

Drugo vrlo značajno područje istraživanja su komunikacijska sučelja između čovjeka i robota odnosno čovjeka i računala. Jedno od njih je sučelje mozak-računalo (engl. *Brain-computer interface (BCI)*) koje pruža alternativni način komunikacije i upravljanja računalom ne koristeći klasične neuro-motoričke komunikacijske puteve. Ova tehnologija ima mnogo mogućih primjena kao što je igranje igara bez upotrebe ruku, rehabilitacija, detekcija kognitivnog napora, detekcija stresa i drugo. Jedna od najizazovnijih primjena BCI-a je povratak mogućnosti komunikacije i manipulacije osobama s težim oblicima invaliditeta. Za tu namjenu razvijeno je više metoda od kojih su neke invazivne, kao što je na primjer upotreba mikroelektroda u mozgu kao sučelje za upravljanje protezom (Collinger et al., 2012), dok su druge manje invazivne i baziraju se na elektroencelofalografiji (engl. *Electroencephalography (EEG)*) (McFarland et al., 2010).

Ovaj rad bavi se, u prvom redu, pregledom i mogućnošću upotrebe već implementiranih metoda za prepoznavanje objekata i izlučivanje točaka hvata u operacijskom sustavu za robote (engl. *Robot Operating System (ROS)*) (poglavlja 3, 4 i 5). U poglavlju 5 razmatra se i izvođenje kompletnog cjevovoda za manipulaciju objektima (engl. *manipulation pipeline*) koristeći dostupan senzor i robotsku ruku s hvataljkom. Ovdje je

prezentiran i jedan od najznačajnijih rezultata, a to je uspješno hvatanje i podizanje objekta u simulacijskom okruženju na temelju razmatranih metoda za detekciju i generiranje točaka hvata. Drugi važan dio je implementacija grafičkog sučelja za generiranje BCI podražaja koje se temelji na detektiranim objektima u sceni (poglavlje 6). Pri tome se uzimaju u obzir svi zahtjevi na takvo sučelje koji su dostupni iz ranijih istraživanja. Na samom kraju, u poglavlju 7, razmatra se mogućnost povezivanja navedenih dijelova rada u svrhu povećanja razine manipulacije i samostalnosti osoba s invaliditetom.

2. Pregled dosadašnjih istraživanja

Problematika detekcije i prepoznavanja objekata intenzivno se istražuje već 15-ak godina i to u različitim područjima, kao što su računalna grafika, robotika i raspoznavanje uzoraka. Najzanimljivija područja primjene u robotici su osobni roboti (Klank et al., 2009) i robotika za čovječanstvo (Chen et al., 2013) za koje je iznimno važan sustav percepcije i praćenja objekata u stvarnom vremenu, kao i sustav za određivanje najboljih točaka hvata. Prema Hinterstoisser et al. (2011) mogu se razlikovati dvije osnovne skupine metoda prepoznavanja 3D objekata: one korištene na slikama intenziteta (engl. *intensity images*) (kao što je RGB slika) i one koje rade s podacima o daljini (3D oblak točaka) odnosno sa slikama koje sadrže podatke o dubini (engl. *depth images*).

Metode koje se baziraju na obradi slika intenziteta mogu se rasporediti u dvije glavne kategorije:

metode zasnovane na učenju (engl. *Learning Based Methods*) koje obično zahtijevaju veliku količinu podataka za učenje i samim time vrlo dugu fazu (ofline) učenja, i

metode zasnovane na uspoređivanju predložaka (engl. *Template Matching Methods*) koje su imale vrlo značajnu ulogu u praćenju-po-otkrivanju (engl. *tracking-by-detection*) aplikacijama.

Prednost prvih metoda je to što mogu prepoznati objekte iz različitih kuteva gledišta s velikim točnošću zbog čega im je i područje primjene široko, od algoritama za prepoznavanje lica do raspoznavanja 3D objekata. Druge metode daju bolje rezultate i lakše se primjenjuju na objekte bez, ili s vrlo malo, teksture s obzirom na metode koje se oslanjaju na različite značajke objekta[1]. Ali iako vrlo robusne, metode zasnovane na usporedbi predložaka nisu primjenjive u aplikacijama gdje je potreban rad u stvarnom vremenu jer broj predložaka za usporedbu naglo raste kada se želi pokriti

[1] Značajke objekata su najčešće dobivene iz teksture površine.

veliki broj različitih točaka (kuteva) gledišta. Više o ova dva pristupa i korištenju slika koje bilježe intenzitet za prepoznavanje objekata, kao i mnoge reference koje se bave tom tematikom mogu se pronaći u Hinterstoisser et al. (2011) u poglavlju „Pregled istraživanja“.

Veliki broj metoda koje se koriste u robotici pripada drugoj skupini algoritama, odnosno, zasnivaju se na obradi 3D oblaka točaka ili slika dubine iz kojih je, za svaku točku prostora, poznat položaj u koordinatnom sustavu senzora. Do prije nekoliko godina za dobivanje takvih podataka korstio se laserski senzor s pan jedinicom, a tada se javljaju stereo kamere s teksturiranim ili strukturiranim svjeltom u vidiljivom ili infra-crvenom području[2]. Prednost stereo senzora u odnosu na laser je velika brzina skeniranja prostora, 30-60 Hz naprema 0.2 Hz sekundi kod lasera s pan jedinicom, ali na račun veće količine šuma (Rusu et al., 2009b).

Neki od najpoznatijih pristupa za obradu podataka iz 3D oblaka točaka su: pomoću sferičnih harmonijskih invarijanti (engl. *spherical harmonic invariants*) (Burel i Henocq, 1995), rotacijskih slika (engl. *spin images*) (Johnson i Hebert, 1999), mapa zaobljenosti (engl. *curvature maps*) (Gatzke et al., 2005) te brzih histograma za određivanje značajki točaka (engl. *Fast Point Feature Histograms (FPFH)*) (Rusu et al., 2009a). Prvim dvijema spomenutim metodama karakteristike degradiraju ako se primjenjuju nad zašumljenim i rijetkim oblakom[3] kao što su podaci sa stereo senzora, ali zato se varijanata FPFH metode pokazala vrlo dobrom u takvim slučajevima što je pokazano u Rusu et al. (2009b).

Uz navedene metode koje većinom koriste teksturu objekata razvijale su se i metode za detekciju objekata bez texture (Hinterstoisser et al., 2010, 2012b; Holzer et al., 2009) te vrlo robusne metode zasnovane na detekciji prirodnih 3D obilježja (engl. *Natural 3D Markers (N3Ms)*) (Hinterstoisser et al., 2007). Zajedničko svim ovim metoda je korak treninga nad snimljenim podacima kada se izlučuju dominantne značajke koje će kasnije pomoći pri detekciji i određivanju položaja stvarnih objekta u stvarnom vremenu.

Osim metoda detekcije objekata i određivanja njihovih značajki razvijani su i algoritmi koji su namijenjeni brzoj i točnoj detekciji i prepoznavanju objekta u svrhu izlučivanja točaka hvata i manipulacije objektom. Za ovakve zadaće potrebno je identificirati i prepoznati potporne ravnine i šest-dimenzionalni položaj objekta. Jedan od zanimljivijih pristupa predstavljen je u Klank et al. (2009) u kojem se korisi 3D sustav percepcije koji može robusno pronaći pripadajući CAD model objekta iako je stol

[2]U ovu vrstu senzora spada i *Microsoft Xbox 360 Kinect* korišten u ovom radu (poglavlje 5.1.1)

[3]3D oblak točaka je rijedak u odnosu na mjerenja lasera s pan jedinicom.

u sceni napunjen s mnogo različitih objekta (engl. *cluttered scene*). Njihova metoda za segmentaciju scene i izlučivanje klastera objekata koristi dva neovisna izvora podataka odnosno dvije vrste kamera: RGB kameru i kameru baziranu na vremenu leta (engl. *Time-Of-Flight (TOF) camera*). U Collet Romea et al. (2009) predstavljena je metoda za izgradnju 3D modela objekta koristeći nekoliko slika, a za to je korištena zanimljiva kombinacija RANSAC (Fischler i Bolles, 1980) i Mean Shift (Cheng, 1995) algoritma za prepoznavanje različitih instanci istog objekta, a time i njegovog položaja i orijentacije. Jedan od najvećih problema kod točnog određivanja šest-dimenzionalnog položaja objekta je određivanje orijentacije. U Glover et al. (2011) razmatra se upravo taj problem gdje se predlaže nova vjerojatnosna metoda koja je mnogo brža od dotadašnjih.

S obzirom na to da su kod različitih skupina objekata važne različite značajke niti jedna metoda za detekciju i prepoznavanje objekata se nije pokazala dovoljno dobra za sve skupine objekata. Zbog toga se u Muja et al. (2011) predlaže infrastruktura za prepoznavanje (engl. *Recognition Infrastructure (RAIN)*) koja omogućava kombiniranje više različitih 2D i 3D metoda za prepoznavanje i estimaciju položaja objekta. Implementirane metode se mogu izvršavati serijski ili paralelno, a čak je moguća i dinamička promjena parametara pojedine metode. Ta infrastruktura se danas više ne koristi (što je naznačeno u Muja et al. (2013)) nego je zamjenjena novim ECTO sustavom (Willow Garage, 2013a) koji je baza nove infrastrukture za prepoznavanje objekata pod nazivom *„Object Recognition Kitchen“*[4].

Do sada su predstavljena neka od istraživanja koja se većinom bave detekcijom i prepoznavanjem objekata, a u nastavku će biti predstavljene moderne metode određivanja hvata. U Goldfeder et al. (2009) predstavljena je metoda koja omogućava hvatanje objekata za koje su poznati samo dijelomični 3D podaci. Iako metoda koristi unaprijed izračunate hvatove, omogućava svojevrsnu ekstrapolaciju hvatova na objekte koji su slični objektima spremljenima u bazu. Mnoge metode za određivanje hvatova objekata baziraju na prepoznavanju objekata i dohvaćaju unaprijed generiranih hvatova iz baze, ali sa stanovišta hvatanja i manipulacije objekata to nije nužno te su razvijene i metode koje iz segmentiranih oblaka točaka određuju hvat objekta. Jedna od takvih metoda predstavljena je u Holz et al. (2011) i bazira se na detekciji ravnina iz oblaka točaka. Slijedeće zanimljivo istraživanje je Dogar et al. (2012) gdje je predstavljena metoda koja dopušta višestruke interakcije između robota i objekta, tj. dopušta robotu da pomiče pomične prepreke i time si oslobodi put do željenog objekta te ostvari hvat

[4]Ovaj sustav se koristi i u ovom radu te je detaljnije opisan u poglavlju 3.2.

koji ranije nije bio ostvariv. Ova metoda se pokazala u praksi uspješnijom u pretrpanim scenama u odnosu na metode koje ne dopuštaju interakciju s predmetima koji nisu prvotan cilj manipulacije. Vrlo važno istraživanje za ovaj rad je Ciocarlie et al. (2010) koji prezentira cjelovitu programsku arhitekturu za detekciju, prepoznavanje, hvatanje i manipulaciju objektima u kućanstvima te se upravo verzija tog cjevovoda koristi i u ovom radu (detaljniji opis dijelova cjevovoda proteže se kroz poglavlja 3 i 4, a detalji o upotrebi se mogu naći u poglavlju 5).

Razvojem metoda za detekciju i prepoznavanje uočeno je da se neki algoritmi vrlo često koriste te je razvijena biblioteka za rad s oblacima točaka *Point Cloud Library (PCL)* (PCL, 2013) i biblioteka za rad s 2D slikama *Open Computer Vision Library (OpenCV)* (Itseez, 2013) koje implementiraju često korištene algoritme i strukture podatka te time olakšavaju rad i obradu 3D oblaka točaka i 2D slika. Obje ove biblioteke su otvorenog koda, a u njihovom održavanju i unaprjeđenju sudjeluju neka od vodećih svjetskih sveučilišta i tvrtki. Detalji o implementaciji te kompletna dokumentacija ovih biblioteka može se naći na službenim stranicama projekata (PCL (PCL, 2013), OpenCV (Itseez, 2013)) ili u radovima kao što su na primjer Rusu i Cousins (2011) te Bradski i Kaehler (2008).

U ovom poglavlju su navedena i ukratko opisana neka od istraživanja koja se bave problematikom detekcije, prepoznavanje i hvatanja objekata. Uz ova istraživanja postoje još mnoga druga zanimljiva istraživanja, ali sva ovdje odabrana istraživanja su izravno ili neizravno povezana s programskim paketom ROS te su zbog toga izdvojena. U slijedećem poglavlju može se naći kratki opis *Operacijskog sustava za robote* - ROS, a nešto više o metodama implementiranim u ROS-u i njihovom korištenju u poglavljima nakon toga.

2.1. ROS - Operacijski sustav za robote

Operacijski sustav za robote (engl. *Robot operating system (ROS)*) je meta operacijski sustav otvorenog koda koji je primarno namjenjen za instalaciju na robotskim sustavima. U ROS-u su implementirane mnoge bibliteke i alati koji pomažu programerima i inženjerima pri izradi robotskih aplikacija. Kao i pravi operacijski sustav, ROS osigurava apstrakciju sklopovlja, kontrolu uređaja na niskoj razini, proslijeđivanje poruka između različitih procesa, upravljanje paketima i drugo. *ROS* je prvenstveno namijenjen upotrebi na operacijskom sustavu *Ubuntu GNU/Linux*, ali ga je moguće pokretati i na drugim *Unix* distribucijama. Iako je *ROS* razvila istraživačka grupa *Willow Garage* iz Sjedinjenih Američkih Država danas se koristi širom svijeta te njegovom razvoju

doprinose mnogi pojedinci i istraživačke grupe čime on postaje jedan od vodećih programskih sustava za upravljanje robotima što se očituje u dostignućima i broju istraživanja koja koriste ovaj sustav. Više o *ROS*-u može se naći na službenim stranicma projekta (Quigley i Willow Garage, 2013).

3. Metode za detekciju i prepoznavanje objekata u programskom sustavu *ROS*

U programskom sustavu *ROS* postoji mogućnost korištenja dva različita sustava za detekciju i prepoznavanje objekata. Jedan od njih je cjevovod podataka koji se nalazi u paketu *Tabletop Object Detector* (Muja i Ciocarlie, 2013) te on može provesti segmentaciju i jednostavno prepoznavanje objekata u sceni[1]. Drugi sustav je već spomenut u poglavlju 2 kao nasljednik REIN sustava (Muja et al., 2013) te je to zapravo implementacija infrastrukture namijenjene za jednostavnu izradu i simultano izvođenje nekoliko različitih metoda za prepoznavanje objekata, a nalazi se u *Object Recognition* stogu (Rublee et al., 2013). Puno ime ovog sustava je *Object Recognition Kitchen (ORK)* te će se ono koristiti u nastavku ovog rada.

U nastavku su detaljnije opisana oba sustava.

3.1. Tabletop Object Detector

Paket *Tabletop Object Detector* u sebi sadrži dvije osnovne komponenete[2] za percepciju objekata:

segmentacija objekata koja služi za izlučivanje klastera točaka koji odgovaraju objektima i

prepoznavanje objekata koja pomoću baze objekata određuje kojem objektu odgovara pojedini klaster.

Ovaj cjevovod nije potpuno općenit te da bi ispravno radio moraju biti zadovoljene slijedeće pretpostavke:

– objekti se nalaze na stolu koji je dominantna ravnina u sceni,

[1] Koristi se u ranije spomenutom cjevovodu predstavljenom u Ciocarlie et al. (2010).

[2] U terminologiji ROS-a radi se o čvorovima.

– minimalna udaljenost između objekata prelazi zadani prag (može se proizvoljno zadati),

– na čvor za prepoznavanje objekata ne utječu samo translacije objekta po x i y osi (pretpostavlja se da z os je usmjerena prema gore u odnosu na ravninu stola). Zbog toga ako se ovim sustavom želi prepoznati objekt tada on mora biti:

  - rotacijski simetričan (oko z osi) i
  - mora imati poznatu orijentaciju, tj. mora biti postavljen „uspravno“ na stolu.

Šematski prikaz protoka podataka u *Tabletop Object Detector* paketu dan je na slici 3.1, a u nastavku je opisan njegov rad. Ulazni, senzorski, podaci su oblaci točaka dobiveni iz stereo kamere ili *Kinect* senzora[3]. Korak segmentacije koristi RANSAC metodu za detekciju dominantne površine u sceni, a zatim se za sve točke iznad ravnine stola pretpostavlja da pripadaju skupini objekata koji se mogu uhvatiti i micati (engl. *graspable objects*). Iz tih se točaka pomoću klastering algoritma identificiraju grupe točaka (klasteri) koje pripadaju pojedinom objektu. U koraku prepoznavanja radi se jednostavna iterativna usporedba[4] s mrežom (engl. *mesh*) svakog objekta koji se nalazi u bazi. Ukoliko je za klaster pronađena odgovarajuća mreža tada se zajedno s klasterom vraća i identifikator modela u bazi kako bi se kasnije mogli dohvatiti pohranjeni podaci o objektu. Trenutno se metoda usporedbe provodi samo u 2D prostoru te pretpostavlja da su ostale četiri dimenzije fiksne (vidi uvjete navedene u prethodnom odlomku). Konačno ovaj cjevovod na izlazu daje lokaciju stola, identificirane klastere točaka i odgovarajući identifikator modela u bazi te procijenjeni položaj predmeta (ukoliko je pronađen odgovarajući objekt u bazi).

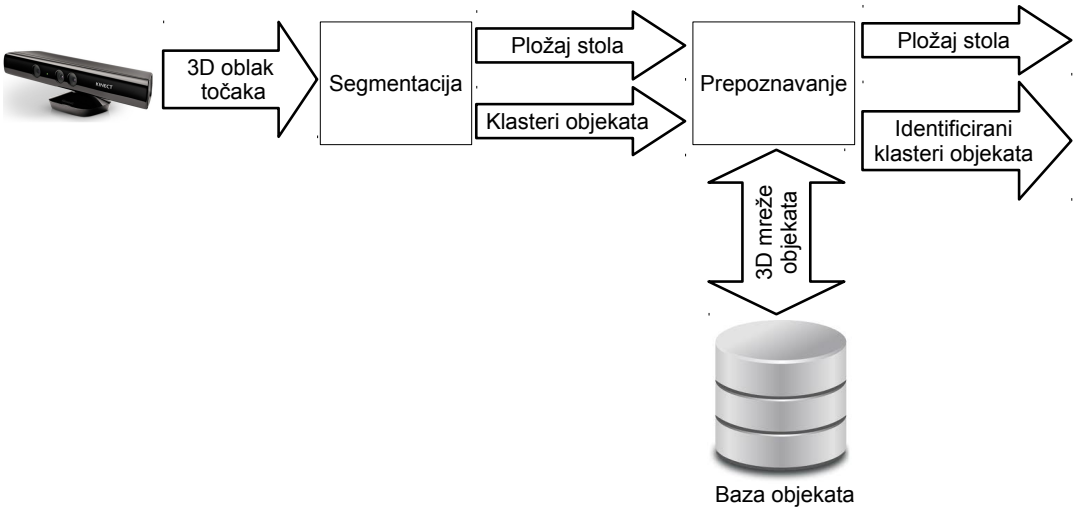
Više detalja o čvorovima i upotrebi ovog cjevovoda može se pronaći u Muja i Ciocarlie (2013).

3.2. Object Recognition Kitchen (ORK)

ORK je implamentacija infrastrukture za pojednostavljivanje izrade i korištenja različitih algoritama za prepoznavanje objekata. Za tu fleksibilnost zaslužan je ECTO

[3] Više o *Kinect* senzoru u odjeljku 5.1.1.

[4] Koristi se jedan od oblika algoritma Iterativna najbliža točka (engl. *Iterative Closest Point (ICP)*) odnosno, algoritam predstavljen u Besl i Mckay (1992).



Slika 3.1: Tabletop Object Detector cjevovod

okvir (Willow Garage, 2013a) u kojem se svaki zadatak može vrlo jednostavno opisati usmjerenim acikličkim grafom čime je definiran redoslijed izvršavanja podzadatka.

Trenutno su u ORK infrastrukturi implementirana četiri metode[5] za prepoznavanje objekta:

1. *Line-mod*,
2. *Tabletop Object Recognition*,
3. *Textured Object Detection* i
4. *Transparent objects*.

Uz ove metode u ORK-u su implementirani i dodatni alati koje koriste neke od metoda, a mogu biti i korisni za razvoj budućih metoda. Neki od važnijih alata su *Capture* (Willow Garage, 2013c) kojim se snimaju objekti za dodavanje u bazu objekata i *Reconstruction* (Willow Garage, 2013f) koji izrađuje aproksimativnu neteksturiranu mrežu iz snimljenih podataka o objektu koja može poslužiti za planiranje točaka hvata. Također uz ove cjevovode i alate implementirane su i ROS komponente za integraciju s ROS okruženjem (Willow Garage, 2013e).

Upotreba svakog algoritma u ORK-u zahtjeva tri koraka:

1. prikupljanje podataka,
2. trening (učenje) i

[5]Ovdje se metodom smatra algoritam za učenje (treniranje) objekata i algoritam za prepoznavanje.

3. detekciju i prepoznavanje.

Prvi korak služi prikupljanju podataka o objektu *Capture* alatom, a ono se mora obaviti na podlozi s nekim od predloženih uzoraka (vidi Willow Garage (2013c)) ili na teksturiranoj podlozi definiranoj od strane korisnika. Slijedeći korak nakon prikupljanja podataka je trening u kojem se definiraju i „izvlače“ važne značajke objekata na temelju kojih će se izvoditi prepoznavanje. Za svaku metodu potrebno je posebno provesti ovaj korak te dodati podatke u bazu. U koraku detekcije uspoređuju se izlučene značajke objekata koji se nalaze u sceni i onih spremljenih u bazi. U koracima treninga i detekcije moguće je paralelno koristiti više metoda čime se omogućava pouzdanija detekcija i prepoznavanje objekata.

U nastavku su detaljnije opisane četiri implementirane metode.

3.2.1. ORK: *Line-mod*

Line-mod metoda implementira *LINE-MOD* algoritam predstavljen u Hinterstoisser et al. (2011). Verzija koja se koristi u ORK sustavu odgovara verziji koja je implementirana u *OpenCV* programskom paketu (Itseez, 2013) te, kako se navodi u dokumentaciji (Willow Garage, 2013d), jedna je od najboljih metoda za prepoznavanje krutih objekata.

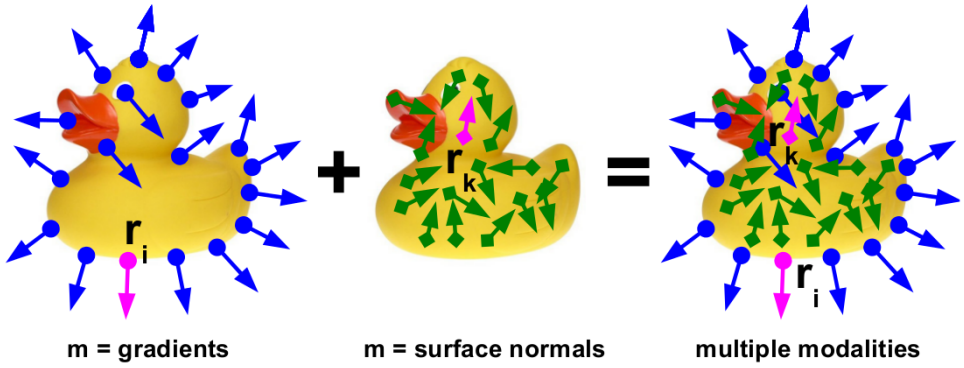
LINE-MOD (kratica od multiMODal-LINE) metoda koristi se za detekciju 3D objekata, a zasniva se na korištenju više različitih, neovisnih i komplementarnih izvora značajki odnosno modaliteta (engl. *modalities*) objekata. Modaliteti su odabrani tako da se jednostavno i jednoznačno mogu odrediti iz različitih vrsta slika (npr. RGB, slika dubine), a pošto je metoda općenita moguće ih je koristiti proizvoljan broj. Tijekom treninga objekata računaju se modaliteti za veliki broj predložaka na kojima je predmet transformiran, rotiran i skaliran. U tu svrhu implementiran je dodatni alat *Random view Generator* (Willow Garage, 2013g) koji pomoću vanjskih biblioteka generira različite poglede na objekt. U fazi prepoznavanja računaju se modaliteti za ulazne podatke koji se uspoređuju s ranije izračunatim modalitetima predložaka te zbog toga ova metoda spada u skupinu metoda baziranih na usporedbi predložaka. Da bi se osigurala velika brzina algoritma usporedbe i izvođenje u stvarnom vremenu izračunati modaliteti se ne uspoređuju izravno nego su prvo diskretizirani i zatim kodirani u binarne riječi (engl. *binary strings*) čime se omogućava iznimno brza usporedba. Prednosti *Line-mod* metode su rad u stvarnom vremenu, ispravan rad u prilično zakrčenim scenama (engl. *cluttered scene*), vrlo efikasna faza treninga i moć detekcije neteksturiranih objekata. Glavni nedostaci ove metode su nemogućnost rada pri djelomičnim

okluzijama te potreba za vrlo velikim brojem snimaka objekta s različitim pogledima i skaliranjima. Detaljan opis *Line-mod* metode kao i usporedba s drugim metodama za prepoznavanje objekata dana je u Hinterstoisser et al. (2011) dok je u Hinterstoisser et al. (2012a) opisano kako se kodiraju i kombiniraju značajke različitih modaliteta za efikasno korištenje memorije i procesorske snage što joj u konačnici i omogućava rad u stvarnom vremenu.

Ovdje korištena *Line-mod* metoda koristi dva modaliteta: normale površina i gradijente. Prednost ova dva modaliteta je u tome što su komplementarni te se lako dobivaju iz dva neovisna izvora podataka. Normale površina određuju se iz mape dubine prostora (engl. *depthmap*), a njima se opisuje unutarnja struktura objekta. Drugi modalitet, gradijenti, određuju se iz RGB slike zbog toga što se iz njih najbolje razlučuju. Prednost gradijenata je da su vrlo robusni na šum i promjene svjetline, a i kod objekata bez teksture su najčešće jedini pouzdani izvor informacija. Također, radi povećanja robusnosti, algoritam računa gradijent zasebno u sva tri kanala (R, G i B) te se kao konačni smjer uzima onaj s najvećom vrijednošću. Konačno se u daljnje kalkulacije prosljeđuju samo smjerovi gradijenata odnosno jedinični gradijenti čime se omogućuje neovisnost algoritma na transformacije i skaliranja. Na slici 3.2 prikazana je na primjeru komplementarnost gradijenata i normala površina objekta te se može vidjeti ideja koja leži iza korištenja više modaliteta za prepoznavanje objekata.

3.2.2. ORK: *Tabletop Object Recognition*

Tabletop Object Recognition cjevovod (Willow Garage, 2013i) je port metoda iz *Tabletop object detector* paketa (vidi poglavlje 3.1). Ova implementacija također ima dva dijela:



Slika 3.2: Line-mod primjer, izvor: Hinterstoisser et al. (2011)

Table Finder koji pronalazi ravnine i segmentira objekte koji se nalaze na njima, odnosno, ekvivalentan je čvoru za segmentaciju objekata i

Object Finder koji je ekvivalentan čvoru za prepoznavanje objekata.

Više o ovom cjevovodu može se naći u poglavlju 3.1.

3.2.3. ORK: *Textured Object Detection*

Textured Object Detection (TOD) metoda koristi tehniku standardnih značajki koja je implementirana u *OpenCV* programskom paketu, a namijenjena je prepoznavanju teksturiranih objekata. Tijekom treninga određuju se različiti pogledi na značajke objekata, te se, ukoliko su dostupne informacije o dubini prostora (mapa dubine), također određuje i 3D položaj objekta s obzirom na izlučene značajke. Tijekom koraka prepoznavanja računaju se značajke na trenutnoj slici te se uspoređuju s podacima određenim u koraku treninga, a sve u smislu određivanja analogne 3D konfiguracije. Za sada ova metoda radi samo ako su poznati i 3D podaci odnosno RGBD podaci te koristi *ORB* metodu za izlučivanje značajki. Više o konfiguriranju ovog cjevovoda može se naći u Willow Garage (2013h).

3.2.4. ORK: *Transparent objects*

Posljednja implementirana metoda *Transparent objects* može detektirati i estimirati položaj transparentnih objekata na osnovi modela oblaka točaka (engl. *point cloud model*). Implementirani algoritam predstavljen je u Lysenkov et al. (2012), a njegova najveća mana je to što se trening mora napraviti na neprozirnoj, odnosno obojanoj verziji predmeta da bi se mogao jednoznačno odrediti njegov 3D model. Ovaj algoritam nije korišten te je ovdje ukratko opisan kao jedna od mogućnosti u ORK sustavu. Više o korištenju algoritma i njegovoj integraciji u *Object Recognition Kitchen* sustav može se naći u Willow Garage (2013k).

4. Metode za izlučivanje točaka hvata i manipulacija objektima u programskom sustavu *ROS*

Slično kao i za prepoznavanje objekata programski paket ROS nudi nekoliko različitih algoritama za izlučivanje točaka hvata te manipulaciju objektima, ali se jasno razlučuju dva glavna cjevovoda (sustava) oko kojih su ti algoritmi implementirani. Prvi cjevovod nalazi se u stogu *Object Manipulation* (Ciocarlie i Hsiao, 2012) koji je stariji od ova dva sustava te time i kompletniji[1]. On implementira sve korake od detekcije i prepoznavanja objekata[2], preko izlučivanja točaka hvata do manipulacije objektima. Drugi, noviji, sustav koji se još uvijek nalazi u alfa stadiju razvoja naziva se *MoveIt!* (Sucan i Chitta, 2013). Ideja ovog sustava je da potpuno zamjeni *Arm Navigation* (Jones, 2013) stog kojeg koristi *Object Manipulation* cjevovod te na transparentniji način omogući planiranje i izvršavanje putanja manipulatora i mobilnih robota. Ranije spomenuti sustav za detekciju prepoznavanje objekata ORK prilagođen je upravo radu s *MoveIt!* sustavom te će u budućnosti cjevovod sastavljen od ta dva sustava u potpunosti zamijeniti *Object Manipulation* cjevovod. *MoveIt!* sustav nije korišten u okviru ovog rada, ali je ovdje naveden zbog važnosti koju ima za budući razvoj metoda za hvatanje i manipulaciju objektima.

U nastavku se detaljnije opisuju metode korištene u okviru ovog rada.

4.1. Metode određivanja hvata

U ovom radu korištena su dva paketa kao izvori točaka hvata, *Household Object Database* (Ciocarlie, 2012), dio *Object Manipulation* stoga te *PR2 Gripper Grasp*

[1] Ovaj sustav se koristi u Ciocarlie et al. (2010) navednom u poglavlju 2.

[2] Paket *Tabletop Object Detector* jedan je od paketa u ovom sustavu. Za detalje o upotrebi i međusobnoj povezanosti i međuovisnosti paketa u *Object Manipulation* cjevovodu vidi poglavlje 5.2

Planner Cluster (Hsiao, 2012), dio *PR2 Object Manipulation* stoga[3] (Hsiao i Ciocarlie, 2012).

Paket *Household Object Database* zapravo samo implementira komunikaciju s SQL bazom podataka u koju su spremljeni objekti za prepoznavanje, a s njima i mogući hvatovi za svaki objekt i definiranu hvataljku. Takav izvor točaka hvata spada u skupinu planera hvata potpomognutih bazom podataka (engl. *database-backed planner*) te se oni mogu koristiti samo ako se objekt nalazi u bazi i uspješno je prepoznat na sceni. Jedan od mogućih načina izlučivanja točaka hvata za objekte spremljene u bazu je upotrebom programa *GraspIt!* (Robotics Lab, 2013) u kojem je implementirana podrška za programski sustav ROS te automatsko učitavanje objekata i spremanje izlučenih hvatova. Važno je još napomenuti da se već popunjena baza objekata[4] čestih u kućanstvima može pronaći na ROS-ovim stranicama, no više o tome u Ciocarlie (2012).

Drugi paket *PR2 Gripper Grasp Planner Cluster* je pravi planer koji omogućava izlučivanje točaka hvata za nepoznate objekte odnosno za one objekte koji nisu prepoznati ili se ne nalaze u bazi objekata. Kao ulaz u planer predaje se segmentirani oblak točaka željenog objekta dobivenog, najčešće, stereo ili RGBD kamerom. Iako je ovaj planer izvorno namijenjen za hvataljku robota PR2 može se koristiti i za druge, ali se najprije treba složiti tako zvani box-model hvataljke u položaju prije hvata (engl. *pregrasp pose*). Više o ovom čvoru i njegovoj upotrebi može se naći u Hsiao (2012), a detalji o implementiranom algoritmu u Hsiao et al. (2010).

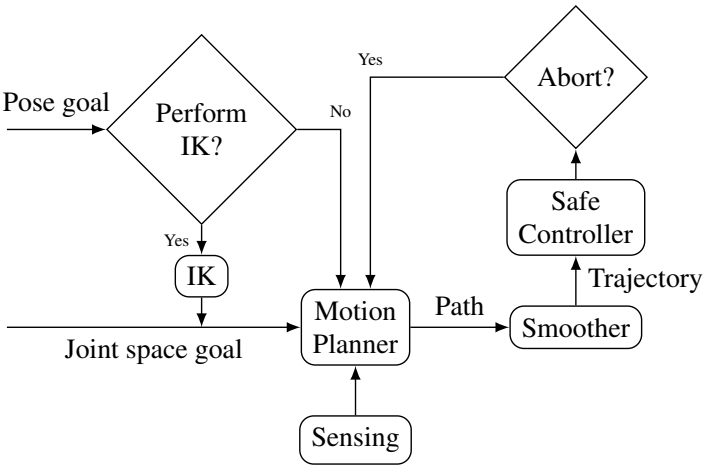
Kada se govori o hvatovima u cjevovodu *Object Manipulation* važno je još spomenuti da postoji i čvor za testiranje izvedbe generiranih odnosno unaprijed određenih hvatova. Testiranje se izvodi pomoću čvora *Tabletop Collision Map Processing* koji omogućava sučelje za popunjavanje i upravljanje mapom kolizija koja sadrži sve statičke prepreke, kao što je na primjer površina stola, te dinamičke prepreke (one koje se mogu micati) koje čine drugi objekti na stolu. Ovo testiranje izvodi se pomoću modula koji rješava problem inverzne kinematike (engl. *Inverse Kinematics (IK)*) odnosno određuje točan položaj ruke i hvataljke u prostoru stanja te se prema tome odlučuje da li je hvat izvediv ili ne. Za rješavanje problema IK moguće je koristiti generički modul, kao npr. modul implementiran u *Kinematics and Dynamics Library* (Smits, 2013), ili modul prilagođen robotskoj ruci, kao što je recimo modul prilagođen ruci PR2 robota (Chitta, 2013). Više o funkcionalnosti i upotrebi ovog čvora može se naći u Ciocarlie (2013).

[3] *PR2 Object Manipulation* stog sadrži specifične implementacije nekih generičkih funkcionalnosti prilagođenih za PR2 robot.

[4] Za gotovo sve objekte u bazi postoje i spremljeni hvatovi za nekoliko različitih hvataljki.

4.2. Manipulacija objektima

Za manipulaciju objektima *Object Manipulation* cjevovodom koristi se paket *Object Manipulator* (Ciocarlie i Hsiao, 2013). Ovaj paket implementira osnovnu funkcionalnost za zadatke uzimanja i premještanja objekata te također nudi servise i akcije za zadavanje tih zadataka. Izvršavanje zadatka uzimanja ili spuštanja objekta zahtjeva jako širok spektar različitih algoritama od percepcije prostora odnosno mogućih prepreka do planiranja putanja i upravljanja hvataljkom. Zbog toga *Object Manipulator* paket koristi akcije i servise velikog broja različitih paketa kao što je na primjer *Arm Navigation* za planiranje putanje ili *Tabletop Collision Map Processing* (Ciocarlie, 2013) za upravljanje kolizijama u okolini (engl. *collision environment*) i tako dalje. Arhitektura cjevovoda za manipulaciju objektima prikazana je na slici 4.1 i ona koristi generički algoritam iz *Kinematics and Dynamics Library* (Smits, 2013) za numeričko rješavanje IK problema, dok sam planer putanje nije ništa drugo nego sučelje prema *Open Motion Planning Library* (Şucan et al., 2012) koja implementira više različitih planera putanja. Detaljan opis svih potrebnih čvorova za ispravan rad *Object Manipulator* cjevovoda te objašnjenje kako se koji od njih koristi može se naći u Ciocarlie i Hsiao (2013); Ciocarlie et al. (2010).



Slika 4.1: Arhitektura cjevovoda za manipulaciju objektima, izvor: Jerbić et al. (2013)

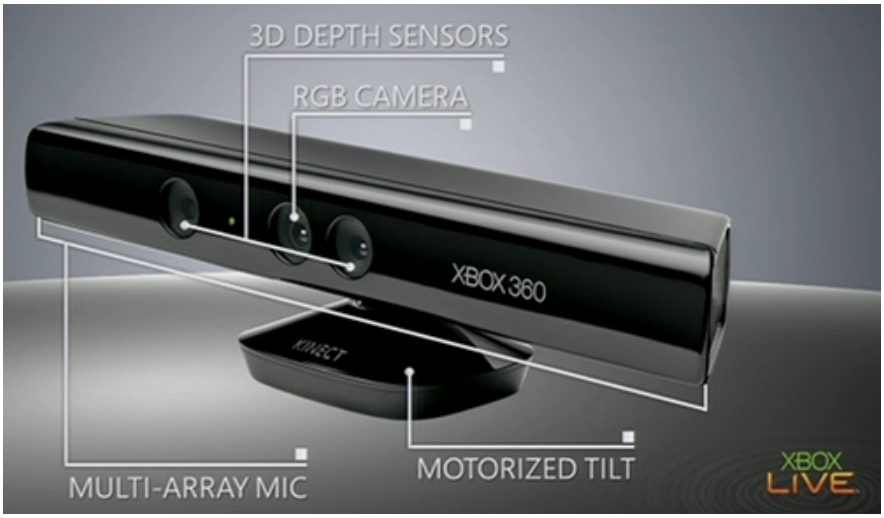
5. Upotreba programskog sustava *ROS* za detekciju, prepoznavanje i manipulaciju objektima

U prethodnim poglavljima teoretski su predstavljene metode za prepoznavanje i manipulaciju objektima implementirane u operacijskom sustavu za robote (ROS). Ovo poglavlje se bavi upotrebom tih metoda unutar simuliranog i stvarnog okruženja. Kao uvod u upotrebu, u poglavlju 5.1, ukratko je predstavljena oprema kojom su rađeni eksperimenti te su navedeni neki od problema s opremom i njihovim upravljačkim programima koji su se javljali tijekom rada. U poglavlju 5.2 opisano je iskustvo korištenja *Object Manipulation* cjevovoda za prepoznavanje i manipulaciju objektima u simulacijskom okruženju. *Object Recognition Kitchen* sustav nije korišten u simulacijskom okruženju zbog toga što bi, bez većih komplikacija, bilo moguće koristiti samo *Tabletop* cjevovod koji se već nalazi u *Object Manipulation* cjevovodu. Konačno u poglavlju 5.3 opisano je korištenje oba sustava za prepoznavanje objekata u stvarnom okruženju te su prezentirana postignuća u prepoznavanju objekata. Važno je još napomenuti da su se tijekom pokretanja i izvođenja cjevovoda mnogi postojeći paketi morali prilagoditi ili poopćiti da bi radili ispravno s opremom korištenom u ovom radu te su za njih kreirani novi repozitoriji čiji se popis može naći u dodatku A.

5.1. Oprema korištena prilikom izvođenja eksperimenata

5.1.1. Senzor Microsoft Xbox 360 Kinect

Prilikom izrade ovog rada za prikupljanje podataka korišten je uređaj Xbox 360 Kinect u kojem se nalazi nekoliko različitih senzora i aktuator (slika 5.1) od kojih su za ovaj

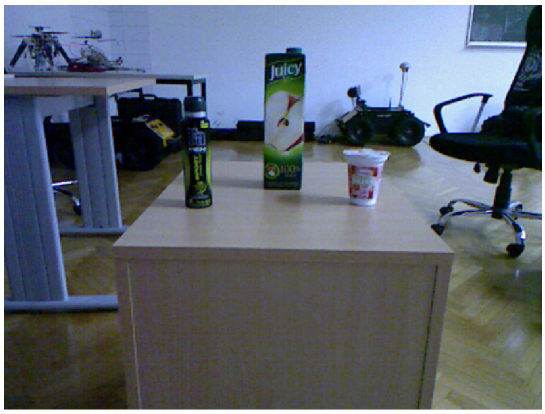


Slika 5.1: Xbox 360 Kinect uređaj, izvor: internet

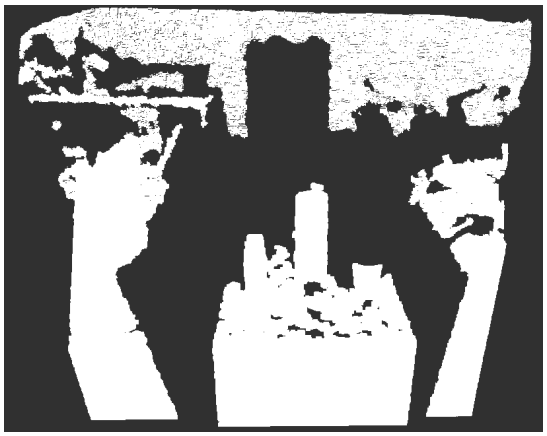
rad važni RGB kamera i 3D senzor dubine. Zbog ova dva senzora i mogućnosti kombinacije podataka senzora dubine i RGB kamere ovaj senzor pripada skupini RGBD (RGB + dubina) kamera. Upravljački program za komunikaciju sa senzorima može se vrlo jednostavno instalirati iz ROS paketa, a konkretan korišteni tip upravljačkog programa je *OpenNI* (Rusu et al., 2013; Mihelich et al., 2013).

Nakon pokretanja čvorova za inicijalizaciju i komunikaciju sa senzorom u ROS-u (vidi Mihelich i Kammerl (2013)) moguće je pristupiti svim važnijim parametrima senzora i upravljačkog programa kao i podacima. Za ovaj rad važne su teme na kojima se objavljuju podaci RGB kamere, senzora dubine te pripadajuće teme na kojima se objavljuju karakteristike pojedinog senzora (detalji o nazivima tema mogu se pronaći u Mihelich i Kammerl (2013)). Jedan od najvažnijih parametra upravljačkog programa je *depth_registration* koji kada je uključen povezuje podatke s RGB kamere i senzora dubine. Na slici 5.2 prikazan je rezultat aktivacije parametra *depth_registration*. Na slikama 5.2a i 5.2b prikazani su nepovezani podaci, a na slici 5.2c rezultat povezivanja podataka s navedenih senzora. Povezivanje se izvodi na način da se podaci senzora dubine (oblak točaka) transformiraju u koordinatni sustav RGB kamere te se zatim svakoj pojedinoj točki u oblaku pridružuje odgovarajući piksel RGB slike (ROS, 2012). Na slikama se može vidjeti da trenutno pridruživanje nije savršeno te da bi se transformacija između RGB i kamere dubine trebala korigirati, ali s obzirom na to da su te postavke zapisane u samom upravljačkom programu senzora nije ih se mijenjalo ni unutar ROS-a zbog moguće nekompatibilnosti s ORK sustavom (vidi snimanje objekata ORK sustavom, poglavlje 5.3.2).

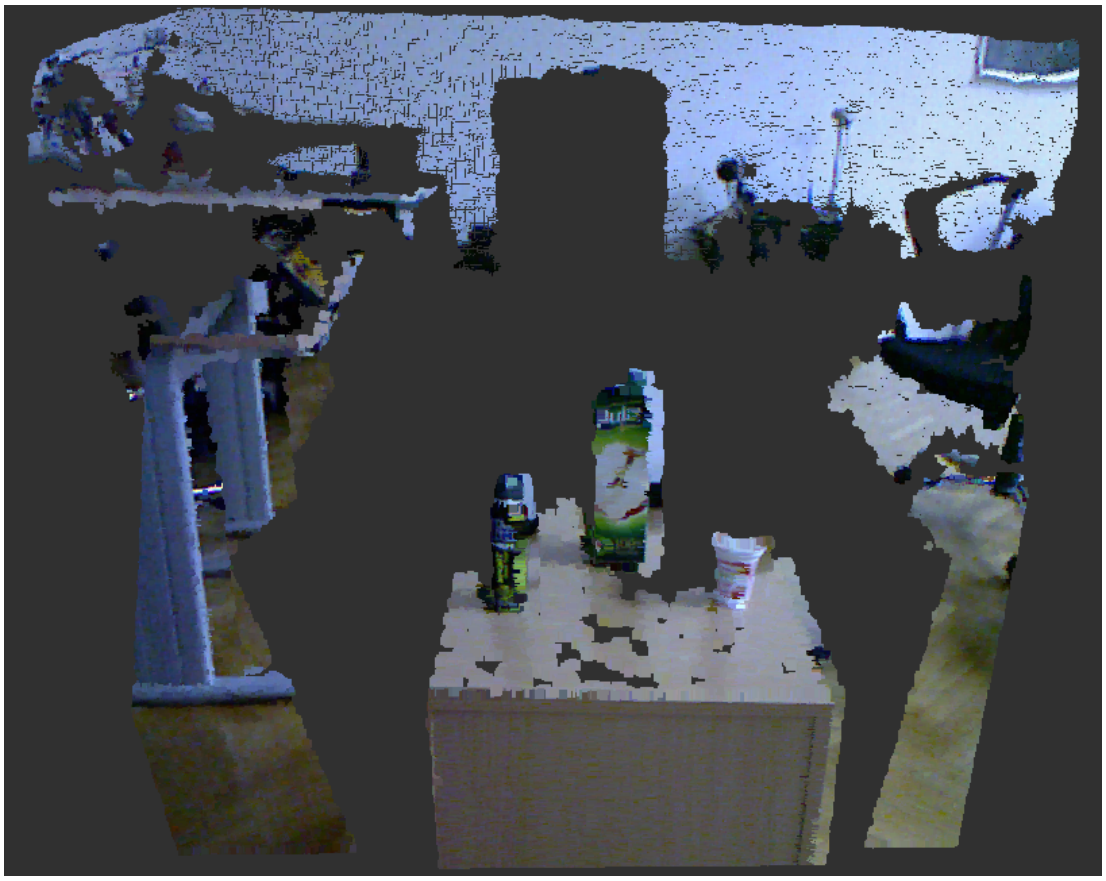
Jedan od problema koji se javljao tijekom rada je bio pogreška u upravljačkom pro-



(a) Podaci RGB kamere



(b) Podaci senzora dubine



(c) Povezani podaci

Slika 5.2: Povezivanje podataka RGB kamere i senzora dubine

gramu senzora koja je uzrokovala da se uključenjem opcije *depth_registration* podaci objavljuju na temi s transformiranim točkama, ali te točke nisu bile transformirane i nisu im bili pridruženi odgovarajući pikseli RGB slike. U najnovijoj inačici upravljačkog programa ovaj problem se više ne javlja, ali ga je bilo važno napomenuti jer su zbog toga u nekim čvorovima implementiranima u skopu ovog rada dodani dijelovi za transformiranje oblaka točaka (ovo će biti naglašeno pri opisu čvora).

Slijedeći problem 3D senzora dubine Kinect uređaja je nemogućnost detektiranja prozirnih objekata zbog toga što se senzor bazira na detekciji strukturiranog svjetla u infracrvenom području koje prolazi kroz prozirne objekte. Primjer tog problema prikazan je na slici 5.3 gdje se na sceni nalazila prozirna plastična boca od koje je 3D senzor dubine samo ispravno detektirao područje na kojem se nalazi (neprozirna) etiketa. Na slici 5.3a prikazana je RGB slika scene, a na slici 5.3b oblak točka s pridruženim odgovarajućim pikselima RGB slike tj. s uključenom opcijom *depth_registration*. Iako je ovaj problem u okviru ovog rada izbjegnut na način da su algoritmi za prepoznavanje objekata testirani samo na neprozirnim objektima, više o ovom problemu i načinu kako ga „zaobići“ prilikom prepoznavanja objekata može se naći u Lysenkov et al. (2012).

5.1.2. Ruka Schunk Powerball i hvataljka WSG-50

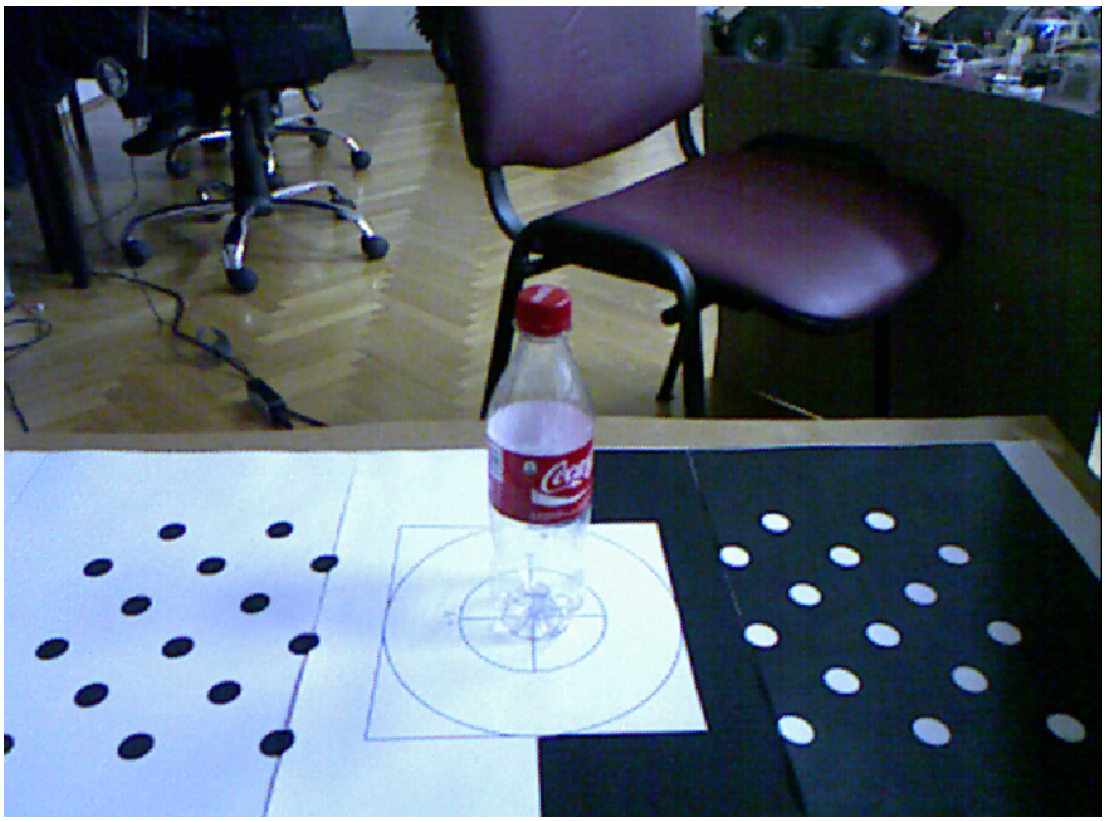
Za testiranje generiranih hvatova i manipulacije objekata u okviru ovog rada korištena je simulirana verzija robotske ruke *Schunk Powerball* s hvataljkom *Schunk WSG-50* (slika 5.4) . Ova ruka i hvataljka su odabrane zbog toga što su dostupne u laboratoriju te su dobro podržane u pogramskom sustavu ROS. Više o samim tehničkim detaljima može se pronaći na službenim stranicama proizvođača (SCHUNK GmbH & Co. KG, 2013), a ovdje, u nastavku, bit će navedeni i objašnjeni svi ROS paketi korišteni za pokretanje ruke i hvataljke.

Za upravljanje rukom i hvataljkom potrebna su slijedeća tri repozitorija:

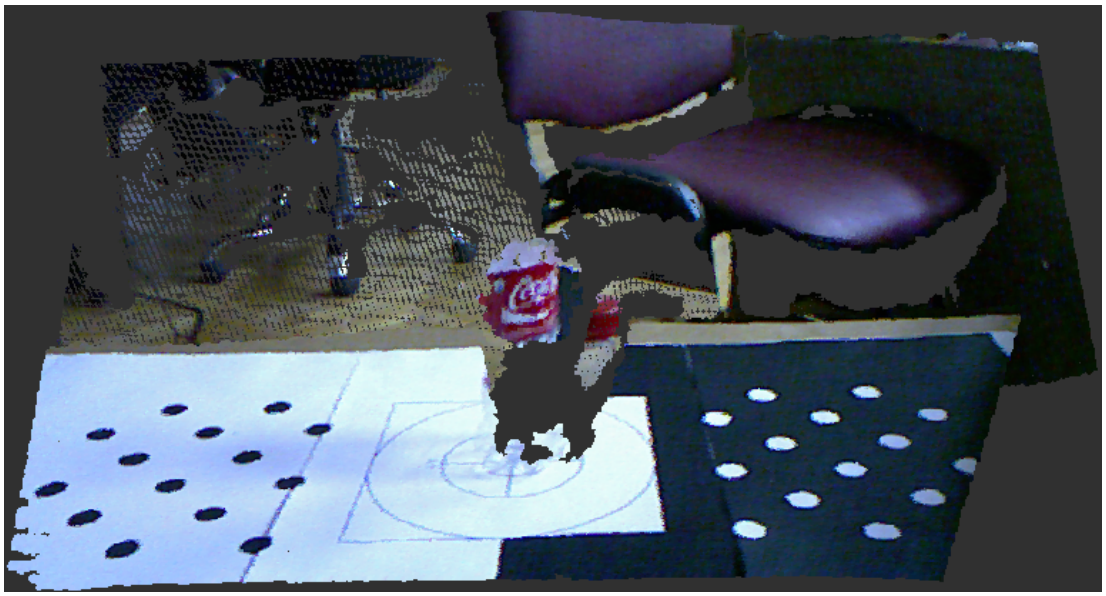
robotnik-powerball-ros-pkg Implementacija upravljačkog program za korištenje *Schunk Powerball* ruku s programskim paketom ROS (trenutno samo simulacija).

wsg50-ros-pkg Implementacija upravljačkog program za korištenje *Schunk WSG-50* hvataljke s programskim paketom ROS.

powerball_robotnik_arm_navigation Konfiguracija i pokretačke datoteke za Schunk Powerball pri korištenju s *Arm Navigation* cjevovoda.



(a) RGB slika



(b) Oblak točaka

Slika 5.3: Problem detekcije prozirnih objekata

Slika 5.4: Ruka Schunk Powerball s hvataljkom WSG-50, izvor: internet

Detalji i točna lokacija ova tri repozitorija može se pronaći u dodatku A. Prva dva repozitorija kopija su originalnih repozitorija te su prilagođeni radu s programskim paketom ROS verzije *"Groovy"* te su implementirani potrebni paketi o čemu se više može naći u nastavku ovog poglavlja.

5.2. Manipulacijski cjevovod u simuliranom okruženju

Simulirano okruženje za testiranje *Object Manipulation* cjevovoda napravljeno je simulatoru u *Gazebo* (Howard et al., 2013) koji je službeni i najrazvijeniji simulator za ROS (Hsu, 2013). Na slici 5.5 prikazan je *Object Manipulation* cjevovod što će omogućiti bolje razumjevanje čvorova i repozitorija. Simulacijsko okruženje prikazano je na slici 5.6, a sastoji se od Shunk Powerball ruke s hvataljkom WSG-50, senzora Kinect te stola na koji su postavljena dva objekta. Jedan objekt je limenka pića „Coca-Cola", a drugi posuda za teniske loptice. Oba predmeta nalaze se u „Household" bazi objekata koja se može skinuti s ROS stranica (vidi Ciocarlie (2012)) i korištena je u ovom radu. Datoteke za prikaz i simulaciju scene nalaze se u mapama *models* i *worlds* u repozitoriju *bci_visual_simuli*. Preporučljivo je koristiti objekte iz navednog repozitorija pošto su mnogi od njih prilagođeni ovoj simulaciji. Konačno za pokretanje

simulacije potrebni su slijedeći repozitoriji:

bci_visual_stimuli u kojem se nalaze pokretačke datoteke za simulaciju i konfiguracijske datoteke te čvorovi koji su implementirani u svrhu ovog rada,

robotnik-powerball-ros-pkg koji implementira upravljački program za korištenje *Schunk Powerball* ruke s programskim paketom ROS,

wsg50-ros-pkg koji implementira upravljački program za korištenje *Schunk WSG-50* hvataljke s programskim paketom ROS,

powerball_robotnik_arm_navigation koji sadrži konfiguracije i pokretačke datoteke za Schunk Powerball ruku za rad s *Arm Navigation* cjevovodom,

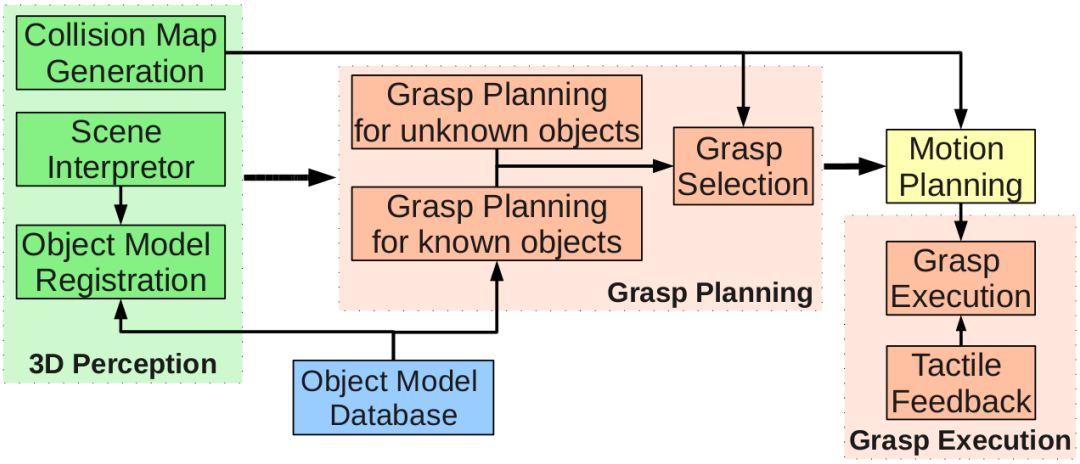
object_manipulation koji implementira čvorove za prepoznavanje, generiranje hvata i manipulaciju,

pr2_object_manipulation koji implementira planere za PR2 robot, ali se mogu koristiti i u ovoj konfiguraciji te

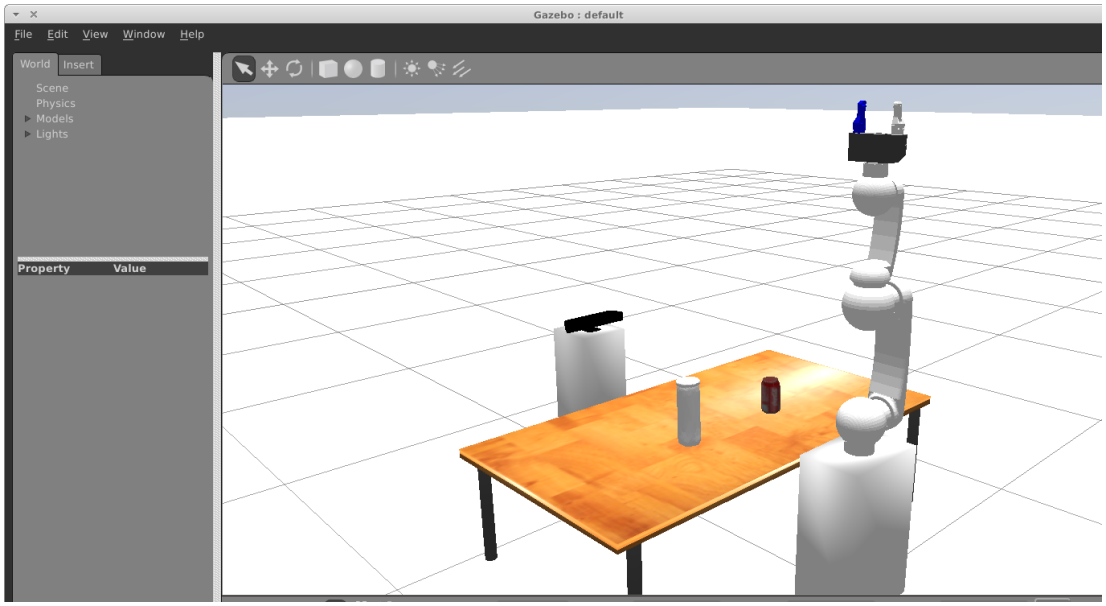
household_mesh_exporter koji implementira čvor za konverziju podataka između tipova *PointCloud* i *PointCloud2*, a potreban je zbog toga što simulirani Kinect senzor daje *PointCloud* tip podataka, a daljnji čvorovi u cjevovodu očekuju *PointCloud2*.

Većina ovih paketa može se instalirati i iz službenog ROS repozitorija prevedenih datoteka (engl. *prebuild files*) te je to dobra početna točka za pokušaj pokretanja bilo kojeg cjevovoda u ROS-u, ali zbog mogućih grešaka ili nemogućnosti izvođenja preporučljivo je provjeriti verzije u repozitorijima s izvornim kodom gdje su neke greške koje se javljaju, najčešće ispravljene. Prilikom izrade ovog rada svi navedeni repozitoriji su se ručno prevodili jer su se gotovo za svaki do njih morale raditi korekcije. Za detaljnija objašnjenja i adrese repozitorija korištenih u ovom radu vidi dodatak A.

Nakon uspješne instalacije svih paketa (ili uspješnog prevođenja ukoliko se koriste repozitoriji s izvornim kodom) može se pokrenuti simulacija pozivanjem pokretačke datoteke *launch/simulation.launch* iz repozitorija *bci_visual_simuli*. Ubrzo nakon pokretanja simulacije otvara se *Gazebo* prozor (slika 5.6). U nastavku će biti detaljnije opisana upotreba pojedinih dijelova *Object Manipulation* cjevovoda (slika 5.5).



Slika 5.5: Object Manipulation cjevovod, izvor: Jerbić et al. (2013)



Slika 5.6: Simulacijsko okruženje

5.2.1. Segmentacija prostora, detekcija i prepoznavanje objekata

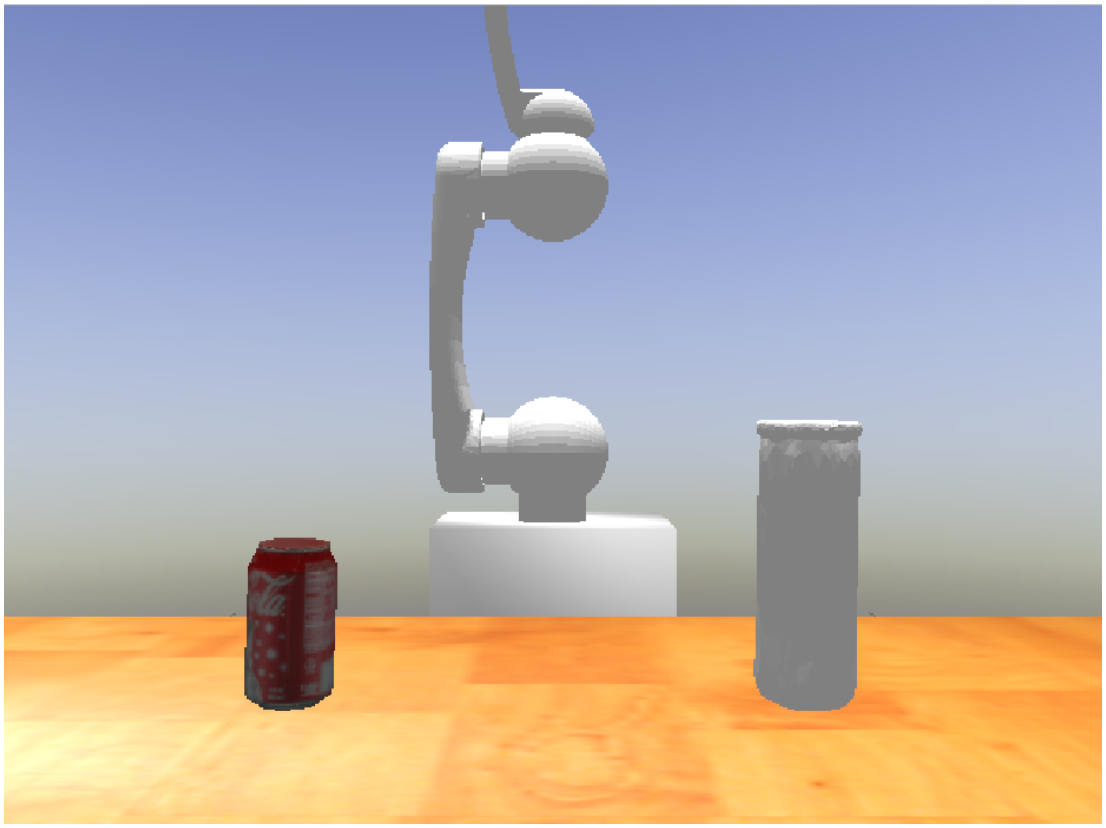
Na slici 5.7 prikazani su podaci dobiveni sa simuliranog Kinect senzora. Koordinatni sustavi RGB slike (slika 5.7a) i oblaka točaka (slika 5.7b) se poklapaju te nije potrebno raditi nikakve transformacije. Iako ovdje oblaku točaka nisu pridruženi podaci s RGB kamere kao što je to opisano u poglavlju 5.1.1 to nikako ne utječe na daljne izvođenje cjevovoda jer su takvi podaci koriste samo za bolju vizualizaciju, a u ovom slučaju se bitni dijelovi scene mogu lako raspoznati.

Nakon pozivanja servisa za segmentaciju prostora (na slici 5.5 interpretator scene (engl. *Scene Interpretor*)) stol i objekti su uspješno detektirani. Na slici 5.8 prikazan je rezultat segmentacije scene gdje je tirkiznom bojom označen dio stola koji je vidljiv u sceni, žutom estimirana površina stola te zelenom i crvenom objekti na stolu. Iako je rezultat segmentacije zadovoljavajući i objekti se nalaze u korištenoj „Household“ bazi objekta, prepoznavanje segmentiranih objekata (na slici 5.5 registracija modela objekta (engl. *Object Model Registration*)) je prošlo neuspješno. Naime vrijednost pouzdanosti (engl. *confidence*) niti jednog vraćenog modela, kao potencijalno odgovarajućeg, ne odskače od ostalih te se ne može sa sigurnošću zaključiti je li objekt prepoznat ili nije. Ovaj problem je detaljnije razmotren uvidom u izvorni kod čvora za prepoznavanje *Tabletop Object Recognition* (Muja i Ciocarlie, 2013), ali uzrok nije pronađen.

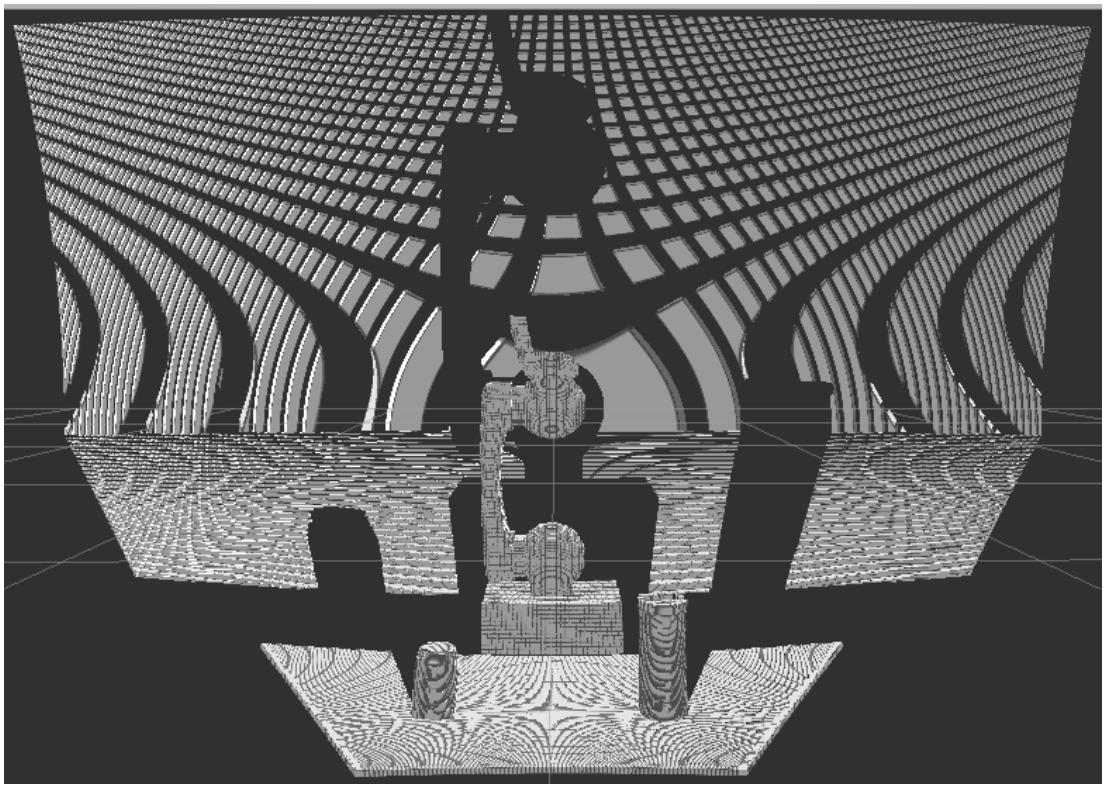
Nakon što je scena uspješno segmentirana potrebno je popuniti mapu kolizija pomoću generatora mape kolizija (engl. *Collision Map Generation*). U ovom slučaju se u mapu kolizija dodaje površina stola te svi segmentirani objekti zbog toga da se može ispravno izvesti korak testiranja odnosno odabira hvata za željeni objekt.

5.2.2. Izlučivanje točaka hvata i manipulacija objektima

Kao što je već napomenuto u poglavlju 4 koristeći cjevovod *Object Manipulation* moguće je koristiti dvije metode kao izvor točaka hvata: za nepoznate i za poznate objekte. Kao što se može vidjeti na slici cjevovoda (slika 5.5) da bi se odredile točke hvata za poznate objekte (engl. *Grasp Planning for known objects*) potrebno je prepoznati objekt i dohvatiti hvatove iz baze podatka. Iako algoritam za prepoznavanje objekata nije uspješno prepoznao niti jedan objekt u simulaciji *dipl. ing.* Srećko Jurić-Kavelj je pomoću programa *GraspIt!* generirao desetak različitih hvatova posude za teniske loptice za hvataljku korištenu u ovom radu (*Schunk WSG-50*). Tako generirani hvatovi pokušali su se iskoristiti u simulaciji tako da su se poslali sustavu za testiranje hvatova (na slici 5.5 odabir hvatova (engl. *Grasp Selection*)) bez obzira na rezultate prepoznavanja, ali niti jedan hvat nije prošao testiranje kao valjan. Mogući razlozi su

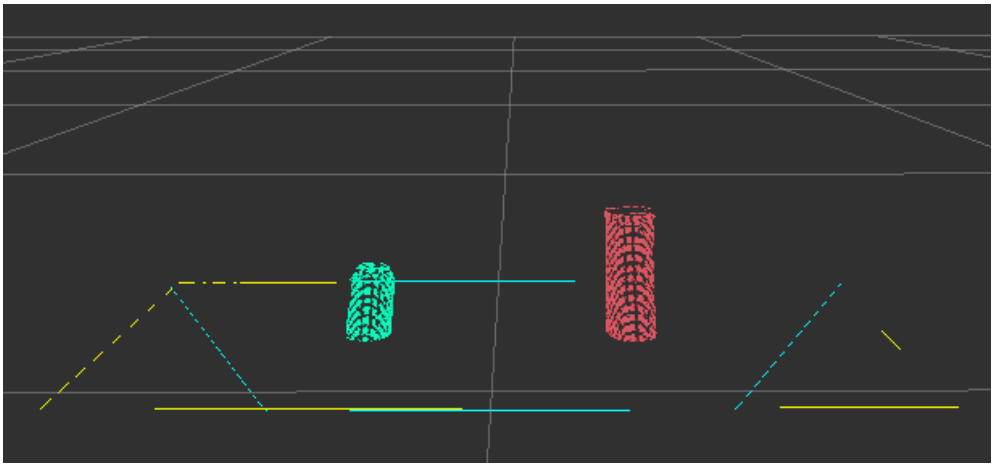


(a) RGB slika



(b) Oblak točaka

Slika 5.7: Prikaz podataka Kinect senzora tijekom simulacije



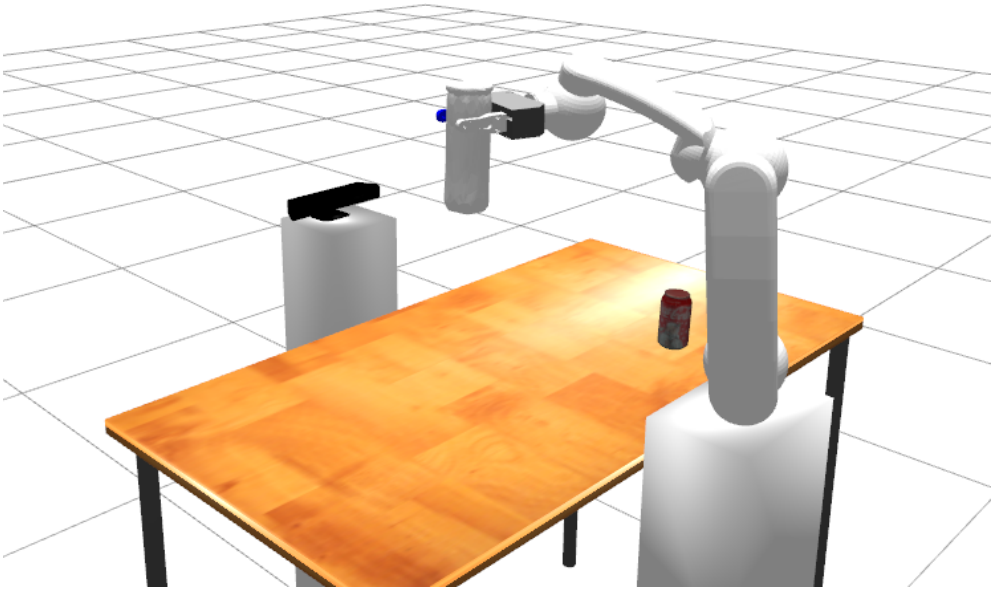
Slika 5.8: Rezultat segmentacije scene

taj što objekt nije prepoznat te tako ni njegov lokalni koordinatni sustav nije postavljen u odgovaraći položaj i orijentaciju ili su unaprijed izlučeni hvatovi doista neostvarivi u ovom slučaju. Moguće riješenje problema je izlučivanje značajno većeg broja hvatova ili redefinicija točaka kontakta hvataljke i objekta na temelju kojih se izlučuju hvatovi. Ali oba navedena rješenja su vrlo složena i vremenski zahtjevna.

Drugi planer točaka hvata za nepoznate odnosno neprepoznate objekte (engl. *Grasp Planning for unknown objects*) koristi kao ulaz samo segmentirane klastere (slika 5.8) iz čega estimira poziciju i oblik objekta. Ovaj planer prosječno generira nekoliko stotina različitih hvatova koji se zatim testiraju isto kao i hvatovi dohvaćeni iz baze objekata. Ovaj planer se pokazao boljim izborom, te se preporuča njegovo korištenje, zbog toga što je s njim uspješno određen adekvatan hvat te je nastavljeno izvršavanje cjevovoda.

Za testiranje hvatova, u ovom radu, koristio se generički modul za rješavanje problema inverzne kinematike koji se može naći u Smits (2013). Konačni odabir hvata bio je na način da čim se nađe prvi zadovoljavajući (kojeg je moguće izvesti) prekida se ispitivanje te se on koristi prilikom hvatanja objekta.

Nakon uspješno određenog hvata pokreće se planiranje putanje te kada je ono uspješno izvedeno pokreće se izvođenje isplanirane putanje i hvata. Da bi se hvatanje predmeta izvelo uspješno *Object Manipulation* cjevovod zahtjeva akciju za regulaciju hvataljke. Pošto ova akcija ne postoji u originalnom repozitoriju s upravljačkim programima hvataljke *WSG-50* (*wsg50-ros-pkg* repozitorij) ona je u sklopu ovog rada implementirana te se može naći u verziji repozitorija navedenog u dodatku A. Iako u dijagramu na slici 5.5 stoji kako za uspješno izvođenje hvata (engl. *Grasp Execution*) je potrebna taktilna povrtatna veza (engl. *Tactile Feedback*) ona prilikom simulacijskog



Slika 5.9: Hvatanje i podizanje predmeta u simuliranom okruženju

izvođenja nije nužna jer ne postoji opasnost od materijalnog oštećenja hvataljke ili predmeta. Konačan rezultat uspješnog izvođenja hvatanja i podizanja predmeta pomoću *Object Manipulation* cjevovoda prikazan je na slici 5.9.

5.3. Segmentiranje i prepoznavanje objekata u stvarnom okruženju

Metode segmentacije i prepoznavanja objekata isprobane su i u stvarnom okruženju. U ovom slučaju koristila su se oba cjevovoda te različite implementirane metode. RGB slika i oblak točaka scene na kojoj su se isprobavali algoritmi prikazani su na slici 5.10. Kao što se može vidjeti na sceni se nalazi dominantna površina (stol) s tri objekta različitih visina i oblika. U pozadini su namjerno ostavljeni objekti koji mogu uzrokovati potencijalne smetnje tako da se može provjeriti robusnost metoda. Ova tri predmeta su odabrana zbog toga što nisu prozirni te imaju teksturiranu površinu čime se omogućava korištenje gotovo bilo koje metode koje se nalaze u ROS-u (vidi poglavlje 3).

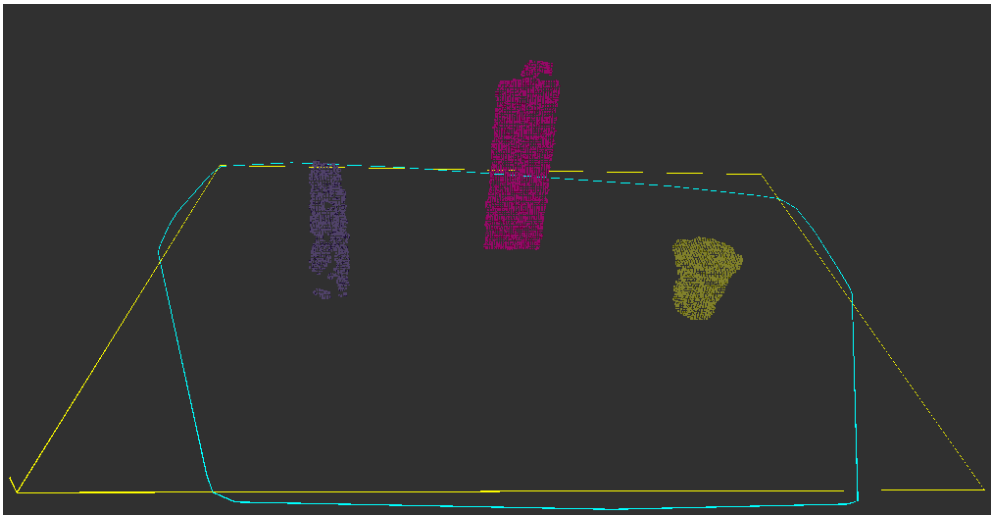
5.3.1. Tabletop Object Detector

U sceni koja je služila za test ovog algoritma nije se nalazio niti jedan objekt iz *„Household“* baze zbog čega se algoritam za prepoznavanje objekata nije testrirao, ali se ovo ne smatra nedostatkom s obzirom na rezultate dobivene u simuliranom okruženju

(a) RGB slika

(b) Oblak točaka

Slika 5.10: Prikaz podataka Kinect senzora sa stvarne scene



Slika 5.11: Rezultat segmentacije scene

(poglavlje 5.2.1). Kao što je rečeno i pokazano ranije, za manipulaciju objektima nije nužno prepoznati objekt nego ga je dovoljno segmentirati iz oblaka točaka prostora i prikazati kao klaster točaka. To je učinjeno i ovdje, na stvarnoj sceni, čime se algoritam za segmentaciju *Tabletop Object Detector* cjevovoda pokazao vrlo pouzdanim. Rezultat segmentacije prikazan je na slici 5.11 gdje je tirkiznom bojom označen rub vidljive površine stola, žutom estimirana površina stola te ostalim bojama segmentirani klasteri objekata. Iako se, u okviru ovoga, rada nije radilo ništa sa stvarnom robotskom rukom svejedno su se, u testne svrhe, odredile točke hvata za segmentirane klastere čime se napravio korak bliže budućoj upotrebi stvarne ruke za manipulaciju predmetima[1].

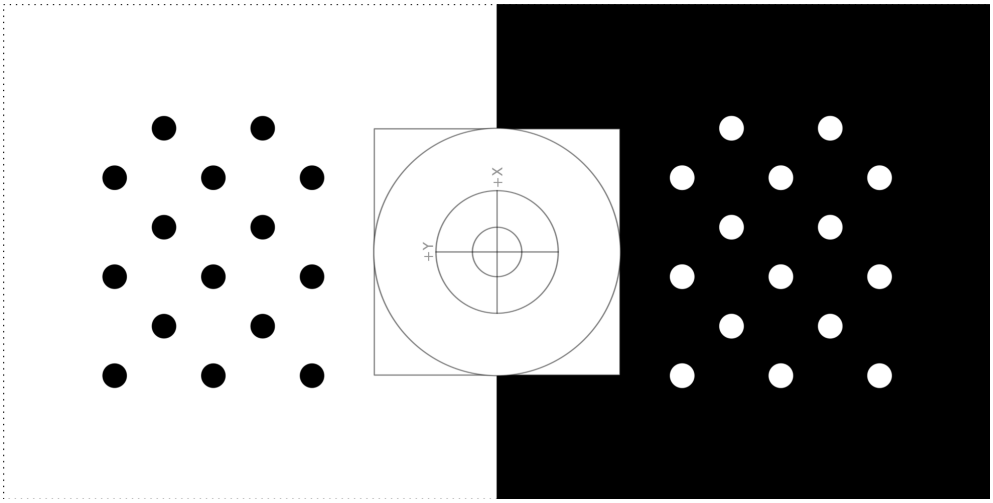
5.3.2. Object Recognition Kitchen (ORK)

Instalacija ORK sustava je također provedena iz izvornog koda koji je dohvaćen iz službenih repozitorija čiji se popis može naći na službenim stranicama projekta (Willow Garage, 2013b). Pri tome treba paziti da ovi paketi koriste novi sustav prevođenja izvornog koda u ROS-u koji se naziva *catkin*[2]. Ukoliko su unaprijed instalirani svi paketi o kojima ovisi ovaj sustav tada bi prevođenje trebalo proći bez problema.

Prepoznavanje objekata pomoću ORK sustava omogućuje korištenje nekoliko metoda za koje prvo objekti trebaju biti snimljeni i dodani u bazu objekata jer ovaj sustav

[1] Samo pokretanje stvarne ruke i hvataljke pomoću *Object Manipulation* cjevovoda je vrlo dugotrajan proces zbog mnoštva parametara koje treba postaviti i provjeriti te zbog vremenskog ograničenja to nije ostvareno. Također, upotreba stvarne ruke i hvataljke zahtjeva i mnogo više vremena zbog sigurnosnih razloga.

[2] Za razliku od repozitorija navedenih u dodatku A koji još uvijek koriste *rosbuild* sustav prevođenja.



Slika 5.12: Točkasti uzorak za snimanje podataka

ne dolazi s unaprijed popunjenom bazom, ali je zato snimanje iznimno lako. Nakon snimanja željenih objekta oni se treniraju željenim metodama pri čemu se određuju njihove značajke te je sustav tek tada spreman za upotrebu. U slijedećih nekoliko podpoglavlja opisana su iskustva upotrebe ovog sustava te će se navesti problemi, i neka rješenja, koji su se pojavili tijekom upotrebe.

U svrhu rada s ORK sustavom napravljen je repozitorij *ork_ros* u kojem su implementirane konfiguracijske i pokretačke datoteke (vidi dodatak A).

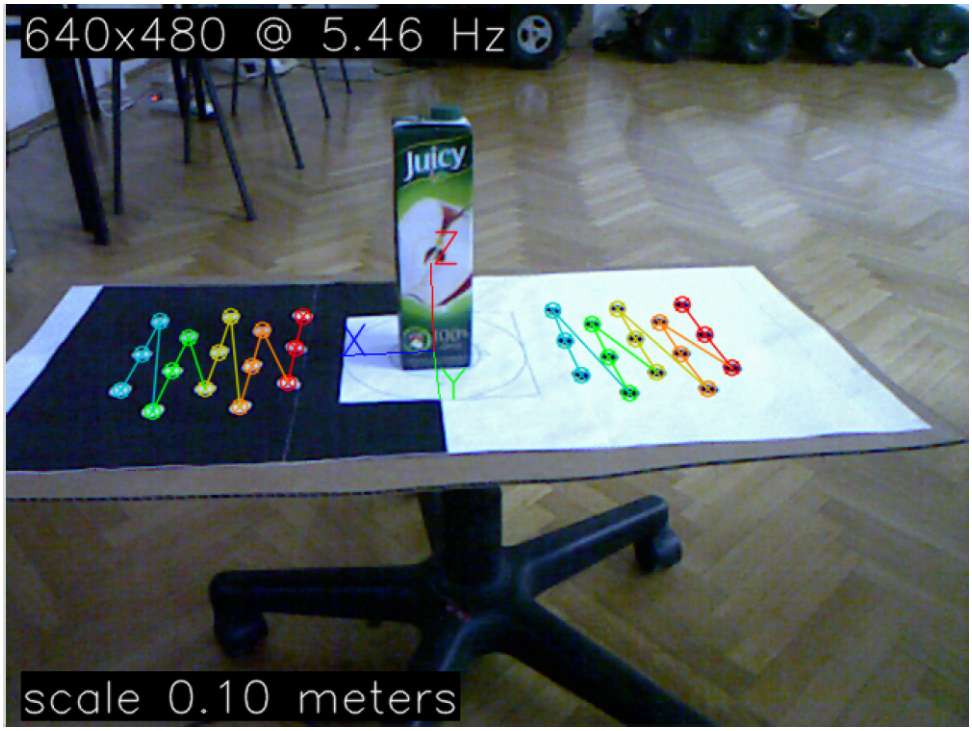
Snimanje objekata ORK sustavom

Snimanje objekata ORK sustavom, za razliku od snimanja objekata za *„Household“* bazu, je vrlo jednostavno i za to se koristi *Caputure* paket ORK sustava (Willow Garage, 2013c). Prilikom snimanja potrebno je koristiti neki teksturirani uzorak kao podlogu kako bi se snimani objekt mogao postaviti u lokalni koordinatni sustav te mu se odrediti veličina. Za to je moguće koristiti više različitih standardnih uzoraka preporučenih u dokumentaciji, ali čak i proizvoljnu, teksturiranu, ravnu površinu. U ovom radu korišten je točkasti uzorak (engl. *Dot Pattern*) (slika 5.12) koji je postavljen na ravnu podloga s mogućnošću rotacije od $360°$ što je nužno da bi se predmet kvalitetno snimio iz što je moguće više pogleda.

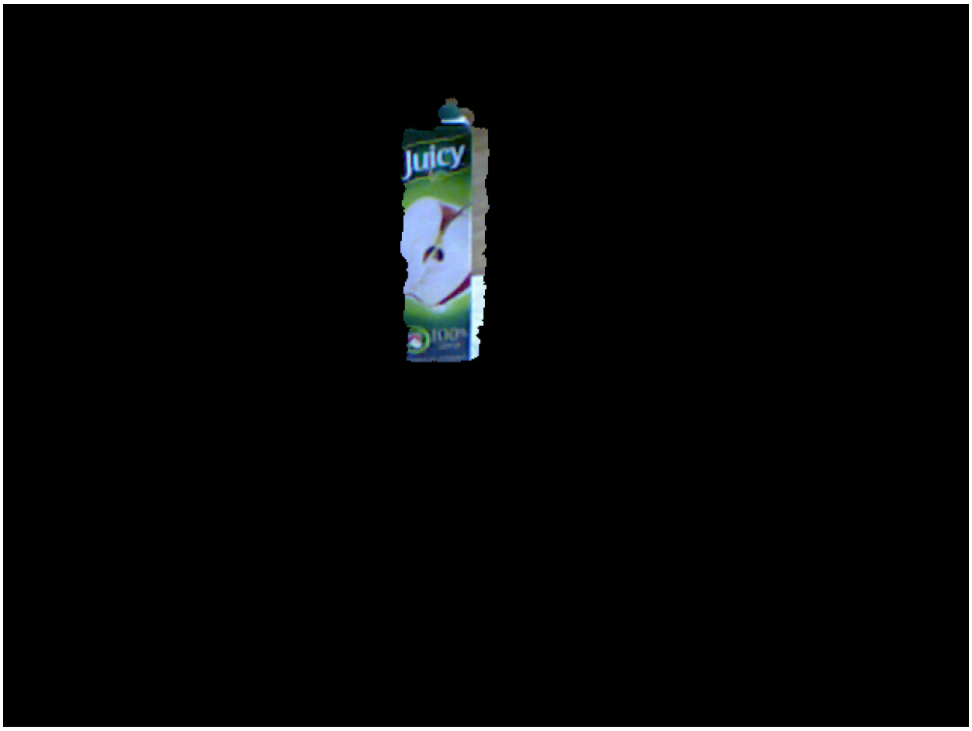
Snimanje se provodi tako da se objekt stavi na označeno mjesto na sredini točkastog uzorka te se pokrene i izvrši snimanje prema uputama u Willow Garage (2013c). Nakon pokretanja snimanja otvaraju se dva prozora (slika 5.13) na kojima je prikazan objekt s prepoznatim uzorkom i lokalnim koordinatnim sustavom (slika 5.13a) i maska na temelju koje se izdvaja sam objekt od ostatka scene (slika 5.13b). Važno je uočiti

da se maska na slici 5.13b ne poklapa savršeno s objektom (postoji pomak u lijevo), a uzrok tome je neprecizno definirana transformacija između RGB kamere i kamere dubine[3] u samom upravljačkom programu na koji se ovaj sustav izravno spaja. Iako se nekoliko puta pokušalo pronaći i prilagoditi transformacijske matrice u *OpenNI* upravljačkom programu, do završetka ovog rada nije pronađeno adekvatno rješenje.

[3]Ovaj problem je spomenut i u odjeljku 5.1.1



(a) RGB slika s prepoznatim uzorkom i koordinatnim sustavom objekta



(b) Maska na temelju koja izdvaja sam objekt

Slika 5.13: Prozori pri snimanju objekata ORK sustavom

U okviru ovog rada snimljeno je, i dodano u bazu, pet objekata koji su prikazani na slici 5.14. Objekti su odabrani tako da ih ima s teksturiranom (slike 5.14b 5.14c 5.14e

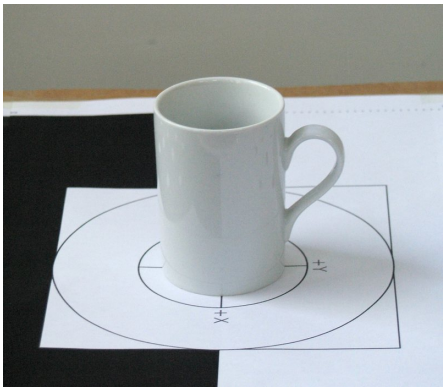
5.14d), neteksturiranom (slika 5.14a) površinom te da su značajno različite veličine. Za sve ove objekte izgenerirana je i mreža (engl. *mesh*) pomoću alata *Reconstruction* (Willow Garage, 2013f) koja je dodana u bazu. Generirana mreža vrlo grubo opisuje objekt, ali dovoljno precizno da može služiti kao ulaz u algoritam za izlučivanje točaka hvata. Kada se govori o objektima u bazi podataka važno je napomenuti i da postoji ROS čvor koji implementira servis za dohvaćanje podataka o svakom pojedinom objektu, ali tijekom ovog rada nije niti jednom uspio dohvatiti informacije o objektu iz baze nego je uvijek vraćao pogrešku.

Najveći problem kod snimanja objekata bilo je rušenje programa na operacijskom sustavu za 32-bitnu arhitekturu kada bi točkasti uzorak bio pronađen. Do kraja pisanja ovog rada uzrok nije pronađen iako se sumnja na *OpenCV* biblioteku (Itseez, 2013) koja se koristi za detekciju i prikaz uzorka na slici. Problem je otklonjen tako da je instalirana 64-bitna verzija operacijskog sustava na kojem je i obavljen prethodno opisani postupak snimanja.

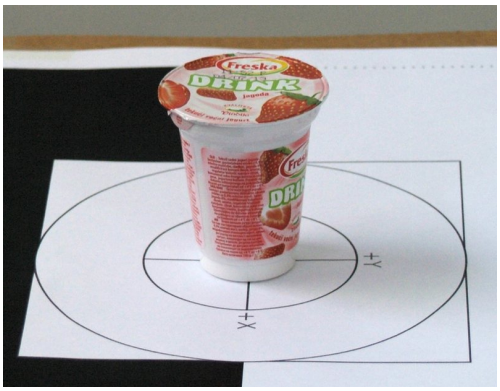
Treniranje objekata ORK sustavom

Kao što je već nekoliko puta navedeno, da bi algoritmi implementirani u ORK sustavu funkcionirali vrlo je važan korak treniranja objekata odnosno određivanja njihovih specifičnih značajki na temelju kojih će se izvršavati prepoznavanje. U okviru ovog rada uspješno je trenirano svih pet objekata dodanih u bazu i to za *LINE-MOD* i *TOD* metode. Detaljno razmatranje ovih metoda dano je u poglavlju 3.2, a upute o pokretanju i konfiguriranju treninga se mogu naći u Willow Garage (2013j). Za *Table-top* metodu nije potrebno provesti trening jer ona koristi već popunjenu *„Household“* bazu objekata.

Jedan od problema kod ovog koraka bilo je rušenje treninga *LINE-MOD* metode s obavijesti *Ilegal instruction*. Uzrok problema bio je u *PCL* biblioteci (PCL, 2013) koja je prevođena na procesorskoj arhitekturi koja podržava neke naprednije procesorske naredbe za razliku od one koja se koristila u okviru ovog rada. Detaljnije o problemu i njegovom rješenju može se naći na slijedećoj poveznici: `https://code.ros.org/lurker/message/20130411.082133.f1e7a1c0.en.html`. Slijedeći problem nastao je zbog korištenja vlasničkog (engl. *proprietary*) upravljačkog programa za grafičku karticu te se tijekom izvođenja nije mogla naći jedna od biblioteka koja se koristi za generiranje različitih pogleda na objekt za treninranje *LINE-MOD* metode. Problem je riješen instalacijom upravljačkih programa otvorenog koda.



(a) Šalica za kavu



(b) Jogurt od jagode

(c) Kutija *Kinect* senzora

(d) Sok od jabuke

(e) Dezodorans FA

Slika 5.14: Objekti snimljeni ORK sustavom

Prepoznavanje objekata ORK sustavom

U okviru ovoga rada isprobani su *LINE-MOD*, *TOD* i *Tabletop* metode za prepoznavanje ORK sustavom. Važno je napomenuti da se za to koristile najnovije verzije iz repozitorija izvornog koda u kojima su popravljene neke vrlo bitne pogreške koje su ranije spriječavale izvođenje. Za svaku od navednih metoda napravljene su konfiguracijske datoteke koje se mogu naći u repozitoriju *ork_fer*.

LINE-MOD metoda se pokazala ne toliko robusnom i pouzdanom kao što je predstavljena u izvornom radu (Hinterstoisser et al., 2011) jer se javljalo mnogo lažnih detekcija čak i kada se niti jedan objekt nije nalazio na sceni. Uzrok tome može biti način generiranja trening podataka za što se koristi alat *Renderer* (Willow Garage, 2013g). Tijekom izvođenja je zapaženo da LINE-MOD metoda često detektira objekte manjih dimenzija, kao što je na primjer jogurt (slika 3.2), kao dio nekog većeg objekta kao na primjer sok od jabuke 3.2.

Implementacija *Tabletop* metode se nije pokazala toliko kvalitetnom kao u *Object Manipulation* cjevovodu jer u niti jednom pokušaju korak segmentacije nije uspješno proveden, dakle nije izlučen stol pa stoga ni objekti iz scene. Razlog tome nije poznat, ali je moguće da neki od parametara nisu bili adekvatno postavljeni kao u slučaju korištenja *Tabletop Object Detector* paketa.

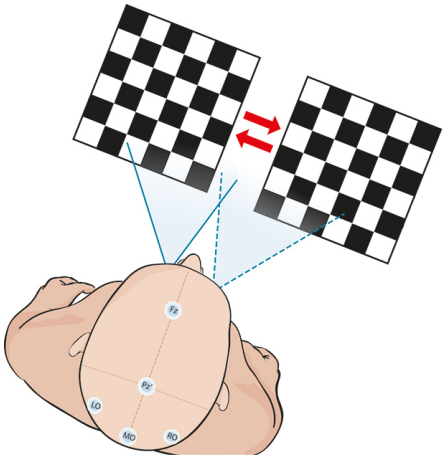
TOD metoda, kao ni prve dvije metode, se nije pokazala uspješnom i pouzdanom pri prepoznavanju objekata. Ova metoda je najčešće vraćala kao rezultat neki od većih objekata iz baze (npr. sok od jabuke ili kutiju Kinect senzora), a ostale objekte uopće nije uspjela detektirati. Jedan od mogućih razloga je mala rezolucija korištenog senzora koja iznosi 640×480 piksela (VGA razolucija) jer se u primjerima dokumentacije koristi *Kinect* senzor s rezolucijom 1024×768 piksela (XGA).

Iako ORK sustav obećava svojom arhitekturom i fleksibilnošću rezultati dobiveni njime su razočaravajući. Ali je važno spomenuti da u okviru ovog rada nije provedeno ekstenzivno konfiguriranje i namješanje parametara za svaki pojedini korak i svaku pojedinu metodu, te također treba uzeti u obzir da je to vrlo mlad sustav koji se intenzivno razvija te je za očekivati veću stabilnost i pouzdanost u budućnosti. Moguća poboljšanja ovdje korištenih metoda su kroz podešavanje parametara svakog pojedinog algoritma, ali i koraka snimanja, treninga i detekcije. Također bi bilo dobro koristiti metode detekcije objekata (kao na primjer čvor za segmentaciju u *Tabletop Object Recognition* paketu) čime bi se djelovanje algoritama moglo ograničiti samo na područja gdje se nalaze objekte, a valjalo bi i bolje definirati transformaciju između koordinatnih sustava RGB kamere i kamere dubine u *OpenNI* upravljačkom programu.

6. Generiranje podražaja za sučelje mozak-računalo

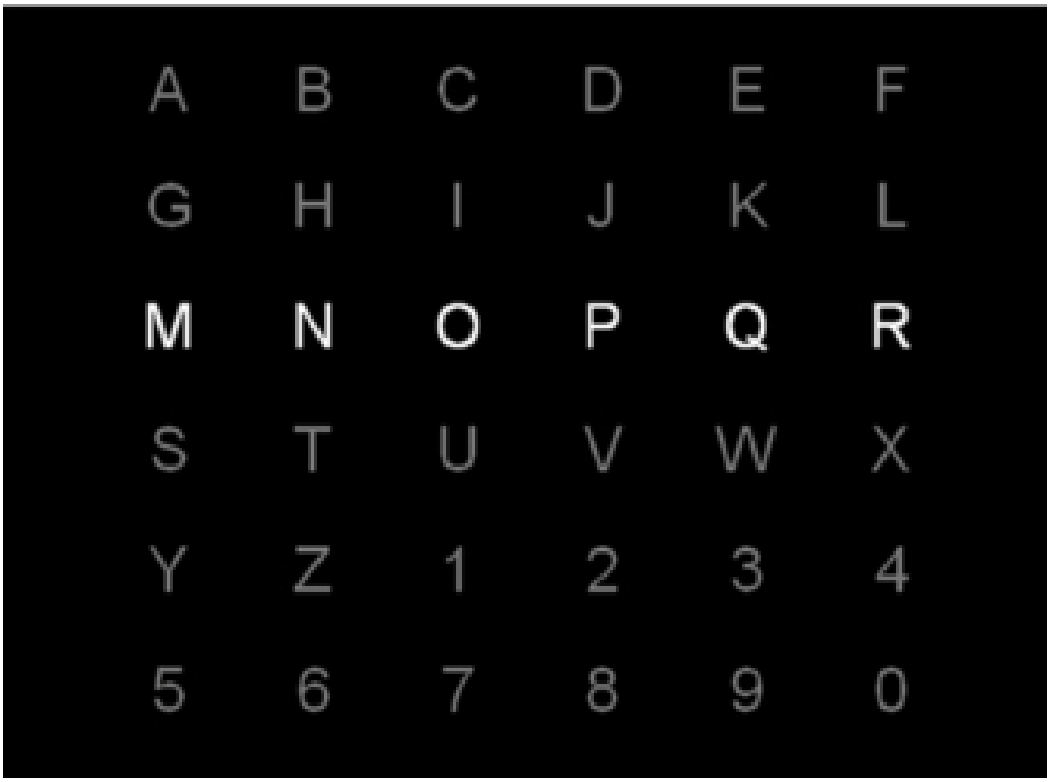
Zadatak sučelja mozak-računalo detekcija je specifičnih uzoraka u električnoj aktivnosti mozga te njihovo korištenje za upravljanje računalima i drugim uređajima. Za ovu primjenu koristi se više različitih uzoraka električne aktivnosti mozga i neki od najpopularnijih su stacionarni vizualno pobuđeni potencijali (engl. *Stady-state visual evoked potentials (SSVEPs)*), P300 val (engl. *P300 wave*) i promjene u oscilacijama neuronskih signala tijekom izvršavanje mentalnih zadaća (npr. zamišljanje motoričkih pokreta).

Stacionarni vizualno pobuđeni potencijali javljaju se prilikom elektroencefalografije (EEG) kada je osobi predočen slijed vizualnih pobuda (Byczuk et al. (2012)). U praksi se koristi više različitih načina generiranja SSVEP stimulacija. Jedan od najčešćih načina korištenja je prekrivanje objekata šahovnicom koja izmjenjuje crna i bijela (prozirna) polja (slika 6.1), a drugi korištenje LE diododa koje se pale i gase određenom frekvencijom. SSVEP metoda za pobudu najčešće koristi frekvencije u rasponu od 5 do 35 Hz, pri čemu je važno za svaku osobu odrediti frekvencije na koje najbolje „reagira“ (Jerbić et al., 2013).



Slika 6.1: SSVEP stimulacija sa šahovnicom, izvor: internet

Druga popularna metoda, P300 val, je komponenta električnog potencijala mozga povezana s događajem (engl. *Event-related brain potential (ERP)*), a manifestira se kao nadvišenje u EEG-u koji ima maksimum otprilike 300 ms nakon rijetkog i značajnog podražaja. Najčešće korištena paradigma koja koristi P300 za BCI je P300 matrica slova (engl. *P300 matrix speller*) koju su predstavili Farwell i Donchin (1988). U ovoj paradigmi korisniku je predočena matrica sa 36 simbola (slika 6.2) te ga se upućuje da se fokusira na jedan od simbola. Tada se naizmjence pale redci i stupci te se istovremeno obrađuje EEG signal u potrazi za P300 valom koji se javlja kada upali redak odnosno stupac u kojem se nalazi fokusirani simbol. Zbog postizanja robusnosti pokus se ponavlja nekoliko puta.



Slika 6.2: P300 matrica slova, izvor: internet

U okviru ovog rada napravljena su dva ROS čvora u svrhu generiranja podražaja za BCI. Jedan od njih je generator SSVEP podražaja tako da preklapa detektirane objekte obojanim kvadratima, a drugi izrezuje detektirane objekte iz RGB slike te ih raspodjeljuje po ekranu tako da je uz njih, na rub ekrana, moguće postaviti LE diode za generiranje podražaja.

6.1. ROS čvor: generator SSVEP podražaja nad predmetima u sceni

Generator SSVEP podražaja nad predmetima u sceni prototipni je čvor za generiranje podražaja za sučelje mozak-računalo. Kod pozivanja čvora moraju se predati slijedeći parametri i teme:

Registracija dubine parametar tipa boolean koji označava da li je ili nije uključena registracija dubine.

RGB slika tema na kojoj se objavljuje slika s RGB kamere senzora.

Informacije o RGB kameri tema na kojoj se objavljuju parametri RGB kamere.

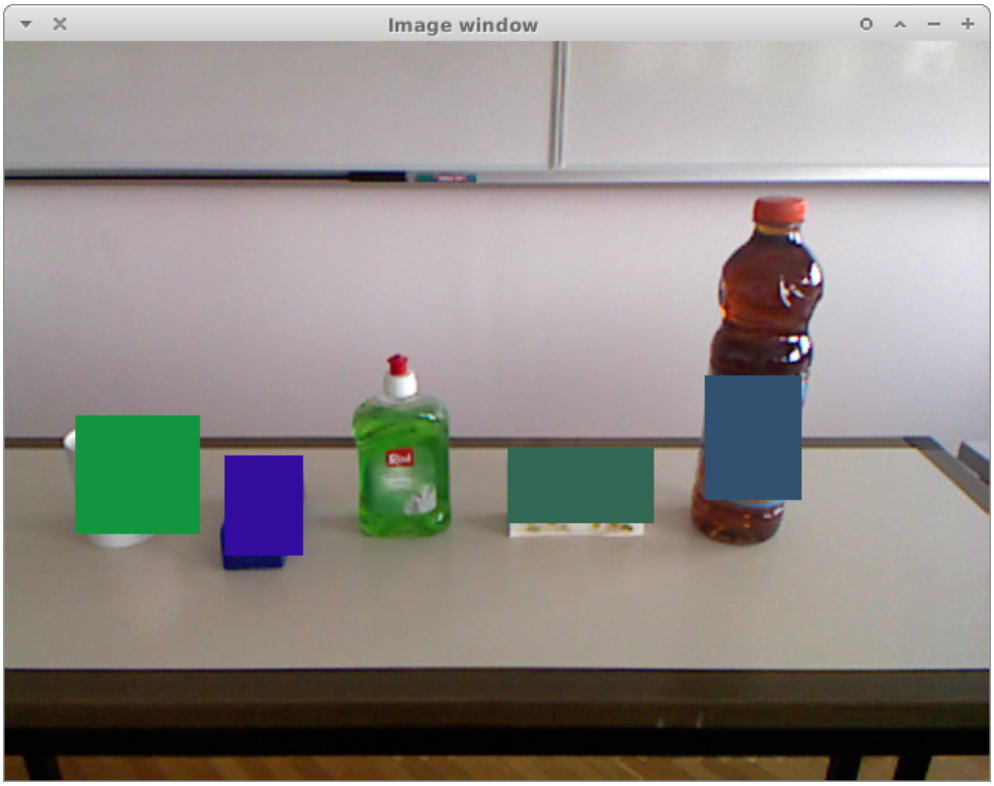
Oznake segmentacije tema na kojoj se objavljuju oznake segmentiranih objekata.

Ovaj čvor radi tako da pozove servis za segmentaciju scene iz *Tabletop Object Recognition* paketa koji vraća segmentirane objekte u obliku oznaka (engl. *markers*) koji se tada, ukoliko je potrebno (ovisi o parametru za registraciju dubine), transformiraju u koordinatni sustav RGB slike te se izračunava kvadrat koji prekriva svaki pojedini objekt. Na slici 6.3 prikazano je sučelje generatora SSVEP podražaja. Pošto je ovo prototipna verzija sučelja postavljeni su samo obojani kvadrati iznad segmentiranih objekata, a ne šahovnica kao što je to uobičajeno. Također se i ovdje može uočiti mali pomak kvadrata u odnosu na položaj objekata, a razlog tome je nesavršena definicija transformacije između koordinatnih sustava dvaju kamera.

Najveći problem kod korištenja SSVEP podražaja sa zaslona računala je stalnost frekvencije koju je teško postići bez izravnog pristupa sklopovlju grafičke kartice, zbog toga je odlučeno koristiti LE diode za pobudu. Također se iz daljnjih konzultacija s kolegama koji se bave obradom EEG signala, dodatno saznalo se da podražaji moraju biti udaljeni na ekranu zbog interferencija te je odlučeno da se koristi scena s maksimalno tri objekta koji će biti što je moguće više udaljeni na ekranu. Zbog navedenih zahtjeva napušten je daljnji razvoj ovog čvora te je implementiran novi čvor koji je prezentiran u nastavku.

6.2. ROS čvor: prikaz detektiranih objekata

Nakon prve verzije generatora SSVEP podražaja (slika 6.1) odustalo se od generiranja podražaja izravno na zaslonu računala te je odlučeno da će oni biti generirani pomoću



Slika 6.3: Generator SSVEP podražaja nad predmetima

LE dioda koje će biti pričvršćene na rub zaslona računala. Za tu svrhu trebalo je odrediti područja RGB slike na kojima se nalaze objekti, što je učinjeno kao i kod prethodnog čvora, te ih izrezati i rasporediti na zaslonu tako da budu maksimalno udaljeni jedan od drugoga. Za razliku od prethodnog ovaj čvor implementira servis kojemu se kod pokretanja predaju slijedeći parametri i teme da bi ispravno funkcionirao:

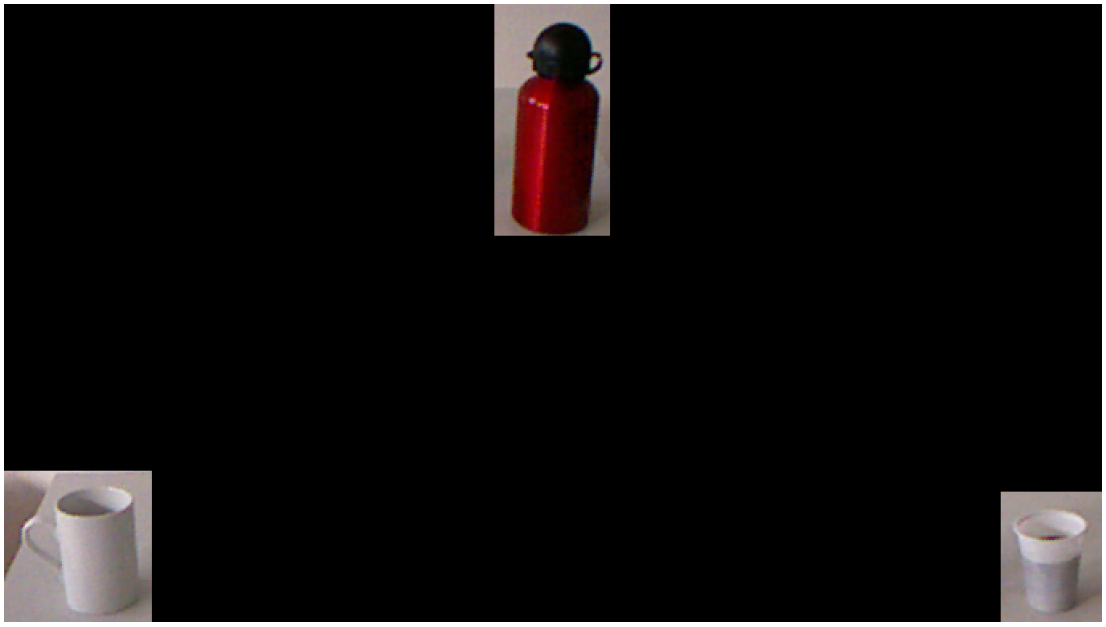
Ime servisa odabir naziva servisa kojim će se pokrenuti izrezivanje objekata iz RGB slike te ih naposljetku i prikazati.

RGB slika tema na kojoj se objavljuje slika s RGB kamere.

Informacije o RGB kameri tema na kojoj se objavljuju parametri RGB kamere.

Simulation zastavica koja označava da li je pokrenuta simulacija ili se koriste podaci sa stvarnog senzora. Ovo se koristi zbog toga što se mora koristiti drugačija transformacija oblaka točaka ako se koristi simulacija u odnosu na stvarni senzor.

Ovaj čvor je nešto napredniji od prethodnog te mu se podaci predaju preko poziva servisa čija definicija se može vidjeti u bloku 6.1. Kao što je vidljivo iz definicije servisa čvor koristi segmentirane klastere iz oblaka točaka koji se tada transformiraju



Slika 6.4: Sučelje za prikaz izlučenih objekata

u točke na RGB slici na temelju čega se izrezuju objekti iz slike. Tako izrezani objekti slažu se na zaslon računala kao što je prikazano na slici sučelja (slika 6.4). Trenutno sučelje može prikazati do najviše tri objekta u jednom trenutku, ako bi se u sceni nalazilo više od tri objekta tada bi jedna pozicija na sučelju mogla imati ulogu „gumba“ na slijedeći zaslon s objektima.

Listing6.1 Definicija servisa za izlučivanje objekata iz RGB slike

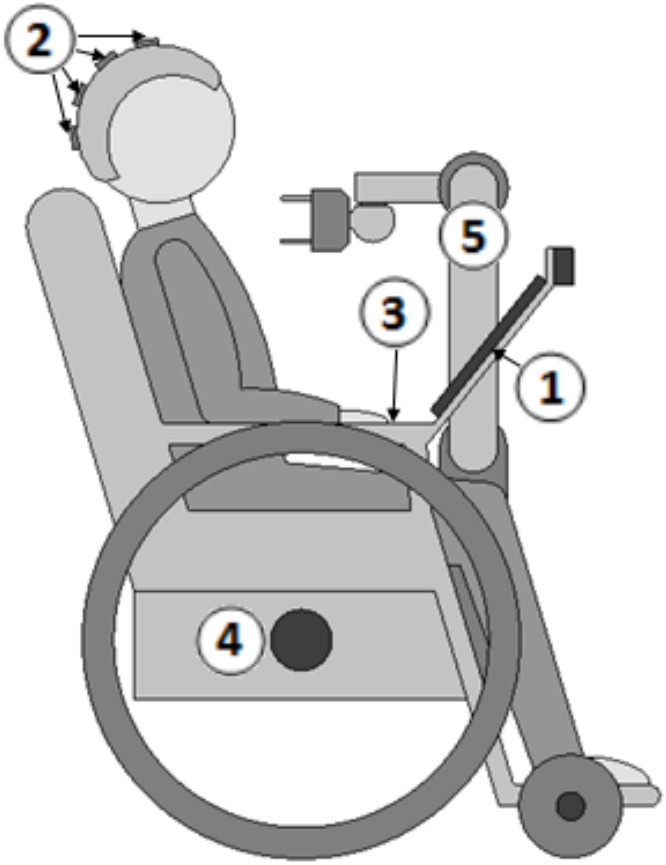
```
# list of point cloud clusters
sensor_msgs/PointCloud[] clusters
---
bool successfull
```

7. Manipulacija objektima pomoću sučelja mozak-računalo

U prethodnim poglavljima detaljno su opisane metode za prepoznavanje i manipulaciju objektima te čvorovi za generiranje podražaja odnosno prikaz objekata na sceni za sučelje mozak-računalo. U ovom poglavlju opisuje se osnovna ideja ovog rada te mogućnost njezinog ostvarenja na temelju ovdje predstavljenih metoda i BCI čvorova.

Osnovna ideja upotrebe sustava za manipulaciju s BCI sustavom je pomoć ljudima s invaliditetom pomoću mobilnog robota (Chen et al., 2013) ili modificiranih invalidskih kolica (Jerbić et al., 2013). Oba ova istraživanja bave se intenzivno problematikom komunikacijskog sučelja te predlažu različita rješenja. U Jerbić et al. (2013) predlaže se BC sučelje implementirano u sklopu ovog rada, a ideja cijelog sustava prikazan je na slici 7.1. Brojevi na slici odnose se na slijedeće dijelove sustava: (1) LCD ekran koji prikazuje BCI sučelje sa slike 6.4 i LE diodama na rubovima za generiranje podražaja, (2) senzorski čvorovi za snimanje EEG signala te predajnik za unutar-tijelesnu komunikaciju (engl. *Intrabody Communication (IBC) Transmitter*), (3) primatelj IBC komunikacije sa sučeljem na (4) sustav za procesiranje EEG signala te (5) robotska ruka za manipulaciju odabranim objektom.

U ovom radu prezentiran je jedan veliki dio nužan za izgradnju sustava predloženog u Jerbić et al. (2013), a to je prepoznavanje, prikaz te manipulacija objektima. Za zatvaranje kompletnog kruga još nedostaje čvor za procesiranje EEG signala unutar ROS-a koji bi sustavu za manipulaciju proslijedio poruku o odabranom objektu.



Slika 7.1: Prijedlog upotrebe BCI sustava za pomoć pri manipulaciji osobama s invaliditetom, izvor: Jerbić et al. (2013)

8. Zaključak

U ovom radu prezentirane su i testirane metode za prepoznavanje objekata na ravnoj podlozi i izlučivanje točaka hvata implementirane u programskom sustavu ROS. Metode su testirane u simulacijskom i stvarnom okruženju uz korištenje *Microsoft Xbox 360 Kinect* senzora te *Schunk Powerball* robotske ruke sa *Schunk WSG-50* hvataljkom. Uz to implementirana su dva čvora za prikaz detektiranih objekata na sceni u svrhu generiranja podražaja za sučelje mozak-računalo i regulator hvata za hvataljku *Schunk WSG-50*.

Rezultati dobiveni metodama za prepoznavanje objekata su razočaravajući jer se niti jedna od četiri testirane metode nije pokazala dovoljno pouzdanom za prepoznavanje. Bez obzira na to uspješno je izlučen hvat i ostvareno podizanje objekta u simulacijskom okruženju, a sve na temelju kvalitetnog čvora *Tabletop Segmentation* za izdvajanje klastera točaka za svaki objekt na sceni. Također je važno napomenuti da se i novo-implementirani čvorovi temelje na ovoj segmentaciji na temelju koje određuju površinu koju objekt zauzima na RGB slici.

Uz to ovaj rad može poslužiti kao početna točka za svakoga koga zanima prepoznavanje i manipulacija objekata u programskom sustavu ROS jer nudi iscrpan pregled mogućnosti i smjernica kako koristiti postojeće sustave.

LITERATURA

Ros: Openni Launch Tutorials: Quickstart: Registration: matching depth to color, 2012. URL http://www.ros.org/wiki/openni_launch/Tutorials/QuickStart#Registration:_matching_depth_to_color. 18

Pcl - Point Cloud Library, 6 2013. URL http://pointclouds.org. 6, 34

P. J. Besl i H. D. Mckay. A method for registration of 3-D shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256, 1992. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=121791. 9

Gary Bradski i Adrian Kaehler. *Learning OpenCV*. O'Reilly Media Inc., 2008. URL http://oreilly.com/catalog/9780596516130. 6

Gilles Burel i Hugues Henocq. 3d invariants and their application to object recognition. 45(1), 7 1995. 4

M. Byczuk, P. Poryzała, i A. Materka. On diversity within operators' EEG responses to LED-produced alternate stimulus in SSVEP BCI. 2012. 37

Tiffany Chen, Matei Ciocarlie, Steve Cousins, Phillip M. Grice, Kelsey Hawkins, Kaijen Hsiao, Charlie Kemp, Chih-Hung King, Daniel Lazewatsky, Adam Eric Leeper, Hai Nguyen, Andreas Paepcke, Caroline Pantofaru, William Smart, i Leila Takayama. Robots for Humanity: A Case Study in Assistive Mobile Manipulation. *IEEE Robotics & Automation Magazine, Special issue on Assistive Robotics*, 20, 2013. URL http://www.willowgarage.com/sites/default/files/Chen_RFH_ram_2013.pdf. 3, 42

Yizong Cheng. Mean Shift, Mode Seeking, and Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995. doi: http://dx.doi.org/10.1109/34.400568. 5

Sachin Chitta. PR2 Kinematics, 2013. URL http://www.ros.org/wiki/pr2_kinematics. 15

Matei Ciocarlie. Household Objects Database, 2012. URL http://www.ros.org/wiki/household_objects_database. 14, 15, 22

Matei Ciocarlie. Tabletop Collision Map Processing, 2013. URL http://www.ros.org/wiki/tabletop_collision_map_processing. 15, 16

Matei Ciocarlie i Kaijen Hsiao. Object Manipulation, 2012. URL http://www.ros.org/wiki/object_manipulation. 14, 53

Matei Ciocarlie i Kaijen Hsiao. Object Manipulator, 2013. URL http://www.ros.org/wiki/object_manipulator. 16

Matei Ciocarlie, Kaijen Hsiao, E. Gil Jones, Sachin Chitta, Radu Bogdan Rusu, i Ioan Alexandru Sucan. Towards reliable grasping and manipulation in household environments. New Delhi, India, 12 2010. 6, 8, 14, 16

Alvaro Collet Romea, Dmitry Berenson, Siddhartha Srinivasa, i David Ferguson. Object recognition and full pose registration from a single image for robotic manipulation. U *IEEE International Conference on Robotics and Automation (ICRA '09)*, May 2009. 5

Jennifer L Collinger, Brian Wodlinger, John E Downey, Wei Wang, Elizabeth C Tyler-Kabara, Douglas J Weber, Angus Jc McMorland, Meel Velliste, Michael L Boninger, i Andrew B Schwartz. High-performance neuroprosthetic control by an individual with tetraplegia. *The Lancet*, 6736(12):1–8, 12 2012. ISSN 01406736. doi: 10.1016/S0140-6736(12)61816-9. 1

Ioan A. Şucan, Mark Moll, i Lydia E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. doi: 10.1109/MRA.2012.2205651. http://ompl.kavrakilab.org. 16

Mehmet Dogar, Kaijen Hsiao, Matei Ciocarlie, i Siddhartha Srinivasa. Physics-based grasp planning through clutter. U *Robotics: Science and Systems (RSS)*, 2012. URL http://www.roboticsproceedings.org/rss08/p08.pdf. 5

Lawrence Ashley Farwell i Emanuel Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and clinical Neurophysiology*, 70(6):510–523, 1988. 38

Martin A. Fischler i Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Technical Report 213, AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025, Mar 1980. 5

Timothy Gatzke, Cindy Grimm, Michael Garland, i Steve Zelinka. Curvature maps for local shape comparison. U *In Shape Modeling International*, stranice 244–256, 2005. 4

Jared Glover, Gary Bradski, i Radu Bogdan Rusu. Bingham mixture models of quaternions for object orientation estimation. U *Robotics Science and Systems (RSS)*, Los Angeles, CA, USA, 07/2011 2011. 5

Corey Goldfeder, Matei Ciocarlie, Hao Dang, Jaime Peretzman, i Peter Allen. Data-driven grasping with partial sensor data. U *IROS*, St. Louis, MO, 10/2009 2009. 5

S. Hinterstoisser, S. Benhimane, i N. Navab. N3m: Natural 3d markers for real-time object detection and pose estimation. 2007. 4

S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, i N. Navab. Dominant orientation templates for real-time detection of texture-less objects. 2010. 4

S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, i V. Lepetit. Gradient response maps for real-time detection of texture-less objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012a. 12

S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, , i N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. 2012b. 4

S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, i V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. 2011. vi, 1, 3, 4, 11, 12, 36

Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, i Sven Behnke. Real-time plane segmentation using rgb-d cameras. U *RoboCup Symposium*, 2011 2011. 5

S. Holzer, S. Hinterstoisser, S. Ilic, i N. Navab. Distance transform templates for object detection and pose estimation. 2009. 4

Andrew Howard, Nate Koenig, John Hsu, i Mihai Dolha. Gazebo, 6 2013. URL http://gazebosim.org/about.html. 22

Kaijen Hsiao. PR2 Gripper Grasp Planner Cluster, 2012. URL http://www.ros.org/wiki/pr2_gripper_grasp_planner_cluster. 15

Kaijen Hsiao i Matei Ciocarlie. PR2 Object Manipulation, 2012. URL http://www.ros.org/wiki/pr2_object_manipulation. 15, 54

Kaijen Hsiao, Sachin Chitta, Matei Ciocarlie, i E. Gil Jones. Contact-reactive grasping of objects with partial shape information. *IROS*, 10 2010. 15

John Hsu. Simulator gazebo, 2013. URL http://www.ros.org/wiki/simulator_gazebo. 22

Itseez. Open computer vision library, 2013. URL http://opencv.org. 6, 11, 34

Ana Branka Jerbić, Željka Lučev, Srećko Jurić-Kavelj, Filip Melinščak, Josip Lončar, Denis Štogl, Mario Cifrek, i Ivan Petrović. Concept of an intrabody networked brain-computer interface controlled assistive robotic system. Submited for RAAD 2013, 2013. vi, 16, 24, 37, 42, 43

Andrew E. Johnson i Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 21(5):433–449, 1999. 4

E. Gil Jones. Arm Navigation, 2013. URL http://www.ros.org/wiki/arm_navigation. 14, 53

Ulrich Klank, Dejan Pangercic, Radu Bogdan Rusu, i Michael Beetz. Real-time cad model matching for mobile manipulation and grasping. U *The 9th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Paris, France, 12/2009 2009. URL http://files.rbrusu.com/publications/Klank09Humanoids.pdf. 3, 4

Ilya Lysenkov, victor Eruhimov, i Gary Bradski. Recognition and pose estimation of rigid transparent objects with a kinect sensor. U *Proceedings of Robotics: Science and Systems*, Sydney, Australia, 7 2012. 13, 20

Dennis J McFarland, William A. Sarnacki, i Jonathan R. Wolpaw. Electroencephalographic (EEG) control of three-dimensional movement. *Journal of neural engineering*, 7(3):036007, Lipanj 2010. ISSN 1741-2552. doi: 10.1088/1741-2560/7/3/036007. 1

Patrick Mihelich i Julius Kammerl. Openni Launch, 2013. URL `http://ros.org/wiki/openni_launch`. 18

Patrick Mihelich, Suat Gedikli, Radu Bogdan Rusu, i Julius Kammerl. Openni Camera, 2013. URL `http://ros.org/wiki/openni_camera`. 18

Marius Muja i Matei Ciocarlie. Tabletop object detector, 2013. URL `http://www.ros.org/wiki/tabletop_object_detector/`. 1, 8, 9, 25

Marius Muja, Radu Bogdan Rusu, Gary Bradski, i David Lowe. Rein - a fast, robust, scalable recognition infrastructure. U *ICRA*, Shanghai, China, 09/2011 2011. URL `https://www.willowgarage.com/sites/default/files/icra11.pdf`. 5

Marius Muja, Radu Bogdan Rusu, Gary Bradski, i David Lowe. REcognition INfrastructure (rein), 2013. URL `http://www.ros.org/wiki/rein`. 5, 8

M. Quigley i Inc. Willow Garage. Ros - Robot Operating System, 2013. URL `http://ros.org`. 7

Columbia University Robotics Lab, Computer Science Department. Graspit!, 2013. URL `http://www.cs.columbia.edu/~cmatei/graspit/`. 15

E. Rublee, T. Straszheim, i V. Rabaud. Object recognition, 2013. URL `http://www.ros.org/wiki/object_recognition`. 8

Radu Bogdan Rusu i Steve Cousins. 3d is here: Point Cloud Library (PCL). U *International Conference on Robotics and Automation*, Shanghai, China, 2011 2011. 6

Radu Bogdan Rusu, Nico Blodow, i Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. U *The IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, 05/2009 2009a. URL `http://files.rbrusu.com/publications/Rusu09ICRA.pdf`. 4

Radu Bogdan Rusu, Andreas Holzbach, Michael Beetz, i Gary Bradski. Detecting and segmenting objects for mobile manipulation. U *ICCV, S3DV Workshop*, 2009b. 4

Radu Bogdan Rusu, Eddie Cohen, Tomoto Shimizu Washio, Matt Bell, Eray Berger, i Robert Wang. Openni: The standard framework for 3d sensing, 2013. URL http://www.openni.org/. 18

SCHUNK GmbH & Co. KG. Powerball lightweight arm lwa 4.6 6-axes-lightweight roboter for mobile handling, 6 2013. URL http://www.schunk.com/schunk/schunk_websites/products/latest_products_detail.html?article_id=21200&country=INT&lng... 20

R. Smits. KDL: Kinematics and Dynamics Library. http://www.orocos.org/kdl, 2013. 15, 16, 27

Ioan A. Sucan i Sachin Chitta. Moveit!, 2013. URL http://moveit.ros.org/wiki/MoveIt! 14

Inc. Willow Garage. Ecto - A C++/Python Computation Graph Framework, 2013a. URL http://plasmodic.github.io/ecto/. 5, 10

Inc. Willow Garage. Object Recognition Kitchen, 2013b. URL http://wg-perception.github.io/object_recognition_core/. 30

Inc. Willow Garage. Object recognition kitchen: Capture, 2013c. URL http://wg-perception.github.io/capture/index.html#ork-capture. 10, 11, 31

Inc. Willow Garage. Object recognition kitchen: Object recognition using line-mod, 2013d. URL http://wg-perception.github.com/linemod/index.html#line-mod. 11

Inc. Willow Garage. Object recognition kitchen: Ros integration, 2013e. URL http://wg-perception.github.io/object_recognition_ros/index.html#ros. 10

Inc. Willow Garage. Object Recognition Kitchen: Reconstruction, 2013f. URL http://wg-perception.github.io/reconstruction/index.html#reconstruction. 10, 34

Inc. Willow Garage. Object recognition kitchen: Random view generator, 6 2013g. URL http://wg-perception.github.io/ork_renderer/index.html#renderer. 11, 36

Inc. Willow Garage. Object recognition kitchen: Textured object detection, 6 2013h. URL http://wg-perception.github.com/tod/index.html#tod. 1, 13

Inc. Willow Garage. Object recognition kitchen: Tabletop object recognition, 6 2013i. URL http://wg-perception.github.com/tabletop/index.html#tabletop. 12

Inc. Willow Garage. Object recognition kitchen: Training, 2013j. URL http://wg-perception.github.io/object_recognition_core/training/training.html#training. 34

Inc. Willow Garage. Object recognition kitchen: Recognition of transparent objects, 6 2013k. URL http://wg-perception.github.com/transparent_objects/index.html#transparent-objects. 13

Detekcija i prepoznavanje objekata u svrhu izlučivanja točaka hvata i generiranja podražaja za sučelje mozak-računalo

Sažetak

U ovom radu prezentirane su i testirane metode za prepoznavanje objekata na ravnoj podlozi i izlučivanje točaka hvata implementirane u programskom sustavu ROS. Metode su testirane u simulacijskom i stvarnom okruženju za što je manipulacijski cjevovod prilagođen korištenoj opremi. Naposlijetku je uspješno izvršeno hvatanje i podizanje objekta u simulacijskom okruženju temeljeno na algoritmu za detekciju objekata i online metodi za izlučivanje točaka hvata. Uz to implementirana su dva ROS čvora za prikaz objekata na sceni u svrhu generiranja podražaja za BCI te je predložena mogućnost korištenja manipulacije objekata i novoimplementiranih sučelja u istom sustavu.

Ključne riječi: detekcija objekta, prepoznavanje objekta, segmentacija scene, izlučivanje točaka hvata, manipulacija objektima, sučelje mozak-računalo, podražaji za sučelje mozak-računalo, operacijski sustav za robote (ROS)

Object detection and recognition for grasping point extraction and excitation generation for brain-computer interface paradigm

Abstract

In this work are presented and tested methods for object recognition on flat surface and methods for grasping point extraction implemented in ROS. Methods are tested in simulated and real environment and manipulation pipeline is adapted to work with used equipment. Finally, grasping and lifting of object is achieved in simulation environment based on algorithm for object detection an online method for grasping point extraction. Besides that two new ROS nodes for representation of object in a scene are implemented in order to generate excitation for BCI and one use possibility of object manipulation and new implemented interfaces in the same system is proposed.

Keywords: object detection, object recognition, scene segmentation, grasping point extraction, object manipulation, brain-computer interface, excitation for brain-computer interface, BCI, Robots Operating System (ROS)

Dodatak A
Popis repozitorija

bci_visual_stimuli

Repozitorij sa svim čvorovima koji su implementirani u sklopu ovoga rada. Također se uz to nalaze i datoteke za pokretanje svih potrebnih čvorova za simulaciju i korištenje algoritama za prepoznavanje u stvarnosti.
`https://bitbucket.org/destogl/bci_visual_stimuli` (privatni)

robotnik-powerball-ros-pkg

Klon robotnik-powerball-ros-pkg repozitorija s Google Code sustava (`http://code.google.com/p/robotnik-powerball-ros-pkg/`). Repozitorij implementira upravljački program za korištenje *Schunk Powerball* ruku s programskim paketom ROS (trenutno samo simulacija).
`https://github.com/jksrecko/robotnik-powerball-ros-pkg` (javni)

wsg50-ros-pkg

Klon wsg50-ros-pkg repozitorija s Google Code sustava (`http://code.google.com/p/wsg50-ros-pkg/`). Repozitorij implementira upravljački program za korištenje *Schunk WSG-50* hvataljke s programskim paketom ROS.
`https://github.com/jksrecko/wsg50-ros-pkg` (javni)

powerball_robotnik_arm_navigation

Sadrži konfiguracije i pokretačke datoteke za Schunk Powerball ruku kada se koristi *Arm Navigation* cjevovod (vidi Jones (2013)).
`https://bitbucket.org/jksrecko/powerball_robotnik_arm_navigation` (javni)

object_manipulation

Implementira stog *Object Manipulation* (više u Ciocarlie i Hsiao (2012)). U

ovom radu se koristi grana *generalisation*.
`https://github.com/jksrecko/object_manipulation.git` (javni)

pr2_object_manipulation

Implementira stog *PR2 Object Manipulation* (više u Hsiao i Ciocarlie (2012)).
U ovom radu se koristi grana *generalisation*.
`https://github.com/destogl/pr2_object_manipulation` (javni)

household_mesh_exporter

Sadrži čvorove za izvoz mreža objekata iz *„Household“* baze te čvor za konverziju između tipova podataka *PointCloud* i *PointCloud2* što je potrebno za simulaciju.
`https://bitbucket.org/jksrecko/household_mesh_exporter` (javni)

ork_fer

Repozitorij s konfiguracijskim i pokretačkim datotekama za rad s *Object Recognition Kitchen* sustavom u okviru ovog rada.
`https://bitbucket.org/destogl/ork_fer` (privatni, catkin sustav prevođenja)