Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Aleksandar Dovenski

Implementacija funkcije kazanja na predmete z robotsko roko

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Danijel Skočaj

Ljubljana 2012

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00264/2012
Datum: 11.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ALEKSANDAR DOVENSKI**

Naslov: **IMPLEMENTACIJA FUNKCIJE KAZANJA NA PREDMETE Z ROBOTSKO ROKO**

IMPLEMENTATION OF POINTING AT OBJECTS WITH A ROBOTIC ARM

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Približevanje predmetu, kazanje na predmet ter prijemanje predmetov so osnovne funkcije robotskega sistema, ki vsebuje dovolj zmogljive senzorje in robotski manipulator. V diplomski nalogi implementirajte funkcijo kazanja na predmete z robotsko roko. Sistem naj zazna predmet na osnovi procesiranja zaznanega oblaka 3D točk, nato pa naj z uporabo inverzne kinematike usmeri robotsko roko v smeri predmeta. Celoten sistem razvijte v robotskem operacijskem sistemu ROS v simulacijskem okolju na način, ki omogoča neposreden prenos implementacije na realen robotski sistem sestavljen iz senzorskega sistema temelječega na barvno-globinski kameri Kinect in robotski roki Katana HD6M.

Mentor:

doc. dr. Danijel Skočaj

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Aleksandar Dovenski, z vpisno številko **63080470**, sem avtor diplomskega dela z naslovom:

Implementacija funkcije kazanja na predmete z robotsko roko

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Danijela Skočaja,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 8. oktobra 2012 Podpis avtorja:

Zahvala

Rad bi se zahvalil svojemu mentorju, doc. dr. Danijelu Skočaju za vso pomoč, potrpežljivost in usmerjavanje pri izdelavi diplomske naloge.

Iskreno se zahvaljujem tudi svojim staršem, ki so me nenehno in brezrezervno podpirali na vseh možnih načinih tako v času študija kot tudi v celotnem mojem življenju. Hvaležen sem tudi sestri, starim staršem in vsem ostalim članom moje družine za njihovo zaupanje in spodbudo, ki so mi ju vedno dajali.

Najlepša hvala tebi Magde, ker si ostala vztrajna in potrpežljiva po vsem tem času. Hvala, da si me navdihnila, da naredim naslednji korak, kot si že neskončnokrat prej.

Hvala tudi mojim prijateljem, ki so me motivirali in spodbujali na njihov lasten način.

Zahvaljujem se tudi vsem članom Laboratorija za umetne vizualne spoznavne sisteme in Fakulteti za računalništvo in informatiko, pa tudi vsem ostalim, ki so mi tako ali drugače pomagali v času izdelave diplomskega dela.

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Uporabljena orodja **5**

2.1 Robot Operating System 5

2.2 Simulator Gazebo 9

2.3 Kinect 12

2.4 Robotska roka Katana 13

3 Zaznavanje predmetov **17**

3.1 Zemljevid globine 17

3.2 Podvzorčanje oblakov točk 18

3.3 Odkrivanje vodoravnih ravnin 19

3.4 Gručenje točk 20

4 krmiljjenje robotske roke **23**

4.1 Parametri robota Katana 23

4.2 Direktna kinematika 25

4.3 Inverzna kinematika 26

5 Sklepne ugotovitve **35**

Povzetek

Tema diplomske naloge je kazanje na predmete z uporabo robotske roke Katana. Robot samostalno najde predmet v svoji okolici s pomočjo kamere in nanj kaže z konicom svoje roke.

Izdelava diplomske naloge je potekala v *Robot Operating System* (v nadaljevanju ROS), ogrodje za razvoj programske opreme za robote. Za simulacijo robotske roke in njegovega okolja je bil uporabljen *Gazebo*, tridimenzionalen simulator za različne vrste robotov. Za odrkivanje predmeta na mizi uporabljamo globinsko tipalo Kinect, ki nam daje oblak točk.

Ko najde predmet, robot poskuša najti način, da z vrtenjem svojih sklepov pokaže natančno na ta predmet. Naslednji korak je načrtovanje gibanja sklepov, da bi lahko roka spremenila svoje stanje iz trenutnega v želeno. Ko se to zgodi, simulator Gazebo prikaže spreminanje stanj roke.

Ključne besede: Robot Operating System, ROS, Katana, robot, robotska roka, Gazebo, Kinect, inverzna kinematika, direktna kinematika, oblak točk.

Abstract

The subject of the gradation thesis is pointing at an object with the robotic arm Katana. The robot autonomously detects the object in its environment, using a camera and points at it with his end effector.

The system was developed in *Robot Operating System*, a software framework for developing robotics software. The robotical arm, alongside its environment, was simulated using *Gazebo*, a three-dimensional multi-robot simulator. The objects on the table were detected with the usage of the depth sensor Kinect, which provides depth data in the form of a point cloud.

After the object is detected, the searches for an inverse kinematics solution to move the end effector towards the object, pointing at it. After such a solution is found, a motion planning algorithm for moving the robotic arm from its current state, to the desired one is run. Lastly, Gazebo plays the joints' trajectories, thus simulating the robotic arm's movements.

Keywords: Robot Operating System, ROS, Katana, robot, robotic arm, Gazebo, Kinect, inverse kinematics, forward kinematics, point cloud.

Poglavje 1

Uvod

Glavni cilj diplomske naloge je uporaba robotske roke za rokovanje z osnovnimi predmeti. Samostojno manipuliranje s predmeti je zelo koristno v vsakdanjem življenju. Robotske roke so nepogrešljive v sodobnih in tehnološko napredhin tovarnah. Čeprav so roke z natančno določenimi nalogami veliko bolj pogoste, imajo velik potencial tudi tiste, ki so prilagodljive in se zavedajo svoje okolice.

Programljive robotske roke bi bile zelo uporabne tudi v gospodinjenju. Z dobro programsko in strojno opremo v ozadju bilo bi možno narediti karkoli. Robot bi se lahko naučil posesati, počistiti sobe, pospraviti vse predmete, ki niso na svojem mestu. Poleg tega, robot bi nam lahko služil kot strežaj, da nam prinaša stvari in jih potem vrne nazaj, ko jih ne rabimo več, ali pa jih vrže v koš za smeti, če mu tako rečemo.

Robot, ki je osrednjega pomena za diplomskego delo, je statičen v prostoru. Zaradi tega robotska roka lahko deluje samo nad predmeti, ki se nahajajo v njegovi neposredni bližini, oziroma v njegovem delovnem obsegu. Robotsko roko lahko postavimo na mobilno platformo in s tem uslovno neskončnokrat povečamo njen delovni prostor. Za ta namen bi programska oprema, ki smo jo razvili, bila temelj za nadaljni razvoj.

Cilj naloge je razvoj sistema za kazanje na predmete z robotsko roko Katana HD6M, v programskem ogrodju ROS, ki omogoča prenos programske

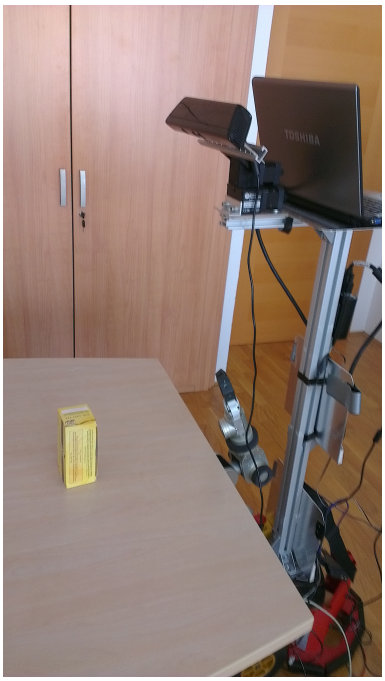
opreme iz simulatorja v realni robotski sitem. Za ta namen je potrebno uporabljati barvno globinsko kamero Kinect, ki zgradi tridimenzionalen oblak točk. S procesiranjem podatkov iz tega oblaka naj program detektira predmet, ki se nahaja na mizi. Potem naj bi najdel rešitev za inverzno kinematiko in premakne roko tako, da ta z svojim konicom kaže na predmet.

Programsko ogrodje, ki smo ga izbrali za uporabo in nadzor robota, se imenuje *Robot Operating System* (v nadaljevanju *ROS*). ROS se uporablja za komunikacijo med roboti in računalniki, oziroma za razvoj programske opreme za robote. V ROSu smo uvozili gonilnike za dejansko robotsko roko, ki smo jo uporabljali(*Katana 6m*). Postavili smo robotsko roko v navidezni svet skupaj s pisarno mizo in predmet, ki se nahaja na mizi in služi za testiranje pravilnosti delovanja roke. Ta svet zgradimo s pomočjo simulacijskega orodja *Gazebo*, ki je simulator za robote, njihove strojne opreme in njihova okolja.

V svetu smo dodali barvno globinsko kamero *Kinect*, ki nam omogoča odkrivanje predmeta. V našem sistemu kamero uporabljamo na dva načina, in sicer izhod prave kamere Kinect, ki jo imamo na razpolago, in simulirano kamero s pomočjo Gazeba. Za lokacijo simulirane kamere smo izbrali točko, ki se nahaja neposredno nad robotsko roko, kot je tudi na dejanski roki. Kamero smo zavrteli tako, da je usmerjena proti mizi, na kateri se naš nahaja predmet. Z uporabo izhoda postavljene kamere določimo lego predmeta in nato kličemo funkcijo, ki izračuna, kako lahko roka pride v to lego. To funkcijo smo napisali za računanje *inverzne* kinematike za robotsko roko Katana.

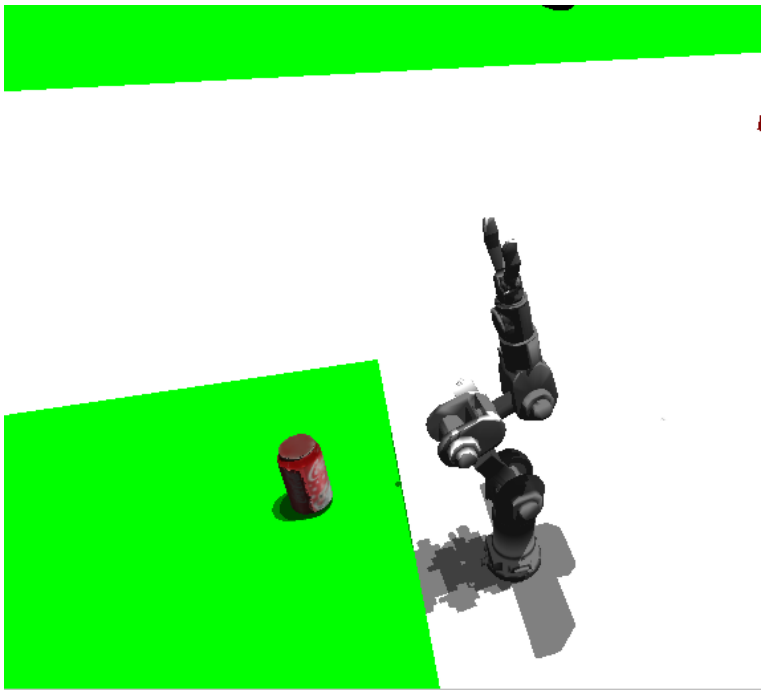
Robotska roka Katana, ki se uporablja v laboratoriju, je trenutno v okvari in zaradi tega moramo vse funkcije robotske roke implementirati v simulatorju. Z druge strani, zaradi težave z grafično kartico, ki ne omogoča simulacije barvno globinske kamere Kinect, smo morali globinske podatke zajemati s pravo kamero. Zaradi tega je spoznavalni del sistema realiziran na realnemu robotu, kinematski pa na simulatorju. Na Sliki 1.1 je prikazan dejanski scenarij, ki ga obdelujemo. Na Sliki 1.2 je prikazan scenarij v simulatorju

Gazebu.



Slika 1.1: Robot Katana v Laboratoriju za umetne vizualne spoznavne sisteme.

V Poglavju 2 bomo spoznali orodja in metode, ki smo jih uporabljalji za rešavanje našega problema. Poglavju 3 vsebuje opis, kako uporabljamo kamero Kinect za odkrivanje predmeta na mizi. V Poglavju 4 opisujemo postopek upravljanja robotske roke za kazanje na predmet. Na koncu sledi zaključek diplomskega dela.



Slika 1.2: Ciljni scenarij predstavljen v Gazebo.

Poglavje 2

Uporabljena orodja

Pri izdelavi diplomskega dela smo uporabljali različne metode in orodja. Z dobrim deležom uporabljenih orodij za potrebe iz področja robotike smo se prvič srečali in smo jih sčasoma raziskali in preučili. Celotno delo je potekalo v operativnem sistemu *Ubuntu*, oziroma v *Linux Mint*u, ki temelji na Ubuntu. Odločili smo se za Ubuntu zato, ker je najbolj podprt operativni sistem za uporabo ROSa.

2.1 Robot Operating System

Robot Operating System, ali Operacijski sistem za robote, je prvo in osnovno ogrodje, ki smo ga uporabljali za izdelavo diplomske naloge. ROS je odprtokodno programsko ogrodje, namenjo razvoju programske opreme za veliko število različnih robotov. Hkrati je tudi vmesnik med računalniki in roboti, ker omogoča nadzor delovanja in stanj robotov ter njihovih strojnih oprem prek računalnika.

Stanford Artificial Intelligence Laboratory je razvil ogrodje ROS, tedaj, leta 2007, z imenom switchyard. Od leta 2008 naprej ga razvija *Willow Garage*, laboratorij za raziskave robotike in razvoj strojne in programske opreme za robote. Glavni razlog nastanka ROS-a je bil ta, da je razvoj programov za robote težek, glede na nenehno progresijo robotike in njeno

povpraševanje. Izumitelji ROS-a so sklepali, da je to tako, ker se roboti med sabo zelo razlikujejo glede strojne opreme, ki jih imajo in namenov, za katere se uporabljajo. Zaradi tega so zasnovali ogrodje, ki vsebuje metode in orodja za razvoj robotskih programskih oprem vseh vrst in namenov[1].

Glavni cilji razvijalcev ROS-a so bili, da le-ta naj:[1]:

- omogoča povezavo vrste "vsak z vsakim";
- je utemeljen na orodja;
- dovoljuje uporabo več programskih jezikov;
- je "tanek", oz. lahko prilagodljiv;
- je brezplačen in odprtokoden.

Vsi ti cilji so se tudi izpolnili[1] :

- v času izvajanja programov večina procesov se prekrivajo in komunicirajo med sabo;
- namesto uporabe strogo določenih okoljij, ROS uporablja veliko število manjših orodij, ki se lažje nastavljajo glede na potrebe;
- omogočena je uporaba 4 jezikov (*C++*, *Python*, *Octave* in *LISP*);
- večina obstoječe programske opreme uporablja gonilnike in algoritme, ki se lahko uporabljajo za druge podobne namene;
- vsa koda ROS-a se dobi brezplačno pod licenco BSD.[1]

Osnovna arhitektura ROS-a je sestavljena iz *vozlišč* (*node*), *sporočil* (*message*), *tem* (*topic*) in *storitev* (*Services*). Vozlišča so osnovni procesi, ki delajo vse operacije. Lahko jih zaženemo bodisi neposredno iz ukazne vrstice z ukazom *rosrun*, bodisi iz dototeke za začetek procesov vrste *.launch* z *HTML* ukazom *<node/>*. Vozlišča med sabo komunicirajo s pošiljanjem sporočil, ki so spremnljivke določene vrste. Da bi prijemali sporočila, vozlišča se morajo

naročiti na ustrezno temo, ki vsebuje to sporočilo. Z druge strani, vozlišča, ki pošiljajo sporočila za ta namen, morajo sporočila objavljati za dano temo. Storitve so sestavljene iz par sporočil, ki ustrezajo zahtevam in odgovoru. Storitve imajo samo enega oglaševalca, za razliko od teme, ki jih lahko imajo veliko.[1]

ROS je bil prvotno razvit za podporo Stanfordskega robota z umetno inteligenco (skrajšano *STAIR*), mobilni robot, namenjen za pomoč doma ali v pisarni. Potem, ko so razvoj ROS-a prevzeli razvijalci iz Willow Garage, so za osnovo izbrali robot Personal Robot 2 (skrajšano PR2), in so na njem zasnovali svojo ogrodje. Tudi PR2, ki je prikazan na Sliki 2.1, je izdelek laboratorija Willow Garage. Vsebuje dve roki za manipuliranje s predmeti in strojno opremo za prepoznavanje predmetov, ki je sestavljena iz laserskih globinskih senzorjev in kamero. Zamišljen je, da bi se uporabljal kot izhodišče za razvoj programskih oprem za ostale robote.

Pakete, ki so že narejeni v ROS-u za krmiljjenje robota PR2, smo spreminjali in prilagajali za potrebe našega robota Katana in smo na ta način razvijali našo programsko opremo. Med ostalimi tukaj sodijo paketi za načrtovanje gibanja, za računanje inverzne kinematike, za simuliranje robotov in njihovih okolij, detekcijo trkov ipd. Vsi ti paketi vsebujejo algoritme iz določenih območij, ki se lahko implementirajo za potrebe našega problema.

2.1.1 Unified Robot Description Format (URDF)

Unified Robot Description Format(skrajšano URDF), je ROS-ov datotečni format jezika XML, ki se uporablja za predstavitev modela robota. Robotski model se predstavlja z uporabo petih elementov, ki natančno opisujejo robota.

Sklepi (angl. *joint*) so osnovni elementi robotov. Njihova glavna odlika je to, da je njihovo stanje spremenjlivo in s spreminjanjem tega stanja robot upravlja roko. V ROS-u obstaja šest vrst sklepov, med katerimi so najbolj pogosti rotacijski oz. vrtljivi (angl. revolute) in translacijski oz. prizmasti (angl. prismatic). Rotacijski se vrti okoli dane osi v danih mejah. V primer-

Slika 2.1: Robot vrste Personal Robot 2.
Vir: Willow Garage.

javi z njim translacijski sklepi drsijo v smeri dane osi v določenih mejah.

Model je element, ki opisuje dinamične in kinematske lastnosti robotskega modela. Ta element združuje povezave in tipala ter tako zgradi robota.

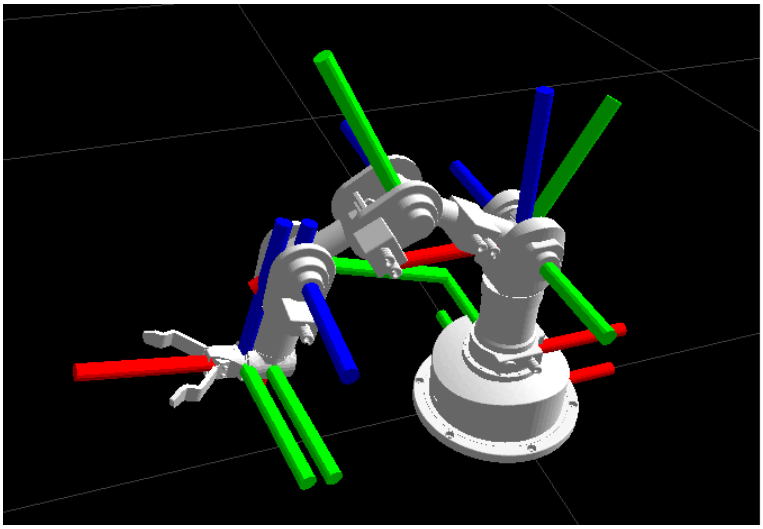
Povezave (angl. *link*) so toga telesa, ki imajo inercijo, težo in vizuelno-geometrijske karakteristike. Tako kot njihova imena nakazujejo, služijo kot povezave med sklepi ali pa druge povezave. Same povezave nimajo spremnjlive atribute, tako da so njihova stanja odvisna od stanja sklepov.

Tipala (angl. *sensor*) so URDF elementi, ki določajo lastnosti vizuelnih tipal, kot so različne kamere in laserji. V ta element določimo, koliko pogosto beremo podatke iz kamere, velikost in format podatkov, razdalji bližje in daljše ravnine sekanja, vodoravno polje pogleda ipd.

Stanje modela (angl. *model state*) je element, ki določa stanja sklepov

robota. Stanje sklepov se določi s položajem, ki je kot vrtenje v primeru rotacijskega sklepa, oz. dolžina premika v primeru translacisjkega; hitrost premikanja sklepa in njegov navor.

Na ta način je bil zgrajen URDF model Katane, prikazan na sliki 2.2, ki ga ROS ponuja vsem uporabnikom skupaj s številčnimi ostalimi roboti.



Slika 2.2: Unified Robot Description Format model Katane 450.[8]

Eden izmed prispevkov diplomske naloge je to, da smo uspeli gonilnike, ki smo jih dobili za model Katane, prilagoditi za različico ROSa *Electric*. Vsi gonilniki so bili spisani za starejšo različico, t.i. *Diamondback*, prehod iz starejše v novo različico pa ni trivijalen. Naslednje, kar bi bilo pametno narediti, je migracija iz Electrica v najnovejšo razlišico ROS-a, ki se imenuje *Fuerte*.

2.2 Simulator Gazebo

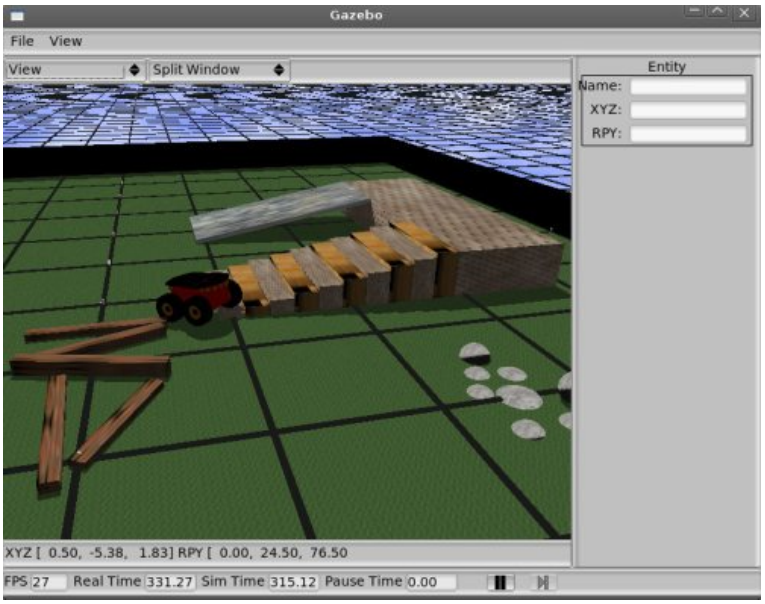
Simulatorji so zelo koristna orodja v razvoju programskih oprem za robote. Njihova vloga je predstaviti robote skupaj z njihovim okoljem v navideznem svetu in pri tem oponašati spremembe stanja v tem svetu, odvisno od delo-

vanja robota. Pri tem izračuna in prikazuje pravilne fizične interakcije med roboti in predmeti ter ponuja smoterne izhodne in vhodne podatke tipal. Glede na to, da prikazuje realistične simulacije robotov in okolj, robotski simulator omogoča razvijalcem robotske strojne opreme, da na varen način razvijajo svojo programsko opremo, tako da se izogibajo tveganja poškodbe strojne opreme [2]. Roboti in robotske roke so dragi, kot so tudi njihova popravila, zaradi tega se simulatorji, kot je Gazebo, izkažejo kot zelo uporabni. Roboti, ki jih simuliramo, se ne pokvarijo, če program poskuša izvesti neumno potezo–dejanski roboti pa se lahko. Simuliranim robotom nikoli ne bo zmanjkalo baterije–dejanskim pa se lahko. Simulirani roboti se ne bodo nikoli poškodovali, ali celotno uničili, če pride do trka z drugim predmetom, robotom, ali drugim delom istega–dejanski pa se lahko. To se nekateri izmed razlogov zakaj so simulatorji obvezen del izdelave robotskih programskih oprem.

Gazebo je odprtokodni simulator za robote v okolju, sposoben za simuliranje tridimenzionalnega sveta z več roboti, tipal in predmetov [2]. Ta simulator realno oponaša svetovne zakone fizike - razen če uporabnik tega noče - in računa interakcije. Poleg tega omogoča tudi vnos izhodnih podatkov iz različnih vrst priklopljenih na robote in jih uporablja za rešavanje danih nalog (Primer uporabe Gazeba je prikazan na Sliki 2.3). Na ta način nam gazebo dovoljuje pravilen prikaz delovanja robotov in s tem nam sporoča, ali je naša programska oprema pripravljena za prehod iz simuliranega sveta v pravi.

Značilne karakteristike simulatorja Gazebo so [2]:

- Dinamična simulacija, ki jo dosežemo z dostopom do različnih fizičnih pogonov in neposrednim nadzorom nad njihovimi parametri, kot so točnost in zmogljivost.
- Napredna tridimenzionalna grafika z uporabo *OGRE*, odprtokodni pogon za 3D grafiko, ki daje realistično izvedbo okolice.
- Simulacije tipala, ki ustvarjajo informacije tipal različnih vrst.



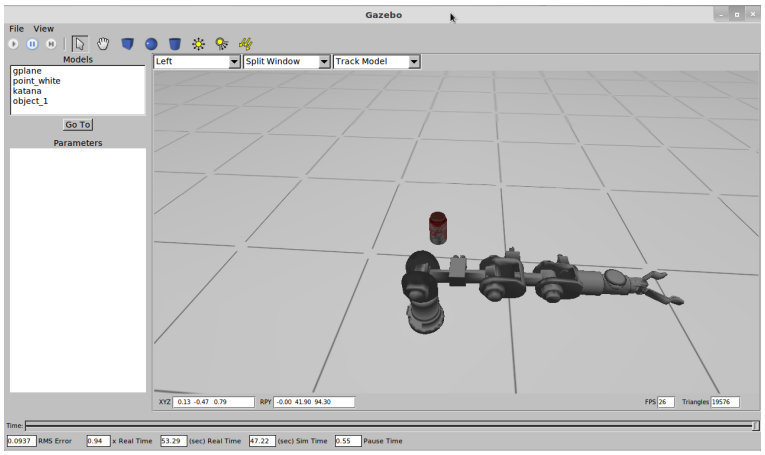
Slika 2.3: Primer uporabe simulatorja Gazebo.[10]

- Veliko število robotskih modelov je na voljo, med katere sodijo *PR2*, *TurtleBot* in *Katana*.
- Dostop do številnih predmetov s preprostimi in bolj zapletenimi oblikami, ki jih lahko dodamo v svet.
- Programabilnost vmesnika, ki podpira ogrodji ROS in *Player*. Obstaja tudi vmesnik API za spreminjanje gazeboevih vmesnikov po meri uporabnika.
- Vtičniki, ki omogočajo neposredno kontrolo vseh aspektov simulacije.
- Komunakacija preko protokola TCP/IP omogoča, da zaženemo Gazebo iz oddaljenih strežnikov.
- Močan grafični uporabniški vmesnik, ki dovoljuje uporabniku nadzor nad več paramterov simulacije in premikanje po svetu.
- Možnost uvoza modelov formata *Collada* s t. i. *Collada Reader*jem.

- Aktivna skupnost uporabnikov, ki obsega veliko raziskovalnih inštitutov, ki uporabljajo in prispevajo h Gazebu.

- Simulacija osebe, ki jo dobimo s predvajanjem podatkov iz kamere za snemanje gibanja.

  Za naše potrebe smo uporabljali ROS paket, ki omogoča uporabiku uporabo simulatorja Gazebo v ROS-u. Na ta način smo zgradili prazen navidezen svet in v njega vstavili URDF model Katane 300 ter en predmet za manipuliranje z robotom. Takšen preprost svet je prikazan na Sliki 2.4.



Slika 2.4: Robotska roka Katana 300 v praznem svetu.

2.3 Kinect

Kinect (prikazan na Sliki 2.5) je vhodno tipalo, ki spoznava gibanje. Razvilo ga je podjetje *PrimeSense* za igralno konzolo *Xbox 360*. Kinect vsebuje barvno kamero in tipalo za globino, ki skupaj vrneta sliko z ločljivostjo 640 x 480 točk formata *RGBD*. Format RGBD (angl. *Red Green Blue Depth*) za vsako točko slike vsebuje tako informacije o vrednosti rdeče, zelene in modre komponente barve, kot tudi o globini

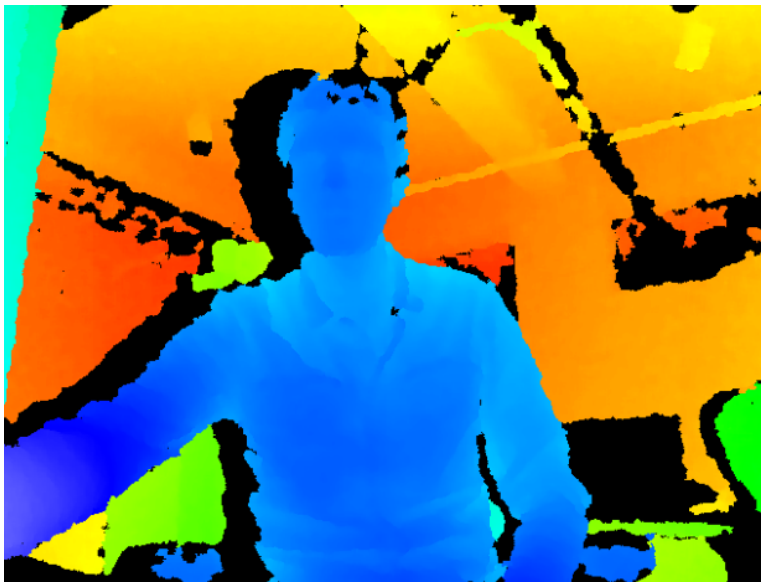
točke. Globina se iizračuna s pomočjo naprav na obeh straneh barvne kamere. Naprava, ki se nahaja levo od kamere, pošilja infrardečo svetlobo, ki producira vzorec globine. Tipalo na desni strani zajame deformacije tega vzorca, ki se je pojavilo, ko se infrardeča svetloba odbija od predmete in na ta način zgradi zemljevid globine. Primer globinske slike Kinecta je prikazan na Sliki 2.6. [3]

Slika 2.5: Prikaz globinske kamere *Kinect*. [7]

RGBD izhod, ki ga zgradi Kinect, je zelo uporaben, ker ni odvisen od barve in ker iz dobljenega zemljevida globine z uporabo praga z določeno razdaljo odstranimo šume in neželene predmete. Po filtriranju lahko iz globinskega zemljevida zaznavamo ovire, predmete in njihove lege ter predvidevamo trke. [3].

2.4 Robotska roka Katana

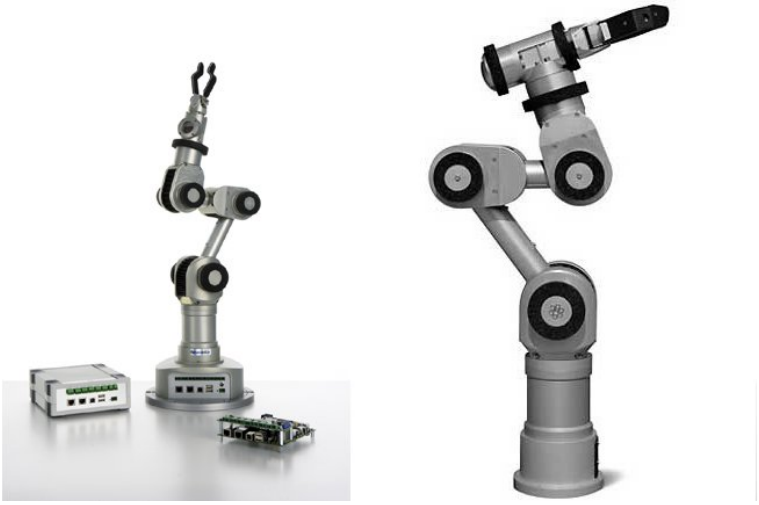
Robotsko roko Katana je razvilo švicarsko podjetje Neuronics AG in je fleksibilna robotska roka, ki služi za rokovanje predmetov in različna merjenja. Označena je kot inteligentna industrijska robotska roka, ki lahko varno deluje v bližini ljudi brez posebnih ograj. [4]



Slika 2.6: Primer globinske slike, pridobljenih iz tipala za globino Kinect.[11]

Naše delo je temeljilo na robotski roki Katana 300 6m180, ki se uporablja tudi v Laboratoriju za umetne vizualne spoznavne sisteme na Fakulteti za računalništvo in informatiko. Ta model Katane vsebuje pet *prostostnih stopenj* (angl. *Degree of freedom*), ki definirajo njegov obseg gibanja v obliki nepopolne sfere. Vsi sklepi na Katani so rotacijski in imajo svoje motorje za njihova vrtenja. Prvi sklep, ki se nahaja v osnovi roke, se vrti okrog navpične osi in ima obseg skoraj 360 stopinj. Drugi sklep se nahaja nad prvim in se, tako kot tudi naslednja dva, vrti okrog vodoravne osi. Peti sklep ima dve različni nastavitvi glede priključitev povezave med zadnjim sklepom in prijemalom. En način priključitve, ki smo ga mi uporabljali, je postaviti povezavo tako, da nadaljuje v isti smeri kot prejšna povezava. Na ta način je prijemalo kolinearno tretjemu in četrtemu sklepu. Drug način priključitve je tak, da podamo zadnjo povezavo pravokotno prejšni in tako je tudi prijemalo vedno pravokotno predzadnji povezavi. Razlike med tema dvema namestitvama se vidijo na Sliki 2.7. Ne glede na to, kako je robotska

roka nameščena, se zadnji sklep vrti na isti način. Na koncu zadnje povezave se nahaja prijemalo, ki ga sestavljata dva prsta. Prsti Katane imajo svoja zasebna motorja in tipala ter na ta način omogočajo prijemanje predmetov.



Slika 2.7: Razlike med namestitvami zadnje povezave robotske roke.[9]

Poglavje 3

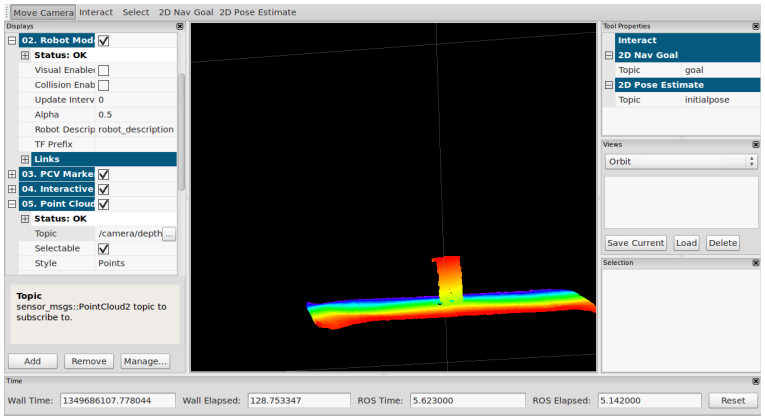
Zaznavanje predmetov

Zaznavanje predmetov in njihovih leg je zelo neizogiben del manipulacije z robostskimi rokami. Za razliko od človeka, ki mu to dejanje pride dokaj intuitivno in trivialno, v robotiki ta proces vsebuje nekoliko korakov, ki jih bomo opisali v nadaljevanju. V našem primeru, moramo najprej iz globinske kamere dobiti globinski zemljevid sveta. Ta globinski zemljevid lahko popravimo tako, da odstranimo šum podatkov. Potem odstranimo vodoravno ravnino, na kateri se nahajajo predmeti, ki jih hočemo dobiti. Ko ravnino porežemo, točke, ki ostanejo, razvrščamo v skupino, oz. gručimo in dobimo gruče točk, ki predstavljajo predmete v svetu. Na ta način dobimo kartezične koordinate predmetov v koordinatnem sistemu kamere, ki jih nato transformiramo v koordinatni sistem robotske roke.

3.1 Zemljevid globine

Tipalo, ki ga rabimo za zajemanje globinskih podatkov sveta (Kinect) nam ponuja globinski zemljevid v obliki *oblaka točk* (angl. *Point cloud*). Oblak točk je množica vozlišč v tridimenzionalnem koordinatnem sistemu, ki predstavljajo zunanje površine predmetov. Kinect ga

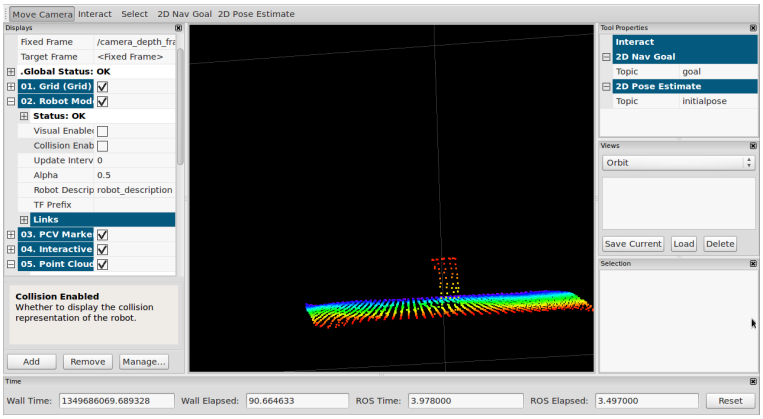
dobi z uporabo svojih dveh naprav za določanje globine, tako da, ena naprava pošilja infrardečo svetlobo, ki se deformira, ko se reflektira od različnih predmetov, kar spozna tipalo na drugi strani Kinectove barvne kamere. Na Sliki 3.1 je prikazan oblak točk, pridobljen iz Kinecta.



Slika 3.1: Primer oblaka točk, ki smo ga dobili s Kinectom.

3.2 Podvzorčanje oblakov točk

Podvzorčanje, oz. zmanjšanje števila točk v oblaku, nam omogoča odstranjevanje šumov iz podatkov. To naredimo z uporabo filtra *VoxelGrid* (*mreža volumetričnih elementov*), ki iz vhodnega oblaka točk naredi tridimenzionalno mrežo vokslov. 3D mreža vokslov predstavlja množico manjših škatelj v prostoru. Filter VoxelGrid za vsako tako škatljo vse točke aproksimira z njihovim centroidom, kar je počasneje kot s središčem škatlje, ampak ponuja bolj pravilne rezultate. Rezultate tega filtra shranimo v nov oblak točk, ki ga uporabljamo v naslednjih postopkih na mestu prvotnega. Rezultati filtra VoxelGrid so prikazani na sliki 3.2.



Slika 3.2: Oblak točk po uporabe filtra VoxelGrid.

3.3 Odkrivanje vodoravnih ravnin

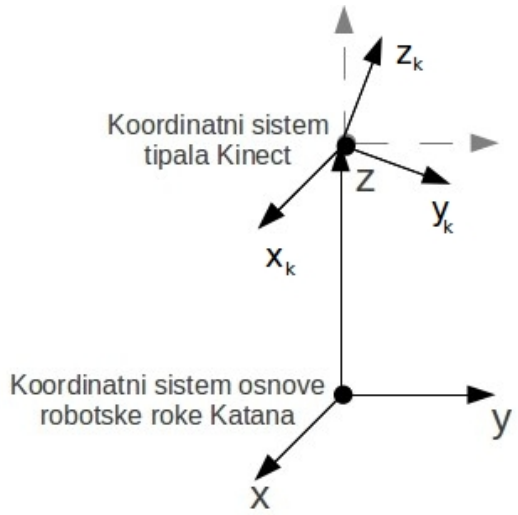
Naslednji korak postopka zaznavanja predmetov je odkrivanje vodoravne ravnine, na kateri se nahajajo predmeti. V našem primeru se predmet, na katerega hočemo kazati nahaja na pisalni mizi, tako, je ta miza ravno ta ravnina, ki jo iščemo. To naredimo z uporabo iterativne metode *Random Sample Consensus* (skrajšano *RANSAC*), ki ocenjuje parametre matematičnih modelov predmetov iz množice opazovanih podatkov. Ta algoritem sta objavila Fischler and Bolles leta 1981. RANSAC izločuje oblike tako, da naključno izbere minimalno množico iz oblaka točk in zgradi ustrezno primitivne oblike. Minimalna množica je najmanše število točk, ki enkratno opisujejo dano vrsto geometriskih primitvnih oblik. Oblike, ki jih dobimo, primerjamo z vsemi točkami, da bi ugotovili, koliko je izmed bližnih točk bilo pravilno aproksimirano z obliko. Po danem številu iteracij se izloči oblika, ki približujejo največjo število točk [5]. Na ta način lahko RANSAC zaznava pravilne matematične oblike kot so krogi in premice.

RANSAC se lahko prilagodi za odkrivanje ravnin. Na začetku algoritem naključno izbere tri točke iz oblaka in iizračuna parametre ravnin, na katerih se nahajajo. Nato zazna vse točke vhodnega oblaka, ki se na-

hajajo na teh ravninah, odvisno od praga tolerantnosti, ki smo ga dali kot vhoden parameter metode in označuje maksimalno razdaljo med ravnino in točkami, ki jih obdelujemo. Ta postopek algoritem ponavlja večkrat, odvisno od števila iteracij, ki smo ga izbrali. Ostala parametra, ki jih podamo algoritmu, sta največje število točk, ki lahko ležijo na ravnini, in minimalna verjetnost, da najde vsaj eno dobro množico opazovanja. [6]

3.4 Gručenje točk

Ko imamo ravnino, ki detektirano ustreza mizi, oblak točk porežemo v to ravnino in nam ostanejo točke, ki opisujejo predmet, ki leži na njej. Te točke gručimo in dobimo gručo (angl. cluster), ki predstavlja naš predmet. Potem iizračunamo centroid gruče in tako dobimo koordinate predmeta v koordinatnem sistemu Kinecta. Da bi lahko na ta predmet kazali s pomočjo robotske roke, moramo določiti lokacijo centroida predmeta v koordinatnem sistemu robota. To naredimo s transformacijo iz sistema tipala Kinect v koordinatni sistem roke. Najprej točko, ki predstavlja centroid predmeta, zavrtimo okrog osi x za isti kot, na katerem je postavljena kamera. Potem točko transliramo v smeri osi z za dolžino, ki ustreza razdalji med kamero in osnovo roke. Ilustracija odnosa med obema koordinatnema sistemoma je dana v Sliki 3.3. Zdaj, ko imamo centroid predmeta predstavljen v koordinatnem sistemu robota, lahko z uporabo inverzne kinematike uporabljamo robotsko roko da bi kazali na njega.



Slika 3.3: Razmerje med koordinatnema sistemoma robotske roke in kamere.

Poglavje 4

krmiljjenje robotske roke

4.1 Parametri robota Katana

Robotske roke krmiljimo s spreminjanjem stanja njihovih sklepov. Vendar se položaj roke ne more določiti brez poznavanja parametrov povezav med sklepi, ki so konstantni. Dolžine povezav na Katani, ki smo jih dobili iz URDF datoteke modela katane, so prikazane v Tabeli 4.1.

Za razliko od povezav, so parametri sklepov spremenljivi in njihovo spreminanje nam omogoča doseči cilje, ki smo jih dobili. Vse pet sklepov Katane je rotacijskih in s tem lahko prijemalo Katane doseže točke, ki se nahajajo okrog njene osnove. Vsak sklep ima svoje strojne ome-

Ime povezave	Dolžina povezave v metrih
Višina osnove	0.2015
Druga povezava	0.1900
Tretja povezava	0.1390
Četrta povezava	0.1523
Peta povezava	0.1505

Tabela 4.1: Tabela dolžin sklepov na Katani.

Ime sklepa	Spodnja meja vrtljivosti	Zgornja meja vrtljivosti
Prvi sklep	-0.3025528	2.891097
Drugi sklep	-0.135228	2.168572
Tretji sklep	-2.221804	2.054223
Četrti sklep	-2.033309	1.876133
Petti sklep	-2.993240	2.870985
Desni prst	-0.44	0.30
Levi prst	-0.44	0.30

Tabela 4.2: Meje vrtljivosti sklepov podane v radijanih, kot so opisani v URDF model Katane.

jitve, ki so dane v Tabeli 4.1. Te meje upošteva tudi naš simulator Gazebo, oz. URDF model, ki opisuje Katano.

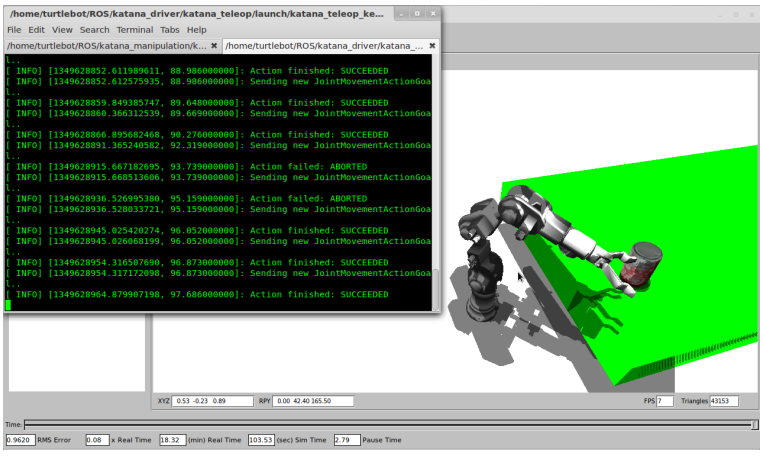
Vsak izmed sklepov vsebuje motor, ki poskrbi za njegovo vrtenje. Kako pomembni so ti motorji kaže to, da laboratorijski robot trenutno ni uporaben samo zaradi tega, da je eden izmed motorjev v okvari. Zaradi tega smo morali program celotno razvijati v simulatorju. Ampak Gazebo dokaj pravilno simulira svet in delovanje robota, kar pomeni, da se naša programska oprema lahko preseli iz navideznega sveta v realni, ko bo roka spet delujoča.

Z vsakim premikanjem enega izmed sklepov se lega prijemala spreminja, spreminjajo se tudi lokacije vseh sklepov, ki pridejo za njim v verigi. To pomeni, da je pozicija in orientacija prvega sklepa odvisna samo od njegovega stanja, zadnjega pa od vseh, ki se nahajajo pred njim v kinematski verigi robota. Kinematsko verigo uporabljamo sicer na dva načina: z direktna in inverzno kinematiko.

4.2 Direktna kinematika

Neposredna ali *direktna kinematika* (angl. *forward kinematics*) kinematske enačbe za računanje pozicije prejamala na robotski roki, z uporabo parametrov sklepov. Direktna kinematika natančno izračuna lego konice roke in ponuja eno samo rešitev. Rešitve za direktno kinematiko v ROS-u dobimo na zelo enostaven način - z uporabo paketa*tf* (skrajšano oblika besede *transformation*, transformacija). Naloga tega paketa je računati transformacije med koordinatnimi sistemi, in med ostalimi lahko ga povprašamo za transformacijo med koordinatnim sistemom osnove katane in prijemala.

Robotsko roko prek ROS-a lahko krmiljimo ročno, oziroma teleoperacijsko z uporabo tipkovnice. Na ta način lahko z roko pridemo kamorkoli v njegovem delovnem obsegu (angl. working envelope). Primer ročnega krmiljjenja Katane v Gazebu je prikazan na sliki 4.1.



Slika 4.1: Prikaz ročnega krmiljenja Katane v simulatorju.

4.3 Inverzna kinematika

Pojem inverzna kinematika je nasproten pojmu direktna kinematika in se nanaša na uporabo kinematičnih enačb za določanje parametrov sklepov tako, da bi bila konica roka na želeni legi. Za razliko od direktne kinematike, lahko inverzna kinematika najde veliko število rešitev ali pa tudi nobene. Problem inverzne kinematike je zaradi tega veliko bolj težek in zapleten kot pri direktni, ki je trivijalen.

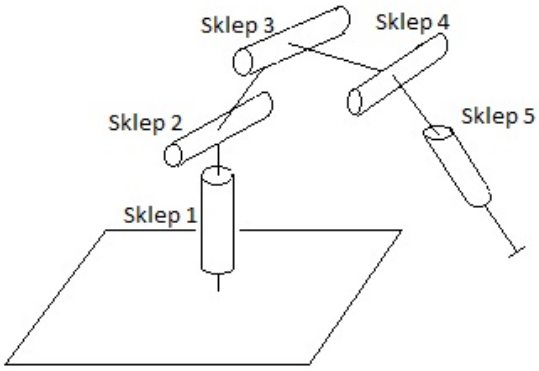
ROS za potrebe svojega robota PR2 uporablja knjižnico *KDL*(*Kinematics and Dynamics Library*), ki se lahko uporablja tudi za Katano. Vendar ta knjižnica nam ni pomagala najti rešitve za inverzno kinematiko zaradi razlogov, ki jih nismo mogli odkriti.

4.3.1 Algoritem za inverzno kinematiko

Zaradi težav s prilagajanjem knjižnice KDL za robotsko roko Katana smo se odločili napisati svoj poenostavljen algoritem za inverzno kinematiko, ki temelji na geometriji in njeno veji, trigonometriji. Vrste in postavitev sklepov so prikazani na Sliki 4.2. Rešitev smo začeli graditi od *Sklepa 1* naprej. Orientacijo tega sklepa smo določili z iskanjem kota med vozliščem, ki je cilj inverzne kinematike, in osnovo Katane ne glede na njihove koordinate višine. Kot dobimo z uporabo Enačbe 1., in od tu naprej iščemo rešitev v dvodimenzionalnem koordinatnem sistemu, namesto v 3D, zato, ker se vsi naslednji sklepi vrtijo okoli vodoravni osi. Koordinata z, ki predstavlja višine, se ne spreminja, drugo koordinato pa dobimo iz Pitagorevega izreka, dan v Enačbi 2.

$$tan\alpha = y/xEna\check{c}ba1..$$

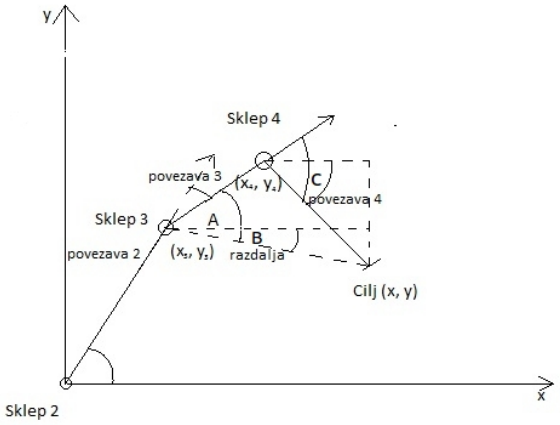
$$c^2 = \sqrt{x^2 + y^2}.Ena\check{c}ba2.$$



Slika 4.2: Shema postavitve sklepov na Katani.

Ko problem preuredimo na 2D, nam ostanejo še trije sklepi in njihove pozvezave, ki smo jih vnesli kot konstante. Naslednji korak algoritma je določanje kota vrtenja *Sklepa 2*. To naredimo tako, da začnemo od njegove zgornje meje in postopno zmanjšujemo kot, dokler ne pridemo do spodnje meje ali dokler ne najdemo rešitve. Za vsak kot izračunamo, kje bi se nahajal *Sklep 3*, če tako zavrtimo *Sklepa 2*, in v primeru da je razdalja med tretjim sklepom in želeno točko (*razdalja*) manjša, kot je seštevek Povezave 3 in 4 (ali je mogoče priti do želene točke), algoritem nadaljuje z iskanjem kota naslednjega sklepa. V Sliki 4.3 je prikažana shema za lažje razumevanje algoritma in njegove spreminljivke.

V slučaju, da je pogoj, dan v Enačbi 3., izpolnjen, potem lahko v točko pridemo v dveh potezah, če meje sklepov to dovoljijo. Geometrijsko si ta problem lahko predstavljamo kot iskanje kotov trikotnika z vsemi stranmi znane (stranica-stranica-stranica). Koti se lahko izračunajo z uporabo kosi-



Slika 4.3: Prikaz spremenjlivke uporabljenih v algoritmu za inverzno kinematiko.

nusne funkcije, ki je podana v Enačbi 4.

$$razdalja \leq \sqrt{Povezava_3^2 + Povezava_4^2} \ \vee \ razdalja \geq \sqrt{Povezava_4^2 - Povezava_3^2}. Enačba 3.$$

$$\cos \ A = \frac{razdalja^2 + Povezava_3^2 - povezava_4^2}{2 * razdalja * Poveyava_3}. Enačba 4.$$

Ko se izračuna kot v Sklepu 3 med Sklepom 4 in ciljno točko (kot A), algoritem z uporabo Enačbe 5. in 6. izračuna kot vrtenja Sklepa 3. Program potem preveri, ali je kot v mejah tretjega sklepa, in če je, nadaljujemo z zadnjim kotom.

$$\tan B = \frac{y - y_3}{x - x_3}. Enačba 5.$$

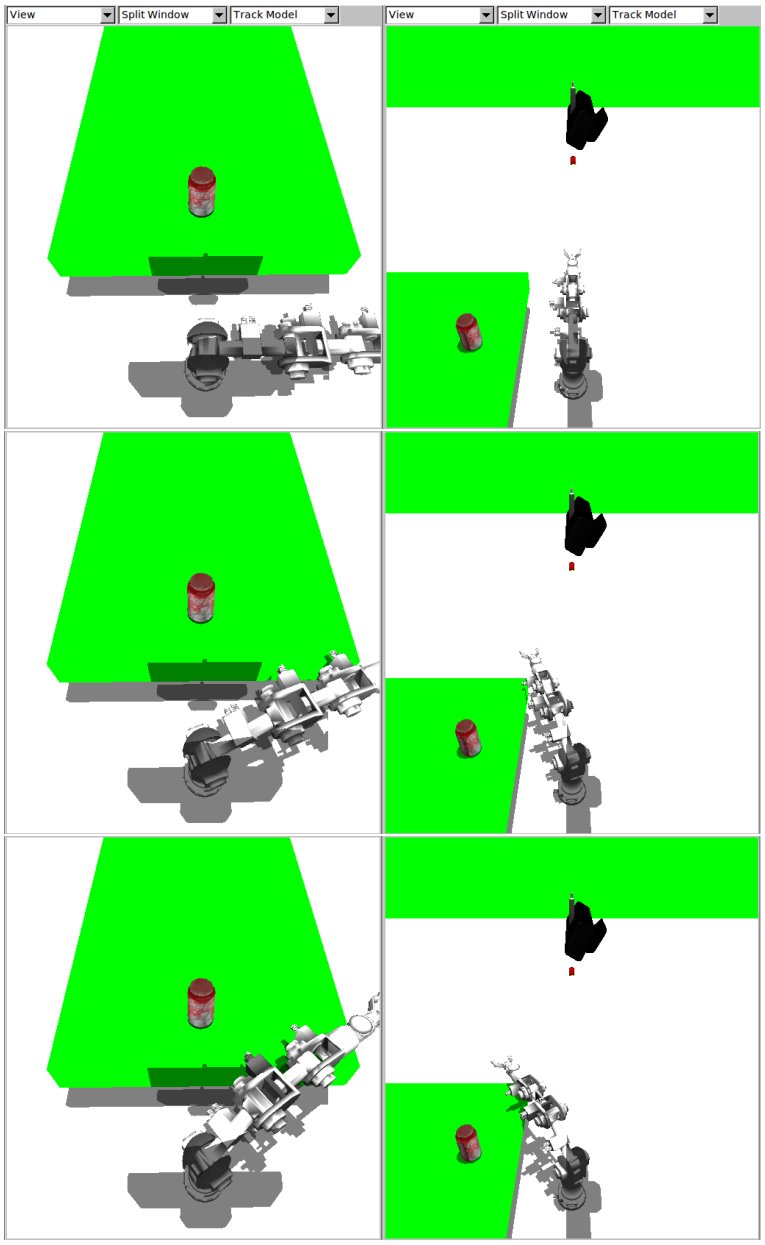
$$Sklep_3 = -1 * (B + Sklep_2 - A). Enačba 6.$$

Na koncu ostane samo še Sklep 4. Njegov kot vrtenja sistem izračuna, tako da poišče kot med želeno točko (ciljem) in Sklepom 4 s pomočjo pravokotnega trikotnika, ki ga zgradijo (kot C) z uporabo Enačbe 7. Ko sistem ta kot iizračuna, iizračunati bi moral samo še Enačbo 8., da bi dobil kot za katerega bi se Sklep 4 zavrtel, da bi roka prišla v željeno pozicijo. Ta kot bi moral biti v meji Sklepa 4. V primeru, da je, algoritem vrne nazaj stanje sklepov, v katerih naj bi bila roka postavljena, da bi kazala na to točko. Če kot, ki ga je sistem izračunal, ni v mejah sklepa, potem iteracija nadaljuje z iskanjem rešitve.

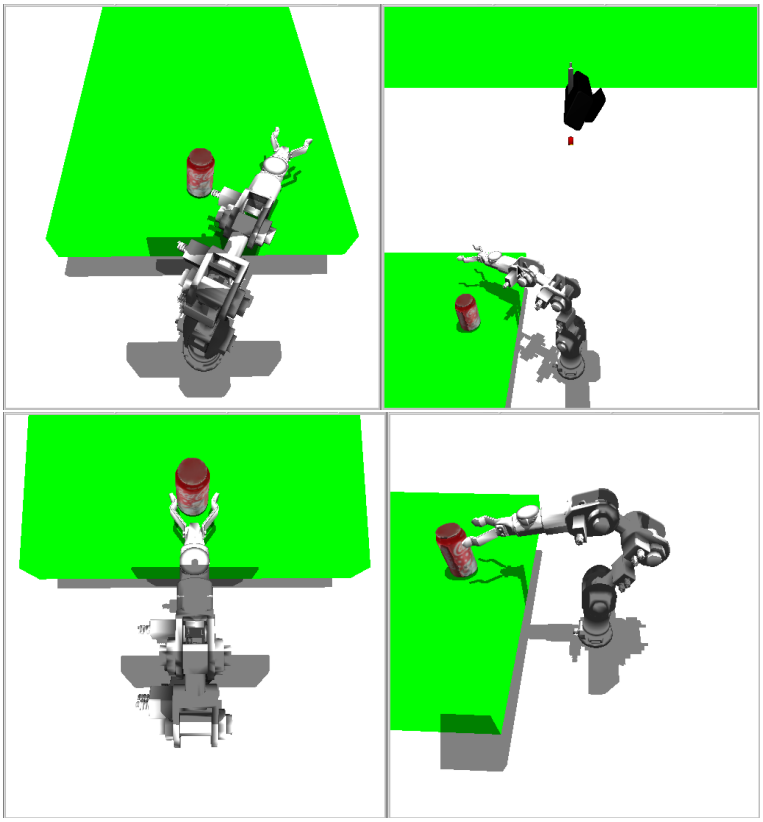
$$\tan\ \alpha = \frac{y - y_4}{x - x_4}. Enačba 7.$$

$$Sklep_4 = Sklep_3 + Sklep 2 + C. Enačba 8.$$

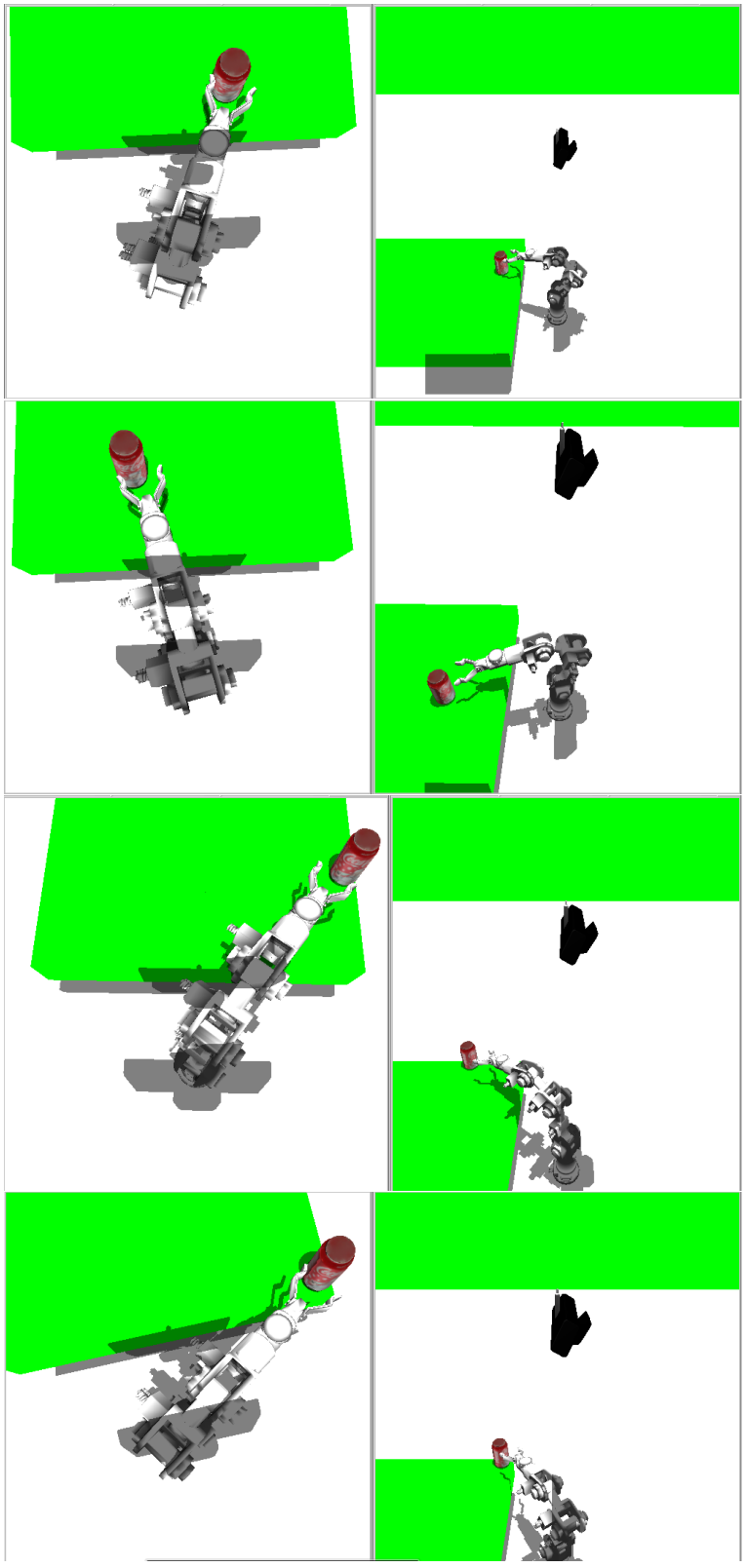
Ko imamo vse kote izračunane, z ROS-ovim pakatom za krmiljjenje roke *arm navigation*, kličemo metodo za premikanje roke. Ta metoda sklicuje knjižnico *Open Motion Planning Library* in s tem načrtuje kako se roka, oziroma njeni sklepi gibajo, da bi prišli v stanje, ki smo ga podali. OMPL je lahka in prilagodljiva knjižnica za načrtovanje gibanja, ki se zasnova na vzorčenje. Knjižnica je odprtokodna in vsebuje kode programskega jezika C++ s ciljem implementirati številčne napredne algoritme za načrtovanje gibanja. OMPL na našo zahtevo precizno iizračuna, kako naj se vsak izmed sklepov premika v danem času, da bi prišel do končne pozicije. Potem se kliče akcija za gibanje roke glede načrta ter se naša roka premakne in pride do želene točke, oziroma kaže na predmet. Primeri delovanja programa so prikazani na Sliki 4.4, Sliki 4.5 in Sliki 4.6. Slika 4.4 in Slika 4.5 prikažujeta gibanje robotske roke proti cilju. Slika 4.6 prikazuje kako roka najde predmet, ko je ta postavljen na različne položaje.



Slika 4.4: Prikaz delovanja programa v Gazebu, gledano z dva različnih pogledov.



Slika 4.5: Prikaz delovanja programa v Gazebo, gledano iz dva različna kota(nadaljevanje).



Slika 4.6: Prikaz delovanja programa ob spreminjanju položaja predmeta.

Poglavje 5

Sklepne ugotovitve

Za diplomsko nalogo smo naredili program, ki krmilji robotsko roko *Katana* in jo uporablja, da bi kazala na predmet na mizi. Robotski sistem zaznava predmet s pomočjo globinskega tipala *Kinect* in algoritme za gručenje globinske mape. Ko spoznamo predmet in dobimo točko, na katero hočemo z robotom kazati, poskušamo najti način, kako to narediti. Zaradi tega smo napisali svoj algoritam, ki računa *inverzno kinematiko* za model našega robota. Naslednji korak je načrtovanje gibanja, ki nam omogoča knjižnica *OMPL* in potem v simulatorju *Gazebo* prikažemo spremembo stanja robota oz. njegovo gibanje.

V času izdelave diplomske naloge nismo imeli možnosti uporabljati našo programsko opremo za krmiljjenje dejanske roke, ki se nahaja na Fakulteti. To je posledica dejstva, da je ta robotska roka trenutno v okvari, in zaradi tega smo za naše potrebe uporabljali simulator Gazebo. Vendar prehod iz simulirane roke v pravo ne bi smel predstavljati veliko težav, zaradi tega, ker smo dokaj realistišno simularili model roke, njegove parametre in njegovo delovanje.

Kazanje je zelo korisno v neverbalnih komunikacijah, kjer lahko robot pokaže sogovorniku, kar je iizračunal oz. na kaj misli. Tudi v verbalnih komunikacijah se kazanje iskaže za koristno, ker lahko pomaga robotu nekaj pojasniti, ali pa prikaže o katerem predmetu govori.

Kazanje na predmete je postopek, ki je podoben prvi fazi postopka prijemanja predmetov. Robotska roka Katana, kot tudi veliko število ostalih robotov, vsebuje dva prsta, ki služijo za prijemanje predmetov. Nas algoritem bi lahko tako nadgradili s funkcijo prijemanja predmetov. Rokovanje predmetov z robotsko roko je mogoče uporabljati na veliko koristnih načinov, še posebej ko imamo barvno globinsko kamero, ki lahko robotu omogoča samostojno delo.

Literatura

[1] M. Quigley, B. Gerkey, K., J. Faust, T. Foote, J. Leibs, E. Berger , R. Wheeler , A. Ng. "ROS: an open-source Robot Operating System", 2008.

[2] Domača spletna stran simulatorja Gazebo. Dostopno na: http://http://www.gazebosim.org/.

[3] M. Tölgyessy, P. Hubinský. "The Kinect Sensor in Robotics Education", 2010.

[4] Spletna stran linuxfordevices.com, "Robotic arm runs Linux", 2008. Dostopno na: http://www.linuxfordevices.com/c/a/News/Robotic-arm-runs-Linux/.

[5] R. Schnabel, R. Wahl, R. Klein. "Efficient RANSAC for Point-Cloud Shape Detection", 2007.

[6] M. Y. Yang, W. Förstner. "Plane Detection in Point Cloud Data", 2010

[7] Spletna strana Wareframe, dostopna na: http://wireframe.co.za. 2012.

[8] Domača spletna stran Robotic Operating Systems, dostopna na: http://www.ros.org. 2012.

[9] Spletna stran Fox News, dostopna na: http://www.foxnews.com. 2012.

[10] Spletna stran Oddelka za elektroniko, Univerza v Yorku, dostopno na: // http://http://www.elec.york.ac.uk. 2012

[11] Spletna stran Thre Byte Intermedia, dostopna na: http://3-byte.com. 2012.