SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 3962

Teleoperacija robotske ruke Kinova Jaco 3D kamerom

Filip Marić

Zagreb, travanj 2016.

SADRŽAJ

1. Uvod

Klasična upravljačka sučelja za robotske manipulatore kao što su tipkovnice ili daljinski upravljači u uporabi su od samih početaka razvoja robotike. Kao pozitivna zajednička karakteristika ovih sučelja mogla bi se navesti njihova jednostavnost izvedbe: stisak odgovarajuće kombinacije tipki ili guranje kontrolne palice uzrokuje neku radnju manipulatora. Dok su se ovakva sučelja pokazala efektivnim i cijenovno povoljnim rješenjem za ustaljene uloge robotskih manipulatora u industriji, njihov učinak u novijim primjenama ostavlja prostora za drugačije pristupe.

Zahvaljujući masovnoj proizvodnji tehnologija koje su za početka razvoja robotike bile u povojima, danas je prostor za razvoj inovativnih pristupa upravljačkim sučeljima veći nego ikad. Robotski manipulatori vrlo su često anatomski slični ljudskoj ruci, što nas je dovelo do zaključka kako bi upravljanje manipulatora ljudskom rukom rezultiralo visokom razinom intuitivnosti, čiji manjak smatramo glavnim nedostatkom klasičnih sučelja.

Zamišljeno upravljačko sučelje razlikuje se od klasičnih po još jednoj osnovnoj karakteristici: ne zahtjeva kontakt između opreme i korisnika. Ovakav sustav ostvarujemo snimanjem pokreta ruke korisnika pomoću 3d kamere, koja uz sve funkcije standardne kamere vraća informaciju o udaljenosti objekata u kadru. Dobivene informacije obrađuju se razvijenim algoritmom detekcije dlana ruke te se šalju preko mreže na upravljačko računalo robotskog manipulatora, koje ih pretvara u naredbe.

Ovakvim upravljanjem dobivamo prirodnije kretanje robotskih manipulatora, što će postupnim ulaskom robota u svakodnevni život postati tražena karakteristika. U industriji zabave jedna moguća primjena je pomicanje kamere na kraju više-osnog manipulatora, gdje ovakvo upravljanje omogućuje intuitivniju kontrolu kadra od strane snimatelja. NASA i ostale svemirske agencije već su jedno vrijeme fokusirane na ovakav pristup problemu upravljanja robota u orbitalnim postajama, gdje klasična sučelja zakazuju zbog pobjega tereta. Ljudima s poteškoćama u kretanju robotska ruka upravljana ovakvom metodom može olakšati dohvaćanje objekata u okolini, pritom dajući obavljanju zadataka ljudsku dimenziju koja je pri uporabi ovakvih pomagala bitna.

U poglavlju "Opći i specifični ciljevi rada" definiramo zadatke, zahtjeve i okvirnu strukturu upravljačkog sučelja koje razvijamo. Poglavlja "Sklopovska podrška" i "Programska podrška" opisuju sklopovlje i programske alate sa više ili manje detalja, ovisno o važnosti i frekvenciji pojavljivanja pojedine komponente u sustavu. Posebno se zadržavamo na opisu ROS operacijskog sustava i pratećih paketa, koji čine centralnu komponentu sustava i povezuju sve ostalo.

Teoretsku pozadinu problema kretanja manipulatora i prijedlog rješenja obrađujemo u poglavlju "Kinematika manipulatora". Predloženo rješenje u ovom poglavlju također programski provjeravamo na razini matematičke ispravnosti. Problem detekcije stanja ljudske ruke obrađen je u poglavlju "Detekcija", gdje izlažemo dvije metode i uspoređujemo njihove performanse.

Nakon što smo pojedinačno objasnili komponente koje sačinjavaju sustav, njihovu povezanost i detalje programske izvedbe izlažemo u poglavlju "Sustav upravljanja". U istom poglavlju izlažemo rezultate testiranja sustava na simulaciji robotske ruke, kako bi dobili uvid u moguće nedostatke sustava. Konačno, izlažemo rezultate dobivene korištenjem stvarnog manipulatora te zaključujemo rad.

2. Opći i specifični ciljevi rada

Opći cilj našeg rada bio je razvoj upravljačkog sučelja koje kao ulaznu informaciju koristi isključivo gestikulacije i pomake ruke korisnika. Zamišljeno sučelje mora biti jednostavno za savladati novom korisniku, intuitivno , pružiti odgovarajuću razinu preciznosti i pri tome zadržati relativno nisku cijenu. Iz ovih smjernica proizlaze specifični ciljevi rada:

- Cijeli sustav mora biti izveden koristeći besplatne, open-source[1] tehnologije. Ovime osiguravamo ažurnost sustava kroz vrijeme te jednostavnije popravke i modifikacije.
- Radi nedostatka open-source rješenja za detekciju ruke korištenom 3d kamerom, potrebno je razviti detekcijski algoritam koji uzima u obzir ograničenja korištene sklopovske podrške i potrebe sustava.
- Udaljeno upravljanje zahtjeva korištenje robotskog operacijskog sustava ROS, u kojem izvodimo cijelokupni upravljački algoritam.
- Sustav mora biti primjenjiv na bilo kojem manipulatoru sa 3 ili više osi. Ovo ukazuje na potrebu za programskim rješavanjem problema kinematike robota uz dostupnu opisnu datoteku i kompatibilnost sa najčešćim izvedbama ROS-API[2] sučelja.
- Sustav mora imati zadovoljavajuće dinamičko ponašanje, što zahtjeva testiranje kinematike i upravljačke petlje na simulaciji razvijenoj za potrebe rada.

Kako bi ostvarili ove ciljeve bilo je potrebno savladati metodologiju korištenja nekolicine programskih paketa i biblioteka, kao i riješiti problem međusobne kompatibilnosti brojnih komponenti sustava. Komponente je potom potrebno uklopiti u strukturu

[1] **open-source** - čiji je izvorni kod i/ili nacrti (dizajn) dostupan javnosti na uvid, korištenje, izmjene i daljnje raspačavanje

[2] **API** (*engl. application programming interface*) - set programskih "blokova" razvijen sa svrhom olakšavanja komunikacije i upravljanja programskim ili sklopovskim sustavom

upravljačke petlje više razine, pri tome vodeći računa o radnim frekvencijama korištenih sklopova.

2.1. Interpretacija pokreta

Samom prenošenju kretnji ljudske ruke na robota pristupili smo na dva načina: prvi ostvaruje praćenje ljudske ruke u stvarnom vremenu, dok se drugi bazira na načelu sličnom upravljanju kontrolnom palicom (*engl. joystick*).

Praćenje ljudske ruke ostvarujemo "poistovjećivanjem" dlana korisnika s alatom na vrhu manipulatora. Pri početku upravljanja ljudska se ruka nalazi na sredini kadra i odgovarajućoj udaljenosti, ova početna pozicija poistovjećuje se sa trenutnom pozicijom alata manipulatora. Pomicanjem dlana korisnik pomiče željenu poziciju alata manipulatora. Upravljačka petlja "primjećuje" ovu razliku te regulacijom brzine zglobova ili izvršnog člana nastoji smanjiti pogrešku između trenutne i željene pozicije manipulatora.

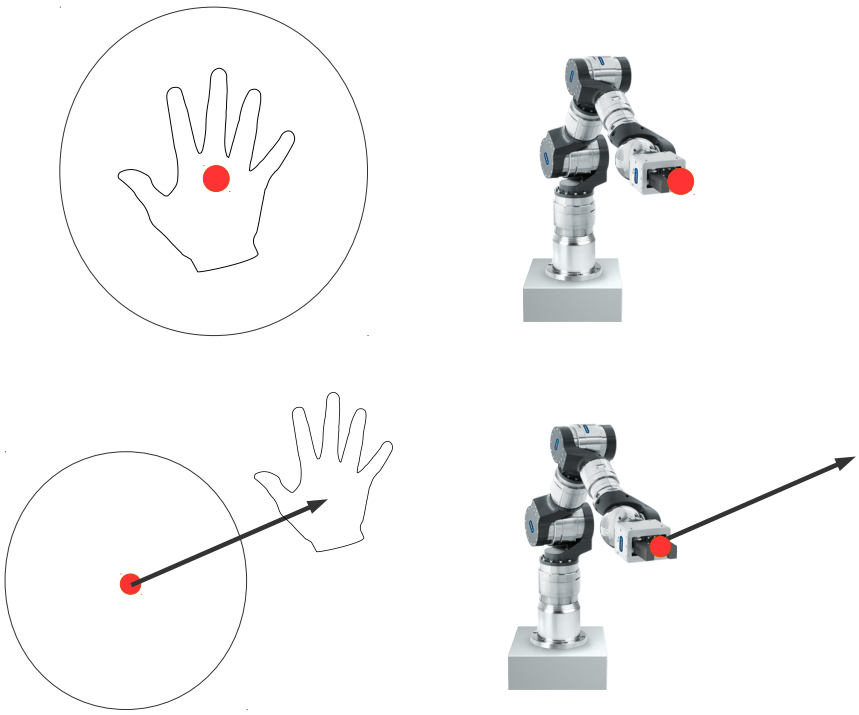
Izvedba *joystick* načina rada oko početne pozicije korisnikove ruke formira sferu promjera ovisnog o veličini dlana. Zadržavanjem dlana korisnika unutar kugle, upravljački algoritam zadržava izvršni član u trenutnoj poziciji. Pomicanjem dlana korisnika izvan sfere, upravljački algoritam pomiče vrh manipulatora u smjeru vektora razlike pozicija dlana od središta kugle.

Zajednička funkcija kod oba pristupa vezana je uz otvaranje i zatvaranje alata (izvršnog člana). Otvaranje šake korisnika upravljački algoritam interpretira kao naredbu otvaranja alata manipulaotra, dok se zatvorena šaka interpretira kao naredba zatvaranja. Ova funkcija naravno ovisi o tipu alata na kraju manipulatora, no mapiranje otvaranja i zatvaranja šake na neku drugu funkciju ne predstavlja problem radi separabilnosti dijela upravljačke petlje vezane uz izvršni član.

2.2. Struktura sustava upravljanja

Sustav upravljanja na najvišoj se razini sastoji od 4 komponente:

- **Detektor** Na dubinskoj RGBD slici pronalazi dlan korisnika i prostorne koordinate (i orijentaciju) njegove sredine.
- **Regulator** Interpretira razliku trenutne pozicije (i orijentacije) dlana i izvršnog člana manipulatora. Rezultat se pretvara u potrban zakret (ili brzinu zakreta) zglobova manipulatora, nakon čega se šalje u API sučelje.



Slika 2.1: Prikaz *joystick* načina upravljanja.

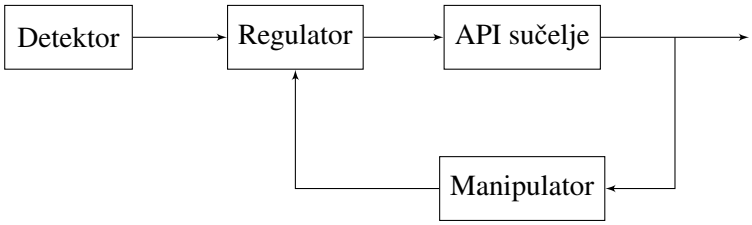
– **API sučelje** Prima upravljačku naredbu i nakon obrade je prosljeđuje na nižu razinu gdje se pretvara u API naredbu. U istom ciklusu API sučelje čita trenutna stanja zglobova manipulatora i šalje ih natrag prema regulatoru.

– **Manipulator** Prima API naredbu te je pretvara u konkretna kretanja zglobova. Ugrađeno sklopovlje bilježi senzorska očitanja sa zglobova te ih priprema za čitanje od strane API sučelja.

Kako bi kvalitetno ostvarili koncepte upravljanja navedene u potpoglavlju 2.1, vrlo nam je bitna povratna veza prikazana na slici 2.2. Povratna veza omogućuje nam uvid u trenutno stanje sustava čime se omogućuje ispravljanje pogrešaka, praćenje referentne veličine i otpornost na smetnje. Pomoću informacija o poziciji pojedinačnih zglobova također osvježavamo matricu Jakobijana manipulatora, koja je ključni element općeg kinematičkog rješenja praćenja u poglavlju 5.1.

Detektor

Zahtjevi postavljeni na detektor relativno su niski. Radna frekvencija trebala bi biti dovoljno visoka da pri prosječnom gibanju referentne veličine bivaju zadane prije no što ih manipulator može sustići kako bi izbjegli oscilacije. Minimalni zahtjev na sam detekcijski algoritam jest mogućnost pronalaženja koordinata ljudskog dlana u barem



Slika 2.2: Blok dijagram izvedbe sustava

dvije dimenzije, ovime se omogućava kretanje manipulatora po ravnini. U našem radu izvedeno je trodimenzionalno translacijsko kretanje, dok je upravljačka petlja potpuno osposobljena za detekciju punih 6 stupnjeva slobode gibanja.

Regulator

Pod pojmom regulator ovdje se podrazumjeva cijela programska struktura koja prima, obrađuje i šalje podatke između više objekata u stvarnom vremenu. Stvarnu strukturu regulatora detaljnije ćemo razmotriti u poglavlju 7, kada se upoznamo sa korištenom programskom podrškom i kinematičkim rješenjem.

Ovdje ćemo samo napomenuti da je za kvalitetan rad regulatora bitna sinkronizacija rada svih njegovih komponenti kako bi se ostvario stabilan protok podataka i stalna radna frekvencija sustava. Ovo je ostvarujemo koristeći mrežnu arhitekturu operacijskog sustava ROS, koji omogućuje određivanje radne frekvencije komponenti i daje detaljan uvid u protok podataka. Također, mrežna arhitektura ROS-a omogućuje pokretanje komponenti pojedinačno na različitim fizičkim računalima, što ide u prilog modularnosti sustava.

API sučelje

Općenitost sustava zahtjeva razdvajanje općih funkcija za regulaciju i obradu podataka od funkcija povezanih uz specifični korišteni manipulator. Konkretnije, ovu komponentu moramo ostvariti tako da se na nju može spojiti API bilo kojeg manipulatora koji zadovoljava fizičke zahtjeve.

Paket `ros_control`ostvaruje zadano sučelje koristeći apstraktne klase na koje "spajamo" API manipulatora. Dovoljno je funkcije čitanja stanja zglobova i zadavanja brzine/pozicije "umotati" u dostupne klase, te ostatak sustava izvesti koristeći njih.

Manipulator

Manipulator je robotski uređaj koji izvršava željenu radnju te je na njega postavljeno nekoliko osnovnih zahtjeva kako bi bio kompatibilan sa sustavom. Kao što je ranije spomenuto, sam manipulator mora imati minimalno dva računalom upravljiva stupnja slobode (time ostvarujemo planarno kretanje). Također, manipulator mora biti opermljen senzorima pozicije i/ili brzine koje je moguće čitati na računalu koristeći API.

3. Sklopovska podrška

Sustav je dizajniran da bude univerzalan s obzirom na manipulator i detekcijski uređaj, ali pri svim testiranjima koristili smo Kinova Jaco robotsku ruku i Microsoft Kinect 3d kameru.

3.1. Microsoft Kinect 3d kamera

[//TODO slika Kinecta] Moderni senzori su u mogućnosti prikazati sliku uz submilimetarsku preciznost. Takvu preciznost nam ne nudi Kinect, ali nudi sasvim prihvatljivu preciznost od nekoliko milimetara (ovisno o udaljenosti objekta).

Kinect je naziv za liniju Microsoftovih senzora pokreta razvijenih primarno za potrebe igraćih konzola, ali u širokoj uporabi u području računalnog vida. Sastoji se od dubinske kamere, klasične kamere te niza mikrofona pomoću kojih je u mogućnosti izvršavati kompleksne zadatke prepoznavanja i praćenja kao što je praćenje ljudskih pokreta.

U ovom radu je korišten *Kinect for Xbox 360* [**?**] te se ne razmatraju parametri drugih Kinect senzora.

Dubinska kamera se sastoji od emitera infracrvene svjetlosti koji projicira pseudoslučajne uzorke točaka u Kinectov vidokrug. Infracrvena kamera koja se nalazi na znanoj udaljenosti snimi položaj točaka, po kodiranom uzorku zna pod kojim kutom je uzorak generiran te jednostavnom triangulacijom izračuna dubinu u danoj točki prostora. Kako postoji određena udaljenost između senzora i emitera, na nekim mjestima Kinect neće biti sposoban odrediti dubinu zbog okluzije (Slika 3.1).

Okluzija se pojavljuje kao posljedica razmaka između emitera infracrvene svjetlosti i kamere. Kako kamera promatra prostor s neke udaljenosti od emitera, vidi neke dijelove prostora koje emiter nije mogao obasjati jer su bili pokriveni drugim objektima. Tako na slici 3.1 bijelom bojom vidimo sjenu koju dubinska slika ruke ostavlja za sobom. Iz položaja sjene i ruke jasno se vidi kako emiter emitira infracrvenu svjetlost s desne strane kamere što i je slučaj ako pogledamo arhitekturu senzora gdje se

Slika 3.1: Okluzija u dubinskoj slici

emiter nalazi s desne strane.

Okluzije predstavljaju znatni problem u detektiranjima. U detekciji dlana, dlan može biti u takvom položaju da jedan dio dlana zasjeni drugi te ćemo dobiti maksimalnu moguću dubinu na tom području. Takvi slučajevi mogu prevariti detektor te zato valja imati na umu učinak okluzije prilikom konstrukcije detektora.

Jedan kvalitetan pristup detekciji dlana koristi sintetizirane slike dlana na kojima simulira i okluziju kako bi detektor bio što bolje naučen na prave slike [**?**]. U istom radu Xu i Cheng su simulirali i neuspješne detekcije malih površina. Jedan takav primjer vidimo na slici 3.2, gdje je kažiprst okrenut u smjeru kamere te ne uspijeva biti detektiran. Razlog takvih neuspješnih detekcija nalazimo u načinu na koji Kinect određuje kut pod kojim je površina obasjana. Kako je već spomenuto, Kinect generira pseudoslučajne uzorke ovisno o kutu pod kojim su oni projicirani te tako kamera uspoređivanjem uzoraka može znati pod kojim kutom je predmet obasjan. Ako je predmet dovoljno male površine, uzorak neće biti potpun. Kinect pri tome pristupa na isti način kao i kod okluzije, dubinu na tom području postavlja na maksimalnu moguću dubinu.

Slika 3.2: Gubitak malih objekata

3.2. Kinova Jaco robotska ruka

Kinova Jaco (3.3) robotska je ruka namijenjena osobama s poteškoćama u kretanju. Jaco 2010. godine na tržište je stavlja kanadska tvrka Kinova robotics s ciljem olakšavanja svakodnevnog života korisnika. Manipulator ima 6 stupnjeva slobode, čime se ostvaruje puna sloboda pozicioniranja alata u prostoru. Alat se sastoji od grabilice sa 3 prsta, što omgućuje stabilne hvatove velike količine svakodnevnih objekata. Ruka je dizajnirana kako bi bila kompatibilna sa raznim dostupnim električnim invalidskim kolicima, ali je zbog svoje kvalitetne izrade i specifikacija našla široku primjenu u znanosti.

Slika 3.3: Kinova Jaco robotska ruka

3.2.1. Tehničke specifikacije

Glavne prednosti Jaco robotske ruke su relativno niska potrošnja električne energije i malena masa komponenata. Aktuatori na zglobovima opremljeni su raznim senzorima kroz koje korisnik može dobiti veliku količinu povratnih informacija o stanju manipulatora.

Opće specifikacije

Potrebni ulazni napon kreće se od 18V do 29V, a to su vrijednosti lako ostvarive baterijom ugradivom u električna invalidska kolica. Masa od 5.6 kg osigurava jednostavnu montažu i sigurnost pri upravljanju, što je također vrlo bitno za primarnu svrhu ove robotske ruke. Detaljan prikaz općih specifikacija manipulaotra nalazi se u tablici 3.1.

Masa	$5.6 \pm 5\%$ [kg]
Ulazni napon	$18 - 29$ [VDC]
Ulazna struja	$2 - 10$ [A]
Srednja snaga	40 [W]
Frekvencija upravljanja	100 [Hz]
Max. teret na alatu pri srednjoj ispruženosti	1.5 [kg]
Max. teret na alatu pri maksimalnoj ispruženosti	1 [kg]
Dohvat	90 [cm]
Linearna brzina alata	$5 - 15$ [cm/s]
Sila stiska prsta	7 [N]

Tablica 3.1: Prikaz tehničkih specifikacija Jaco robotske ruke

Aktuatori

Manipulator se sastoji od 2 grupe od 3 identična aktuatora, modela K-75+ i K-58. Veći aktuatori (K-75+) čine 3 donja zgloba koji podnose najveće terete, dok manji aktuatori (K-58) zauzimaju mjesto zglobova čija je primarna funkcija postavljanje orijentacije alata u prostoru. Postoje još 3 aktuatora koji preko mehanizma arhimedovog vijka pokreću prste. Detaljan pregled specifikacija aktuatora nalazi se u tablicama 3.2, 3.3, 3.4.

Slika 3.4: a) aktuator K-75+ b) aktuator K-58

Masa	$0.64 \pm 2\%$ [kg]
Promjer	$74.5(+0.00/-0.03)$ [mm]
Visina	67 [mm]
Maksimalna brzina	8 [RPM]
Apsolutna pogreška pozicije	$\pm 0.5°$
Ulazni napon	$18-29$ [VDC]
Nominalni moment	15 [Nm]
Maksimalni moment	26 [Nm]

Tablica 3.2: Prikaz tehničkih specifikacija većih aktuatora.

Masa	$0.39 \pm 2\%$ [kg]
Promjer	$58(+0.00/-0.03)$ [mm]
Visina	69 [mm]
Maksimalna brzina	10 [RPM]
Apsolutna pogreška pozicije	$\pm 0.5°$
Ulazni napon	$18-29$ [VDC]
Nominalni moment	4 [Nm]
Maksimalni moment	7 [Nm]

Tablica 3.3: Prikaz tehničkih specifikacija manjih aktuatora

Senzori

Jaco robotska ruka opremljena je velikom količinom senzora. Svaki zglob posjeduje senzore pozicije, temperature, struje i napona.

Senzori pozicije služe za bilježenje zakreta svakog pojedinačnog zgloba manipula-

Maksimalna brzina	600 [RPM]
Ulazni napon	$18 - 29$ [VDC]
Nominalni moment	15 [mNm]
Maksimalni moment	30 [mNm]

Tablica 3.4: Prikaz tehničkih specifikacija aktuatora prstiju

tora, diferenciranjem zakreta dobivamo informaciju o brzini okretanja. Informacijom o temperaturi aktuatora osiguravamo manipulator od moguće štete uzrokovane prekomjernim zagrijavanjem. Pomoću senzora struje moguće je dobiti povratnu informaciju o momentu razvijenom na pojedinom aktuatoru, ovaj zaključak proizlazi iz dobro poznatog identiteta:

$$m_m \approx k\, i_a; \tag{3.1}$$

Ovdje i_a predstavlja trenutnu struju armature akturatora, dok je k konstanta definirana specifikacijama motora.

3.2.2. Računalno sučelje

Jaco robotska ruka s računalom se povezuje preko USB 2.0 (*engl. Universal Serial Bus*) sučelja. SDK[1] Jaco robotske ruke kompatibilan je s novijim inačicama Linux Ubuntu i Windows operativnih sustava, a uz API sadrži grafičko upravljačko sučelje i bazu primjera korištenja API biblioteka.

[1]SDK (engl. Software Development Kit) - paket je programskih biblioteka, alata i uputa za razvoj vlastitih aplikacija za određeni programski ili sklopovski sustav.

4. Programska podrška

Ovo poglavlje sadrži opise svih programskih paketa, biblioteka i elemenata korištenih u ovom radu. Kratko opisujemo karakteristke i uloge svake pojedine stavke programskog dijela arhitekture sustava. Potpoglavlja koja opisuju ROS i Gazebo sadrže nešto detaljnije opise, jer smatramo da je razumijevanje načela njihova rada ključno za razumijevanje načela rada sustava.

4.1. ROS

Slika 4.1: ROS logotip

ROS (engl. *Robot Operating System*) je operacijski sustav koji omogućuje jednostavno povezivanja alata i biblioteka potrebnih za razne izvedbe u robotici. ROS je razvijen 2007. godine pod imenom *switchyard* unutar laboratorija za umjetnu inteligenciju sveučilišta Stanford, kao razvojni alat za STAIR[1] robota. 2008. godine razvoj ROS-a preuzima institut Willow Garage iz Kalifornije. Od 2013. na dalje ROS postaje potpuno open-source i razvoj preuzima Open Source Robotics Foundation.

Glavna gradivna jedinica svake izvedbe u ROS-u je paket. Paketi u sebi sadrže C++/Python aplikacije koje nazivamo čvorovima (engl. *nodes*). Čvorove koristimo za komunikaciju s hardwareom (aktuatorima , senzorima ...), upravljanje simulacijama i prikazivanje podataka. Izvedbe se u stvarnosti najčešće sastoje od više paketa, stoga se prava prednost ROS-a krije u ostvarivanju brze i učinkovite peer-to-peer komunikacije među čvorovima.

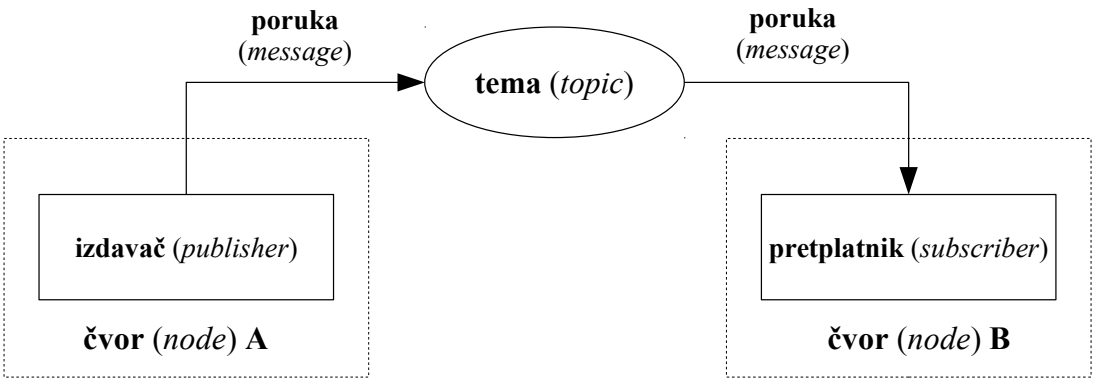
Komunikacija u ROS-u ostvaruje se mrežno (TCP/IP i UDP/IP protokolom) pomoću tema (engl. *topics*) i poruka (engl. *messages*). Uz gore navedeno, ROS nudi

[1] **STAIR** - STanford Artificial Intelligence Robot

razne naredbe unutar terminala koje olakšavaju prikaz podataka te dijagnostiku pri uklanjanju pogrešaka u kodu. Treba napomenuti da postoje i druge opcije za izvedbu aplikacija u robotici , kao što je OROCOS (engl. *Open Robot Control Software*). ROS je za ovaj projekt izabran zbog jednostavnosti izvedbe i postojećih paketa za JACO robotsku ruku.

4.1.1. Osnovni koncepti

Osnovna komunikacija u ROS-u može se vizualizirati u obliku grafa povezanih čvorova i tema. Ovakva struktura daje praktičan način prikazivanja sustava na najvišoj razini, čime se olakšava programska izvedba logike upravljanja.



Slika 4.2: Čvor A i čode B ostvaruju jednosmjernu komunikaciju koristeći topic

Čvorovi spremaju podatke u standardizirane strukture koje zovemo porukama te ih objavljuju na odgovarajuće mrežne lokacije koje nazivamo temama. Svaka tema prima samo jednu točno definiranu vrstu poruke i njen sadržaj dostupan je svim čvorovima na mreži. Izdavač (engl. *publisher*) je objekt unutar čvora koji šalje poruku na temu, dok pretplatnik (engl. *subscriber*) čini njegov komplement i čita poruku trenutno sadržanu na temi.

Ovaj način komunikacije omogućuje distribuciju raznih procesa na više računala što izvedbe čini manje ovisnima o sistemskim zahtjevima i izboru programskog jezika za pojedine čvorove.

4.1.2. Ostale korištene funkcionalnosti

Uz navedene osnovne funkcionalnosti , ROS sadrži neke složenije koncepte od kojih ćemo ovdje izdvojiti servise i parametarski server.

Servisi

U ovom radu koriste se servisi (engl. *services*). Pomoću servisa čvor A može direktno zahtijevati odgovor čvora B preko mreže. Čvor A objavljuje servis na mrežnu lokaciju sličnu temi, na tu lokaciju čvor B šalje zahtjev i adresu za spremanje odgovora. Čvor A učitava zahtjev s mrežne lokacije, obrađuje ga i odgovor sprema na adresu.

Zahtjev i odgovor najčešće su standardne ROS poruke, što znači da se pomoću servisa omogućuje korištenje funkcija čvora A u čvoru B. Ova funkcionalnost vrlo je korisna za distribuirane izvedbe procesno intenzivnih zadataka.

Parametarski server

Parametrima (engl. *parameters*) nazivamo karakteristične veličine unutar ROS čvorova koje se koriste pri izračunima varijabli. Glavna razlika između parametra i varijable nalazi se u frekvenciji promjene, koja je za parametre znatno niža.

ROS nudi mogućnost učitavanja parametara s mrežnih lokacija čime se omogućuje jednostavno fino podešavanje ponašanja čvora pri testiranju. Ovakvim pristupom se izbjegava potreba ponovnim kompiliranje koda pri svakoj promjeni parametara.

4.1.3. ros control

ROS paket `ros_control`ostvaruje generičke PID regulatore za razna sklopovska sučelja unutar ROS arhitekture. Jednostavna i standardizirana integracija s ostatkom ROS ekosustava čini ovaj paket čestom komponentom sustava izvedenih u ROS-u.

Kako bi se postiglo poopćenje PID upravljanja neovisno o konkretnom robotu (tj. o API-ju), API funkcije robota "umataju" se unutar apstraktnih `hardware_interface` klasa koje čine ROS-API sučelje. Umjesto da regulatori šalju i čitaju podatke koristeći API funkcije direktno, ove operacije obavljaju se pozivanjem generičkih funkcija unutar kojih su "umotane" API funkcije . Kako bi primjenio postojeći sustav na drugog robota (uz dostupnost istog seta naredbi koje regulator koristi), korisnik mora samo uklopiti odgovarajuće API funkcije unutar generičkih. Za dobar dio robota ovo sučelje već postoji radi open-source karaktera ROS ekosustava.

4.2. Matlab

Slika 4.3: Mathworks Matlab logo

Matlab je računalno okruženje namjenjeno rješavanju širokog spektra tehničkih, računalnih i znanstvenih problema. U užem smislu Matlab je viši programski jezik četvrte generacije koji omogućava manipulaciju matricama, numeričko računanje, iscrtavanje funkcija i još velik broj korisnih aplikacija. Unutar Matlaba nalazi se veliki broj programskih paketa koji se koriste za specifične probleme, a nama je bitan paket za simbolički račun. U ovom radu Matlab koristimo pri razvoju rješenja kinematike manipulatora za preliminarno testiranje matematičke provedivosti zamišljenog koncepta upravljačke petlje.

4.3. OpenCV

Slika 4.4: OpenCV logo

OpenCV je biblioteka otvorenog koda koja je revolucionarizirala područje računalnog vida. Mnogobrojnim ugrađenim funkcijama koje su dobro dokumentirane omogućava početnicima lagani početak bavljenja računalnim vidom. U ovom radu korištene su ugrađene funkcije dostupne u biblioteci za obradu slike od kojih valja izdvojiti brze izvedbe algoritma za pronalaženje kontura na slici, određivanje obujmica kontura te implementacije morfoloških operacija nad slikom.

4.4. OpenKinect/libfreenect

Slika 4.5: OpenKinect logo

OpenKinect je otvorena zajednica programera entuzijasta koji su razvili biblioteku *libfreenect* kao biblioteku otvorenog koda koja daje na raspolaganje sučelje za upravljanje Kinect uređajima na različitim platformama. Funkcijama za dohvat podataka s priključenog Kinect uređaja moguće je dohvatiti trenutnu dubinsku sliku u preglednom obliku koji je kompatibilan s bibliotekom OpenCV kako bi se ona mogla dalje procesirati.

4.5. KDL

KDL (engl. *Kinematics Dynamics Library*) je open-source C++ biblioteka razvijena za modeliranje i računanje kinematičkih lanaca. Pomoću ove biblioteke moguće je jednostavno programski rješavati direktnu i inverznu kinematiku te dinamiku manipulatora sa manje od 7 stupnjeva slobode. Nama je vrlo korisna mogućnost konstrukcije Jacobijeve matrice manipulatora (definirana kasnije), čiji se pseudoinverz pokazuje vrlo važnim rješenju problema praćenja putanje u općem slučaju. KDL također vodi računa o izbjegavanju singulariteta u matricama, čime je osiguran kontinuiran rad upravljačkog algoritma.

4.6. Gazebo

Slika 4.6: Gazebo logo

Gazebo je open-source programski paket za simulaciju robotskih sustava, njihovih senzora i okoline. Kako bi se postiglo simuliranje dinamike robota korsitimo DART, jednu od nekoliko biblioteka ovog tipa[2] koje Gazebo podržava. Kvalitetno iscrtavanje 3d modela korištenog robota u Gazebu je omogućeno korištenjem OGRE 3d modula.

Semantički opis robota, njegovih komponenti, prijenosa i zglobova Gazebo dobiva iz pripadajuće URDF datoteke. URDF je vrlo često korišten format za opis kinematičkih lanaca koji nudi kvalitetnu integraciju u sa Gazebom i brojnim drugim robotičkim

[2]DART(engl . Dynamic Animation and Robotics Toolkit) - `http://dartsim.github.io/`

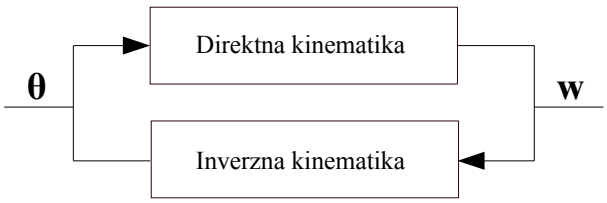
aplikacijama. Proces pripreme simulacije znatno ubrzava činjenica da gotovo svaki komercijalni robotski manipulator ima pripadajuću URDF datoteku.

4.6.1. ROS integracija Gazebo paketa

Glavni razlog za korištenje ovog simulacijskog paketa uska je integracija sa ROS-om , čime postižemo jednostavan transfer sustava razvijenog na simulaciji na stvarni manipulator. Gazebo simulaciju moguće je potpuno upravljati kroz ROS kroz regulatore i sučelja ostvarene `ros_control` paketom, što nam omogućuje simuliranje korištenjem koda za stvarni manipulator. Ova kompatibilnost ostvaruje se kroz `gazebo_ros`, `gazebo_msgs`, `gazebo_plugins` ROS pakete. Vrstu sklopovskog sučelja simulacije definiramo unutar URDF datoteke te se pri pokretanju Gazeba unutar ROS okruženja ono objavljuje kao stvarno. Zahvaljujući ovoj činjenici, sustav možemo razvijati za simulaciju i stvarni sustav istovremeno, što ubrzava proces razvoja.

5. Kinematika manipulatora

Modularna arhitektura našeg sustava dopušta integraciju manipulatora sa različitim razinama kompleksnosti API-ja. Unatoč tome što većina komercijalnih manipulatora dolazi sa već izvedenim naprednim kinematičkim funkcijama, smatramo da je radi kompletnosti izvedbe bitno predviditi i slučaj kada su dostupne samo naredbe pomicanja zglobova. Ovo poglavlje prikazuje teoretsku pozadinu programskog rješenja kinematike manipulatora izvedenog unutar našeg sustava na primjeru Jaco robotske ruke. Pošto Jaco robotska ruka ima 6 stupnjeva slobode , sa stajališta složenosti kinematike ovaj izbor predstavlja svojevrsni najgori slučaj. Kinematičko rješenje provjeravamo u Matlab programskom paketu kako bi ustanovili numeričku stabilnost rješenja i moguće nedostatke.



Slika 5.1: Model Jaco robotske ruke sa označenim parametrima

U idealnom slučaju, rješavanje problema kinematike manipulatora sa do 6 stupnjeva slobode zahtjeva sintezu matrica homogene transformacije između koordinatnih sustava baze i alata. Iz članova dobivene matrice potom analitički dobivamo izraze za Kartezijsku poziciju i orijentaciju alata u ovisnosti o zakretima zglobova. Sintezu direktne kinematike započinjemo D-H postupkom postavljanja koordinatnih sustava vezanih uz zglobove. Određivanjem matrica homogenih transformacija između tih sustava dobivamo surjektivnu[1] funkciju koja prostor zglobova preslikava u 6-dimenzionalni prostor pozicije.

Problem inverzne kinematike znatno je složeniji jer zahtjeva rješavanje sustava nelinearnih jednadžbi s više nepoznanica. Za složenije manipulatore ovaj problem je

[1] Uz pretpostavku restrikcije zakreta zglobova na vrijednosti $[0, 2\pi]$.

netrivijalan te rješenje nije uvijek moguće pronaći koristeći osnovne matematičke metode. Još jedna otežavajuća okolnost nalazi se u činjenici da će dobivena rješenja često biti višestruka, a pronalaženje odgovarajućeg ovisi o konstrukciji i poziciji manipulatora.

Pronalaženje analitičkih rješenja inverzne kinematike pomoću računalnog algoritma još je složeniji zadatak jer zahtjeva implementaciju univerzalnog pristupa rješavanju nelinearnih jednadžbi čija su rješenja višestruka. Pronašli smo da je ovaj problem moguće zaobići koristeći iterativnu metodu baziranu na pronalaženju (pseudo)inverza matrice Jakobijana manipulatora, koja je puno prikladnija za programsku izvedbu.

U prvom odlomku opisati ćemo osnovne postupke rješavanja kinematičkog problema na primjeru Jaco robotske ruke. Pri tome ćemo radi sažetosti izlaganja preskočiti pojedinosti D-H postupka i izvod matrica homogene transformacije. Drugi odlomak sadrži opis ideje rješenja inverzne kinematike koje koristimo u računalnom algoritmu. Rezultate zamišljenog postupka potom ispitujemo koristeći Matlab.

5.1. Direktna i inverzna kinematika Jaco robotske ruke

5.1.1. Direktna kinematika

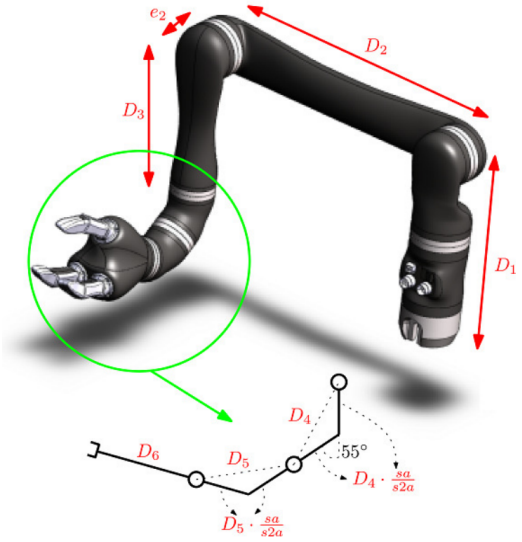
Problem direktne kinematike rješavamo tako da prvo postavimo koordinatne sustave koji odgovaraju svakom pojedinačnom zglobu prema pravilima D-H postupka. Iz međusobnih udaljenosti i razlika u orijentaciji ovih zglobova porizlaze D-H parametri koje koristimo za sintezu matrica homogenih transformacija. Množenjem dobivenih matrica dobivamo matricu koja izražava poziciju i orijentaciju alata izražene Kartezijevim koordinatama u ovisnosti o zakretu zglobova.

DH Parametri

Pri sintezi matrica transformacije ruke korišteni su D-H parametri iz službene dokumentacije za Jaco robotsku ruku prikazani u tablici 5.2. Za sintezu D-H parametara koristimo duljine članaka kao i parametre koji su iz njih izvedeni, prikazani su na slici 5.2. Parametre d_{4b}, d_{5b} i d_{6b} određujemo iz izraza 5.1, 5.2, 5.3.

$$d_{4b} = D_3 + D_4 \frac{1}{2 \cdot \cos(a)} = D_3 + D_4 \frac{\sin(a)}{\sin(2a)} \tag{5.1}$$

$$d_{5b} = D_4 + D_5 \frac{1}{2 \cdot \cos(a)} = D_4 + D_5 \frac{\sin(a)}{\sin(2a)} \tag{5.2}$$



Slika 5.2: Model Jaco robotske ruke sa označenim parametrima

D_1	0.2755 [m]
D_2	0.4100 [m]
D_3	0.2073 [m]
D_4	0.0743 [m]
D_5	0.0743 [m]
D_6	0.1687 [m]
e_2	0.0098 [m]
a	$\frac{11 \cdot \pi}{72}$ [m]
d_{4b}	$D_3 + D_4 \frac{\sin(a)}{\sin(2a)}$ [m]
d_{5b}	$D_4 + D_5 \frac{\sin(a)}{\sin(2a)}$ [m]
d_{6b}	$D_6 + D_5 \frac{\sin(a)}{\sin(2a)}$ [m]

Tablica 5.1: Konstrukcijski parametri Jaco robotske ruke

$$d_{6b} = D_6 + D_5 \frac{1}{2 \cdot \cos(a)} = D_6 + D_5 \frac{\sin(a)}{\sin(2a)}. \tag{5.3}$$

Ne ulazeći u pojedinosti DH metode, potrebno je napomenuti da d i a parametri predstavljaju međusobne odmake koordinatnih sustava zglobova, dok kut θ predstavlja rotaciju koordinatnih sustava oko vlastite osi rotacije. Navedenim parametrima dodajemo i razlike u orijentaciji rotacijskih osi zglobova α. Konačni parametri prikazani su u tablici 5.2.

$$\theta_1 = -q_{1_{Jaco}} \tag{5.4}$$

k	α_k	a_k	d_k	θ_k
1	$\pi/2$	0	D_1	θ_1
2	π	D_2	0	θ_2
3	$\pi/2$	0	$-e2$	θ_3
4	$\frac{11\pi}{36}$	0	$-d_{4b}$	θ_4
5	$\frac{11\pi}{36}$	0	$-d_{5b}$	θ_5
6	π	0	$-d_{6b}$	θ_6

Tablica 5.2: DH parametri

$$\theta_2 = q_{2_{Jaco}} - \frac{\pi}{2} \tag{5.5}$$

$$\theta_3 = q_{3_{Jaco}} + \frac{\pi}{2} \tag{5.6}$$

$$\theta_4 = q_{4_{Jaco}} \tag{5.7}$$

$$\theta_5 = q_{5_{Jaco}} - \pi \tag{5.8}$$

$$\theta_6 = q_{6_{Jaco}} + \frac{5 \cdot \pi}{9} \tag{5.9}$$

Odnos kuteva u matematičkom modelu i stvarnih kuteva zglobova Jaco ruke dan je izrazima 5.4, 5.5, 5.6, 5.7, 5.8, 5.9. Varijabla $q_{i_{Jaco}}$ predstavlja trenutačnu rotaciju i-tog zgloba Jaco robotske ruke očitanu kroz API, dok θ_i predstavlja stvarni zakret i-tog zgloba.

Matrice homogene transformacije

Matrica homogene transformacije između dva koordinatna sustava postavljena prema D-H postupku određena je izrazom 5.10. Vektor $\mathbf{p} = [x\ y\ z]^T$ sadrži poziciju ishodišta k-tog koordinatnog sustava u koordinatnom sustavu k-1. Matrica $\mathbf{R}$ je matrica rotacije koja definira orijentaciju koordinatnog sustava k u koordinatnom sustavu k-1.

$$\mathbf{T}^{\mathbf{k}}_{\mathbf{k-1}} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_k & -\cos\alpha_k \sin\theta_k & \sin\alpha_k \sin\theta_k & a_k \cos\theta_k \\ \sin\theta_k & \cos\alpha_k \cos\theta_k & -\sin\alpha_k \cos\theta_k & a_k \sin\theta_k \\ 0 & \sin\alpha_k & \cos\alpha_k & d_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.10}$$

Uvrštavajući podatke iz tablice 5.2 u izraz 5.10 dobivamo matrice homogenih tran-

sformacija za koordinatne sustave Jaco robotske ruke.

$$\mathbf{T_0^1} = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & 0 \\ \sin\theta_1 & 0 & -\cos\theta_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.11}$$

$$\mathbf{T_1^2} = \begin{bmatrix} \cos\theta_2 & \sin\theta_2 & 0 & a_2\cos\theta_2 \\ \sin\theta_2 & -\cos\theta_2 & 0 & a_2\sin\theta_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.12}$$

$$\mathbf{T_2^3} = \begin{bmatrix} \cos\theta_3 & 0 & \sin\theta_3 & 0 \\ \sin\theta_3 & 0 & -\cos\theta_3 & 0 \\ 0 & 1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.13}$$

$$\mathbf{T_3^4} = \begin{bmatrix} \cos\theta_4 & -\cos\alpha_4\sin\theta_4 & \sin\alpha_4\sin\theta_4 & 0 \\ \sin\theta_4 & \cos\alpha_4\cos\theta_4 & -\sin\alpha_4\cos\theta_4 & 0 \\ 0 & \sin\alpha_4 & \cos\alpha_4 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.14}$$

$$\mathbf{T_4^5} = \begin{bmatrix} \cos\theta_5 & -\cos\alpha_5\sin\theta_5 & \sin\alpha_5\sin\theta_5 & 0 \\ \sin\theta_5 & \cos\alpha_5\cos\theta_5 & -\sin\alpha_5\cos\theta_5 & 0 \\ 0 & \sin\alpha_5 & \cos\alpha_5 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.15}$$

$$\mathbf{T_5^6} = \begin{bmatrix} \cos\theta_6 & \sin\theta_6 & 0 & 0 \\ \sin\theta_6 & -\cos\theta_6 & 0 & 0 \\ 0 & 0 & -1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.16}$$

Množenjem ovih matrica dobivamo matricu homogene transformacije između baze robota i alata 5.17

$$\mathbf{T_{alat}^{baza}} = \mathbf{T_1^0}\,\mathbf{T_2^1}\,\mathbf{T_3^2}\,\mathbf{T_4^3}\,\mathbf{T_5^4}\,\mathbf{T_6^5} \tag{5.17}$$

Članove ove matrice određujemo koristeći matlab skriptu, rezultat računa i skripta ispisani su u dodatku. Iz matrice $\mathbf{T_{alat}^{baza}}$ dobivamo potpunu informaciju o položaju alata u baznom koordinatnom sustavu kao funkciju zakreta zglobova:

$$\mathbf{w}(\boldsymbol{\theta}) = \begin{bmatrix} \mathbf{p} \\ \mathbf{r}(\mathbf{R}) \end{bmatrix} \tag{5.18}$$

Vektor 5.18 još nazivamo vektorom konfiguracije alata.

5.1.2. Inverzna kinematika

Ciljane konfiguracije izvršnog člana u $\mathbb{R}^6$ dane su vektorom $\vec{\mathbf{t}} = (\mathbf{t_1}, \ldots, \mathbf{t_k})^T$, a trenutne konfiguracije vektorom $\vec{\mathbf{w}} = (\mathbf{w_1}, \ldots, \mathbf{w_k})^T$. Kod mehaničkih manipulatora , $\vec{\mathbf{w}}$ je funkcija pozicija zglobova $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)^T$, pa za ovaj slučaj zapisujemo $\vec{\mathbf{w}} = \vec{\mathbf{w}}\left(\vec{\boldsymbol{\theta}}\right)$. Problem inverzne kinematike definiramo kao pronalaženje vektora $\vec{\boldsymbol{\theta}} = (\boldsymbol{\theta_1} \ldots \boldsymbol{\theta_k})^T$ koji zadovoljava izraz:

$$\mathbf{t_i} = \mathbf{w_i}(\boldsymbol{\theta_i}) \tag{5.19}$$

Ono što čini problem inverzne kinematike problemom u užem smislu jest netrivijalnost rješavanja sustava jednadžbi 5.19 kako bi dobili $\boldsymbol{\theta}_\mathbf{i}(\mathbf{t_i})$. Razvijeni su različiti pristupi ovom problemu od kojih se mnogi oslanjaju na numeričke metode ili metode pronalaženja minimuma odgovarajuće funkcije.

Korištenjem analitičke metode rješavanja opisane u uvodu ovog poglavlja, vektor $\boldsymbol{\theta}_\mathbf{i}(\mathbf{t_i})$ dobiva se rješavanjem sustava jednadžbi dobivenog iz vektora konfiguracije alata 5.18. Promatrajući puni izraz za vektor konfiguracije alata $\mathbf{w}$, primjećujemo da je analitičko rješavanje ovog problema netrivijalan zadatak čije poopćavanje zahtjeva iznimno složen algoritam. Ovo nas dovodi do zaključka da univerzalni algoritam rješavanja inverzne kinematike zahtjeva drugačiji pristup.

5.2. Iterativno rješavanje kinematičkog problema

Kako bi izbjegli komplicirano analitičko rješavanje problema inverzne kinematike, koristimo metodu koja zaobilazi rješavanje sustava jednadžbi 5.19. Prvi se korak sastoji od određivanja matrice Jakobijana manipulatora koja povezuje brzine okretanja zglobova sa linearnim i rotacijskim brzinama alata.

Pronalaženjem inverza ove matrice, moguće je odrediti potrebne brzine rotacije zglobova kako bismo ostvarili željene brzine gibanja alata. Sintezom upravljačke petlje moguće je ostvariti praćenje referentne veličine u obliku zadanog vektora konfiguraciije alata, čime ostvarujemo iterativni postupak rješavanja inverzne kinematike.

5.2.1. Jakobijan manipulatora

Definiramo Jacobijevu matricu manipulatora kao promjenu vektora konfiguracije $\mathbf{w_i}$ u ovisnosti o promjeni $\boldsymbol{\theta}_\mathbf{i}$. Radi jednostavnosti zapisa, u nastavku teksta ispuštamo indekse ovih vektora. Sada je izraz za Jacobijevu matricu:

$$\mathbf{J} = \frac{d\mathbf{w}}{d\boldsymbol{\theta}} \tag{5.20}$$

Počinjemo pretpostavkom da za dovoljno malene vrijednosti $\Delta\mathbf{w}$ vrijedi 5.21:

$$\Delta\mathbf{w} \approx \mathbf{J}\Delta\boldsymbol{\theta} \tag{5.21}$$

Inverzom matrice $\mathbf{J}$ možemo dobiti sljedeći vrlo koristan identitet:

$$\Delta\boldsymbol{\theta} \approx \mathbf{J}^{-1}\Delta\mathbf{w} \tag{5.22}$$

Identitet 5.22 važan je jer zahvaljujući njemu ostvarujemo regulator brzine definiran u 5.2.2.

U generalnom slučaju Jacobijeva matrica predstavlja matricu dimenzija $n \times k$ gdje je $\mathbf{w_i} = (w_1, \ldots, w_k)^T$:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial w_1}{\partial \theta_1} & \frac{\partial w_1}{\partial \theta_2} & \cdots & \frac{\partial w_1}{\partial \theta_n} \\ \frac{\partial w_2}{\partial \theta_1} & \frac{\partial w_2}{\partial \theta_2} & \cdots & \frac{\partial w_2}{\partial \theta_n} \\ \vdots & & & \\ \frac{\partial w_k}{\partial \theta_1} & \frac{\partial w_k}{\partial \theta_2} & \cdots & \frac{\partial w_k}{\partial \theta_n} \end{bmatrix} \tag{5.23}$$

U slučaju kada ova matrica predstavlja prostor vektora konfiguracije $\mathbf{w}$, vrijedi $k = 6$. Ove parametre ponekad nije jednostavno analitički odrediti, stoga je korištena geometrijska metoda kojom zaobilazimo deriviranje. Neka je $\mathbf{p}_j$ pozicija j-tog zgloba u baznom koordinatnom sustavu i neka je $\mathbf{z}_j$ jedinični vektor osi rotacije j-tog zgloba. Komponente rotacijskog dijela Jakobijana $J_{\boldsymbol{\omega}_j}$ jednake su jediničnim vektorima osi rotacije zgloba u baznom koordinatnom sustavu:

$$J_{\boldsymbol{\omega}_j} = \frac{\partial \mathbf{w}}{\partial \theta_j} = \mathbf{z}_j \tag{5.24}$$

Uz pretpostavku da je rotacija mjerena u radijanima i da je smjer rotacije određen pravilom desne ruke , za translacijski dio Jakobijana $J_{\boldsymbol{v}_j}$ članovi iznose:

$$J_{\boldsymbol{v}_j} = \frac{\partial \mathbf{w}}{\partial \theta_j} = \mathbf{z}_j \times (\mathbf{w} - \mathbf{p}_j) \tag{5.25}$$

U slučaju translacijskog zgloba, računanje člana još je jednostavnije jer je promjena pozicije jednaka jediničnom vektoru "osi rotacije" zgloba:

$$J_{\boldsymbol{v}_j} = \frac{\partial \mathbf{w}}{\partial \theta_j} = \mathbf{z}_j \tag{5.26}$$

Dobivena Jacobijeva matrica ima strukturu 5.27, 5.28 :

$$\mathbf{J} = \begin{bmatrix} J_{\boldsymbol{v}} \\ J_{\boldsymbol{\omega}} \end{bmatrix} \tag{5.27}$$

$$\mathbf{J} = \begin{bmatrix} \mathbf{z}_0 \times (\mathbf{w} - \mathbf{p}_0) & \mathbf{z}_1 \times (\mathbf{w} - \mathbf{p}_1) & \dots & \mathbf{z}_{n-1} \times (\mathbf{w} - \mathbf{p}_{n-1}) \\ \mathbf{z}_0 & \mathbf{z}_1 & \dots & \mathbf{z}_{n-1} \end{bmatrix} \tag{5.28}$$

Izrazom 5.22 linearizirano model gibanja robota oko točke $\boldsymbol{\theta}$. Uz odgovarajuću frekvenciju osvježavanja matrice $\mathbf{J}$, moguće je dobiti kvalitetnu aproksimaciju $\Delta\boldsymbol{\theta}$ za željeni $\Delta\mathbf{w}$.

Problem kod ovog pristupa nastaje u slučajevima kada matrica $\mathbf{J}$ nije kvadratna, što je slučaj kod svih manipulatora sa brojem zglobova različitim od 6. Nekvadratnu matricu nije moguće invertirati, što znači da metoda nije univerzalna. Radi potrebe za općim rješenjem koristimo Moore-Penroseov pseudoinverz, koji je svojevrsno poopćenje inverza matrice proizvoljnih dimenzija. Uz određene uvjete (ref), pseudoinverz Jakobijana određujemo izrazom 5.29:

$$\mathbf{J}^{+} = \left(\mathbf{J}^{\dagger}\mathbf{J}\right)^{-1}\mathbf{J}^{\dagger} \tag{5.29}$$

U izrazu 5.29 simbol † označava operaciju hermitske transpozicije. Ovom modifikacijom poopćavamo ideju izraza 5.22 na sve manipualtore te osiguravamo veću numeričku stabilnost naspram običnog inverza. U praksi se koriste naprednije metode koje osiguravaju dodatnu numeričku stabilizaciju dodatnim članovima, pa u praksi izraz 5.22 poprima oblik 5.30.

$$\Delta\boldsymbol{\theta} \approx \mathbf{J}^{+}\Delta\mathbf{w} + \mathbf{S} \tag{5.30}$$

U izrazu 5.30 matrica $\mathbf{S}$ predstavlja stabilizacijski član koji može biti ovisan o trenutnoj brzini, poziciji ili akceleraciji zglobova. Programskom izvedba pomoću KDL biblioteka oslanja se na poboljšanu varijantu metode 5.30 zvanu metoda prigušenih najmanjih kvadrata (*engl. damped least squares*). Za razliku od ranije navedenih metoda, ova ne traži najmanji vektor $\Delta\boldsymbol{\theta}$ koji zadovoljava 5.20. Metoda prigušenih najmanjih kvadrata nalazi vektor $\Delta\boldsymbol{\theta}$ koji pronalazi minimum iznosa:

$$\|\mathbf{J}\Delta\boldsymbol{\theta} - \Delta\mathbf{w}\|^2 + \lambda^2\|\Delta\boldsymbol{\theta}\|^2 \tag{5.31}$$

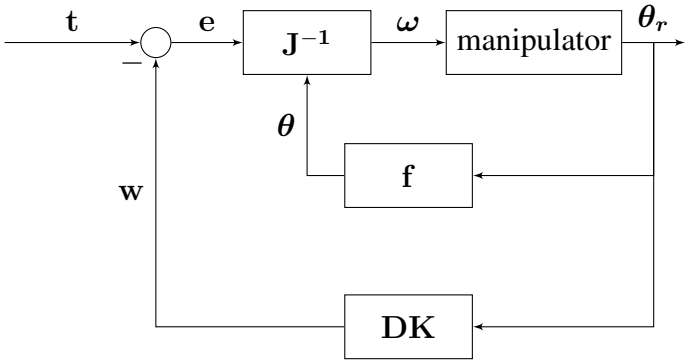
Izraz 5.31 može se svesti na izraz 5.32, gdje konstantu λ odabiremo ovisno o konstrukciji manipulatora prema nekoj od razvijenih metoda (ref). Jedan prijedlog korišten pri testiranju koncepta jest postavljanje konstante na vrijednost maksimalne dopuštene promjene vrijednosti nekog zgloba unutar jednog radnog ciklusa.

$$\Delta\boldsymbol{\theta} = \mathbf{J}^{T}\left(\mathbf{J}^{T}\mathbf{J} + \lambda^2\mathbf{I}\right)^{-1} \tag{5.32}$$

Pokazalo se da je ovakva izvedba superiorna prema ostalim metodama, što pokazujemo u odlomku 5.2.2.

5.2.2. Upravljačka petlja

Razmatranjem izraza 5.22 možemo zaključiti da smo posredno došli do metode čijom primjenom u upravljačkoj petlji u teoriji možemo ostvariti kretanje po putanji. Iterativnom primjenom izraza 5.22 moguće je ostvariti kretanje ruke prema željenom vektoru konfiguracije dokle god poznajemo pogrešku $\mathbf{e}$. Za sintezu Jacobijeve matrice potrebni su nam podaci o trenutnim pozicijama zglobova koji čine povratnu vezu petlje, dok je za računanje potrebne promjene pozicije potrebna informacija o željenom vektoru konfiguracije što je referentna veličina.



Slika 5.3: Blok dijagram izvedbe upravljače petlje manipulatora

Izrazom 5.33 definiramo vektor razlike translacija u iteraciji i. Njime je definirana razlika u poziciji između ciljane i trenutne konfiguracije $\mathbf{t}_T$ i $\mathbf{c}_T$ unutar baznog koordinatnog sustava.

$$\mathbf{e}_T = \mathbf{t}_T - \mathbf{c}_T \tag{5.33}$$

Kako bi potpuno definirali potrebnu promjenu konfiguracije, moramo poznavati i razliku u orijentaciji. Također, ova mjera razlike u orijentaciji mora biti kompatibilna sa geometrijskom izvedbom Jakobijana koju koristimo u izračunima.

$$\mathbf{e}_O = 0.5\left(\mathbf{R}_{x_c} \times \mathbf{R}_{x_t} + \mathbf{R}_{y_c} \times \mathbf{R}_{y_t} + \mathbf{R}_{z_c} \times \mathbf{R}_{z_t}\right) \tag{5.34}$$

Koristimo izraz 5.34 definiran u (RMPC-139), koji je pogodan za kinematičke lance koji ne posjeduju sferne zglobove. Elementi izraza predstavljaju stupce rotacijskih matrica koji korespondiraju sa orijentacijom koordinatnog sustava koji je njima definiran.

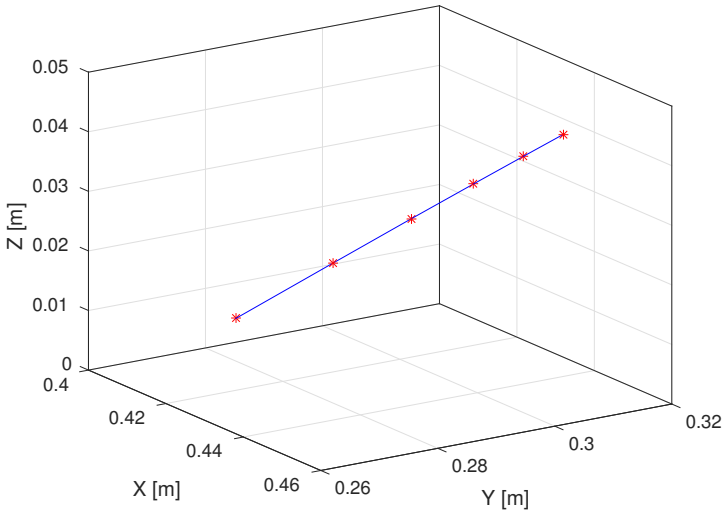
Razliku referentnog i stvarnog vektora konfiguracije alata sustav prema izrazu 5.22 množi inverzom matrice $\mathbf{J}$. Kao rezultat množenja dobivamo potrebne zakrete zglobova $\Delta\mathbf{q}$. Pri praćenju putanje manipulator nikad neće biti daleko od tražene pozicije, čime bez gubitka općenitosti dobivene potrebne pomake također možemo tretirati kao brzine.

$$\Delta\mathbf{q} \approx \boldsymbol{\omega} \tag{5.35}$$

Vektor funkcija $\mathbf{f}$ u ovom slučaju označava preračunavanje kuteva očitanih sa manipulatora ruke $\boldsymbol{\theta}_r$ u kuteve koji odgovaraju postavljenom matematičkom modelu $\boldsymbol{\theta}$. Dobivene kuteve koristimo kako bi osvježili matricu $\mathbf{J}$ i kako bi dobili trenutni vektor konfiguracije alata $\mathbf{w}$. Iz 5.3 primjećujemo kako se radi o strukturi sa P (proporcionalnim) regulatorom jediničnog pojačanja, no sustav je moguće proširiti složenijim oblicima regulatora.

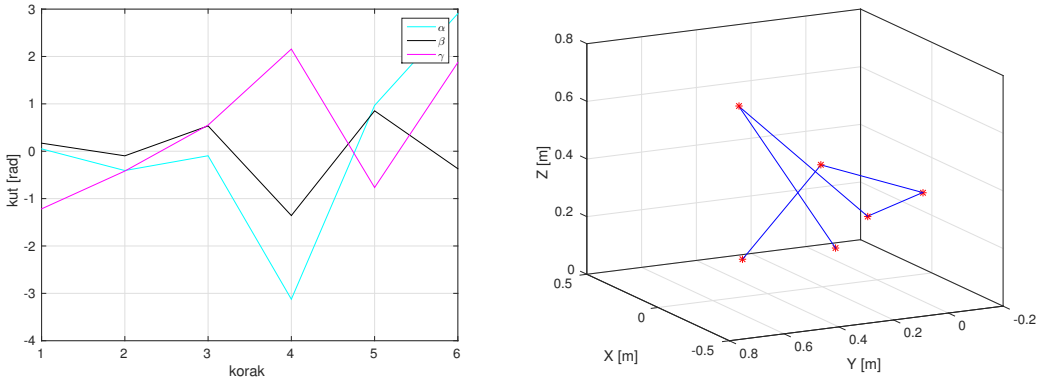
Provjera sustava

Zamišljenu upravljačku shemu sustava prikazanu na slici 5.3 provjeravamo u Matlabu. Prvi korak je rješavanje direktne kinematike te generiranje funkcije `jacoFK` koja kao argumente prima zakrete zglobova $\mathbf{q}$, a kao rezultat vraća vektor konfiguracija alata.



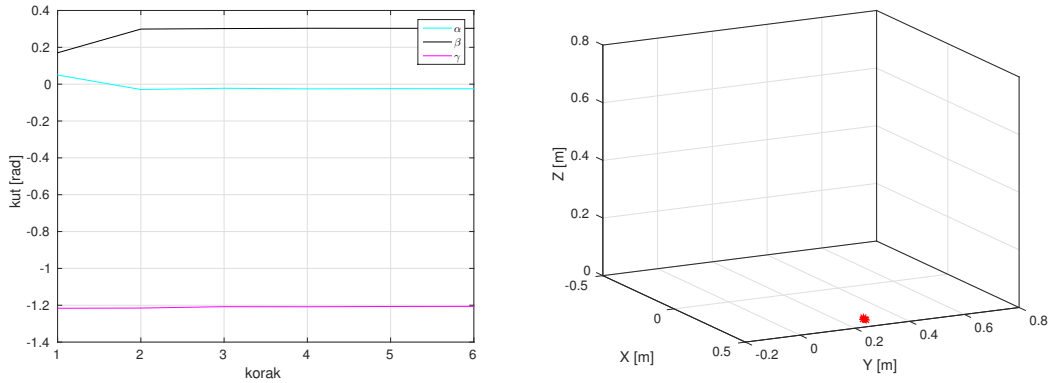
Slika 5.4: Kretanje matematičkog modela Jaco robotske ruke prema zadanoj točki. Orijentacija alata je konstantna.

Radna frekvencija detekcije kreće se oko 30 Hz , a matricu Jakobijana planiramo osvježavati oko 200 puta u sekundi. Iz ovih smjernica uzimamo kako ćemo pri testiranju osvježavati matricu Jakobijana 6 puta pri izvršavanju kretanja. Iz grafa 5.4 vidljivo je kako pri čistom translacijskom kretanju upravljačka petlja sa P regulatorom ostvaruje željenu konfiguraciju bez oscilacija u kretanju. Kretanje koje uključuje rotacije pokazalo se nešto većim izazovom za predloženu metodu. Sa slike 5.5 vidimo kakop pri isključivo rotacijskom kretanju koristeći izraz 5.22 ne možemo dobiti zadovoljavajuće rješenje. Ovaj slučaj jednostavan je dokaz ranije tvrdnje kako "obični" Jakobijan često nije dovoljno numerički stabilan. Kako bi doskočili ovom problemu koristimo



Slika 5.5: Promjena orijentacije inverzom Jakobijana. Slika a) (lijevo) sadrži translacijski odziv, b) (desno) sadrži odziv orijentacije sustava u Eulerovim kutevima.

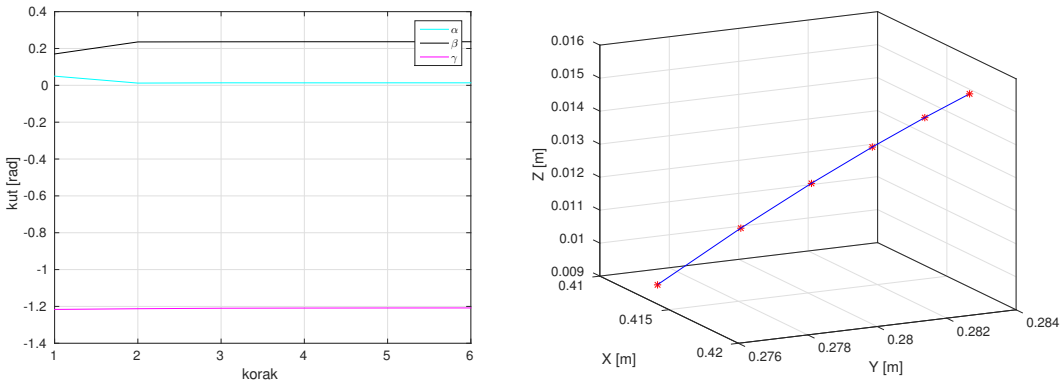
metodu prigušenih najmanjih kvadrata iz izraza 5.32 sa faktorom $\lambda = 0.2$.



Slika 5.6: Promjena orijentacije metodom prigušenih najmanjih kvadrata. Slika a) (lijevo) sadrži translacijski odziv, b) (desno) sadrži odziv orijentacije sustava u Eulerovim kutevima.

Iz slike 5.6 vidimo kako je ostvaren drastično bolji odziv. Ostaje nam još ispitati ponašanje pri promjeni cijelog vektora konfiguracije.

Rezultati prikazani na slici 5.7 dovoljni su nam kao dokaz da se metoda prigušenih najmanjih kvadrata u teoriji može koristiti za upravljanje manipulatorom sa 6 stupnjeva slobode. U ovom preliminarnom testiranju nismo uključivali model motora i dinamike manipulatora i stoga će konačnu programsku izvedbu biti potrebno testirati na simulaciji.



Slika 5.7: Promjena vektora konfiguraciije metodom prigušenih najmanjih kvadrata. Slika a) (lijevo) sadrži translacijski odziv, b) (desno) sadrži odziv orijentacije sustava predstavljene u Eulerovim kutevima.

6. Detekcijski algoritam

Upravljanje pokretima ruke može se izvesti na više načina. U ovom radu odlučeno je koristiti dubinsku kameru kojom se snimaju čovjekovi pokreti ruke te se svaka snimljlena slika prevodi u informaciju o trenutnoj lokaciji i orijentaciji ruke u trodimenzionalnom prostoru kojeg dubinska kamera promatra.

Problem detekcije poze u kojoj se nalazi ruka s nekom dozom uspješnosti riješen je i kod klasičnih slika (npr. //todo 3 citata), ali taj pristup većinom obilježava prevelika ovisnost o svjetlosnim parametrima slike te potreba za većim računalnim resursima. Takvi zahtjevi otežavaju detekciju u stvarnom vremenu na osobnom računalu.

Većina pristupa detekciji objekata kod klasične slike s intenzitetom svjetlosti u pojedinim točkama može se primijeniti i kod dubinske slike. Pri tome dubinsku sliku možemo promatrati kao klasičnu sivu sliku te raditi detekcije nad istom samo moramo imati na umu kako informacija koju sadrži pojedini slikovni element ne odgovara intenzitetu, već je mjera udaljenosti. Također uobičajena crno-bijela slika se sastoji od 8-bitnih vrijednosti što polučuje 256 različitih slikovnih elemenata te u većini slučaja možemo biti zadovoljni s tim. Kod dubinske slike to nije slučaj. Ako imamo 256 različitih slikovnih elemenata sposobni smo detektirati 256 različitih dubina, s čime ne možemo biti zadovoljni ako želimo detekciju na većem rasponu ($1m$) uz veliku preciznost ($1mm$). Zato je uobičajen zapis dubinske slike pomoću 2 okteta ili broja s jednostrukom preciznosti po IEEE 754 standardu. Ovaj potonji zapis kompatibilan je s korištenom OpenCV bibliotekom.

Razvijeni detekcijski algoritam podijeljen je u dvije slijedne komponente. Lokalizacija dlana u slici te određivanje parametara poze u kojoj se dlan nalazi.

6.1. Lokalizacija dlana u slici

Jednako kao i kod klasičnih slika prvi korak u preciznom detektiranju poze u kojoj se nalazi ruka jest određivanje njene lokacije na dubinskoj slici. Različite metode su predložene za rješavanje takvog problema. Pri tome rijetko koja metoda ne koristi

neki oblik pomagala kako bi što lakše i brže odredila lokaciju u dubinskoj slici te više vremena mogla posvetiti samom određivanju poze u kojoj se ruka nalazi.

Kako se ovaj rad orijentira na dubinske slike, a ne klasične slike s dodanim dubinskim kanalom, takvi pokušaji su zanemareni, ali valja spomenuti kako se filtriranjem isključivo boje kože može olakšati pretraga ruke. Neki od takvih pristupa koriste brzu detekciju lica i uzorkovanje parametara boje kože detektirane osobe radi poboljšavanja filtera boje pomoću dobivenih parametara.

6.1.1. Slične metode

Qian i suradnici razvili su jednu od najkvalitetnijih metoda danas [**?**]. Metoda inicijalno pretpostavlja kako se ruka nalazi najbliže kameri u odnosu na ostale objekte u dubinskoj slici. Također, pretpostavlja korištenje nereflektirajuće narukvice na zapešću kojom se lako može izolirati ruka od okoline algoritmom poplavljivanja interesnog područja (eng. *flood fill*).

Bolja inicijalna metoda preuzeta je iz rada Shottona i suradnika [**?**] u kojem detektiraju položaj čovjeka na dubinskoj slici. Implementirano prepoznavanje ljudskog kostura pomoću Kinect uređaja temelji se na njihovoj metodi. Tu inicijalnu metodu koriste Thompson i suradnici [**?**]. Metoda koristi jednostavne značajke $f_\theta(I, \mathbf{x})$ za svaki slikovni element $\mathbf{x}$ koje su invarijantne na dubinu $d_I(\mathbf{x})$ (Izraz 6.1).

$$f_\theta(I, \mathbf{x}) = d_I(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}) - d_I(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}) \tag{6.1}$$

Dalje za svaki slikovni element $\mathbf{x}$ učenjem šume nasumičnih stabala odluke (eng. *random decision forrest*) odredi se svega nekoliko parova odmaka $(\mathbf{u}, \mathbf{v})$ koji dobro binarno klasificiraju svaki slikovni element pripada li on naučenom skupu objekata ili ne.

Grupiranjem slikovnih elemenata koji su imali pozitivan odziv klasifikatora možemo izolirati ruku ili više njih na dubinskoj slici.

Iako je metoda uspješna i relativno brza, pogotovo na grafičkoj kartici za koju je primarno namijenjena, u slučaju ovog završnog rada nedostatak takve metode bilo bi trajanje njenog učenja. Naime, autori navode kako na klasteru od 1000 jezgara učenje šume od 3 stabla do dubine 20 traje 1 dan, a uzevši u obzir nedostatak navedenih resursa, takav pristup u ovom radu nije moguć.

6.1.2. Razvijena metoda

Dubinska slika kakvu stvara korištena dubinska kamera je veličine 640x480 slikovnih elemenata. Pri tome vrijednost svakog slikovnog elementa odgovara izračunatom disparitetu na toj poziciji. Pomoću *freenect* biblioteke takav podatak se može pretvoriti u dubinu u milimetrima.

Definiran je koordinatni sustav sa središtem u očištu kamere i osima x i y s odgovarajućim indeksom $(u+320, v+240)$ slikovnog elementa (kako bi središte koordinatnog sustava pomakli u središte slike). Osi x i y su skalirane na odmak od središta u milimetrima. Os z odgovara dubini slike također u milimetrima. Prirodom perspektivne projekcije kakvu daje dubinska kamera udaljenost od kamere ne odgovara udaljenosti do projekcijske ravnine. Samim tim se ne može izravno preslikati u z-os koordinatnog sustava, ali kako dlan na radnim udaljenostima ne zauzima više od 15% slike na tom području je aproksimacija prihvatljiva.

Jedna od mogućih aproksimativnih formula za preslikavanje dispariteta u udaljenost od kamere, kakvu koristi *freenect* biblioteka [**?**] je sljedeća (Izraz 6.2):

$$z_d(\mathbf{x}) = \frac{1000}{-0.00307 * d(\mathbf{x}) + 3.33} \tag{6.2}$$

Dobivenim vrijednostima dubine možemo izračunati x i y vrijednosti koordinantnog sustava na sljedeći način:

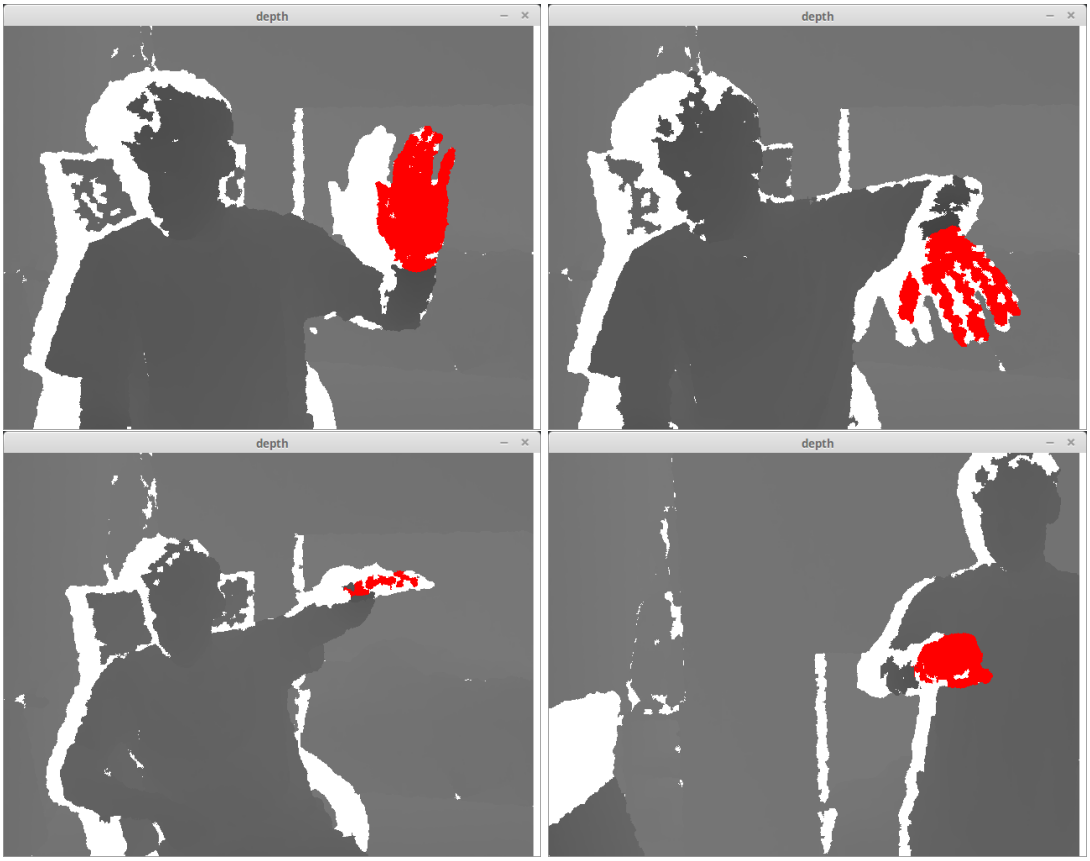
$$x(z, u) = 0.0021(v - 320)(z - 10) \tag{6.3}$$

$$y(z, v) = 0.0021(v - 240)(z - 10) \tag{6.4}$$

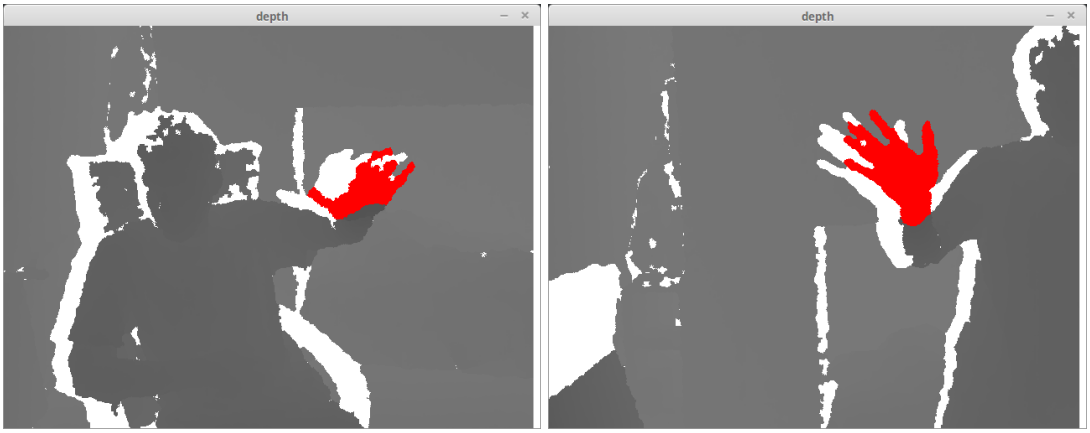
Kao i kod metode Qiana i suradnika [**?**], ali i mnogih drugih, i u ovom radu se pretpostavlja kako se ruka nalazi na najmanjoj dubini (niti jedan predmet u vidnom polju kamere nije bliži od ruke).

Za razliku od spomenute metode, u ostvarenoj metodi nije potrebno koristiti pomagala kao što je nereflektirajuća crna narukvica na zapešću. Izolacija dlana od okoline vrši se jednostavnim dubinskim filterom na empirijski određenoj dubini od $110mm$ u odnosu na globalni minimum dubine slike, što je ujedno i globalni minimum dijela slike na kojem se nalazi ruka. Iako najveći raspon prosječne ruke premašuje $110mm$, to ne predstavlja prevelik problem.

Na slikama 6.1 vidimo primjere ispravne izolacije dlana na dubinskoj slici. Radilo se o uspravnom dlanu (a), dlanu s dorzalne strane (b), pa čak dlanu u vodoravnoj ravnini (c) ili zatvorenom dlanu (d), ovakav jednostavni detektor je sposoban razlučiti



Slika 6.1: Uspješno izoliranje dlana na slici. Slika (a) (lijevo gore) predstavlja izoliranje uspravnog dlana, slika (b) (desno gore) izoliranje dorzalne strane dlana, slika (c) (lijevo dolje) izoliranje dlana u vodoravnoj ravnini, slika (d) (desno dolje) izoliranje šake

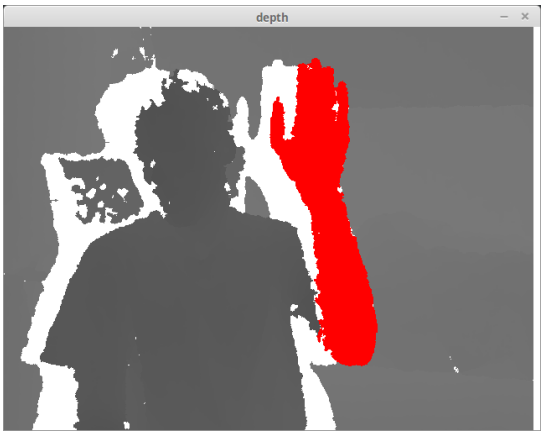


Slika 6.2: Prihvatljivo izoliranje dlana na slici. Slika (a) (lijevo gore) predstavlja izostavljanje karpalnog predjela dlana, slika (b) (desno gore) zahvaćena podlaktica

dlan kako bi što više procesnog vremena mogao prepustiti sljedećim fazama detekcije.

U nekim slučajevima detekcija nije ispravna, ali je u granicama otpornosti kasnijih

faza detektora. Tako se pri većim kutovima dlana u odnosu na projekcijsku ravninu mogu desiti dva slučaja (Slika 6.2). Ako je najbliža točka dlana na području zapešća, zahvaćen je i dio podlaktice, dok ako je najbliža točka na području prstiju, dlan djelomično izlazi iz granica dubinskog filtera.



Slika 6.3: Pogrešna izolacija dlana, zahvaćen i dio podlaktice

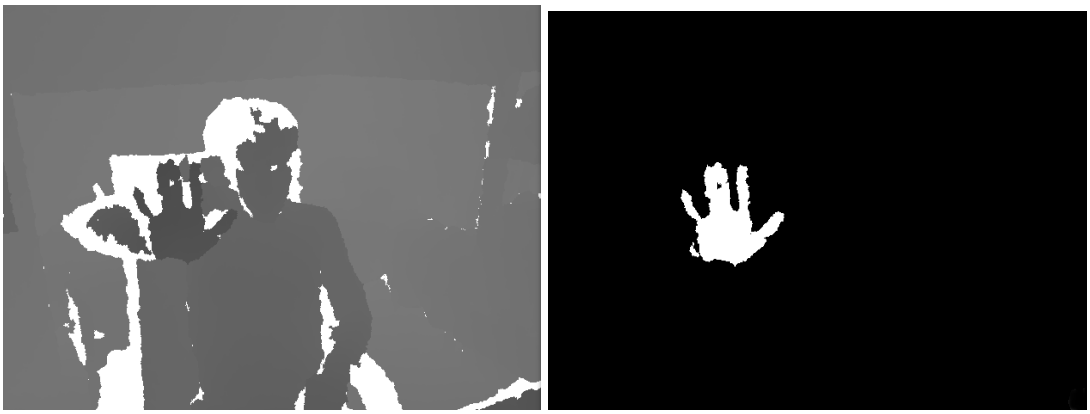
Oba slučaja većinom ne predstavljaju prevelik problem, no postoje i neke iznimke. Naime, ako je podlaktica u razini dlana, detektor neće biti sposoban razlučiti dlan od podlaktice (Slika 6.3). Takav slučaj predstavlja problem za detektor, ali takav slučaj ne predstavlja problem u korištenju detektora ako je osoba koja ga koristi svjesna takvog nedostatka.

Rezultat izolacije dlana uz sliku na kojoj se nalazi samo dlan je i (x, y, z) koordinata središta dlana u prostoru. Središte je dobiveno pronalaskom konture dlana te izlučivanjem njenog središta. Prije pronalaska konture na binarnoj slici se izvodi morfološka operacija zatvaranja s jezgrom veličine u radijusu od 2 susjedna slikovna elementa (Manhattan udaljenost). U slučaju da je središte konture izvan konture pa samim time i dlana, takva detekcija biva odbačena. Iako postoje položaji ruke gdje je to slučaj, takve položaje u većini slučajeva Kinect nije sposoban registrirati (Poglavlje **??**).

Jednom kada je dlan segmentiran od pozadine znatno se olakšavaju daljnje operacije. Za razliku od spomenutih naprednijih radova ovdje se neće razmatrati određivanje precizne lokacije ruke u vidu modela od više od 20 stupnjeva slobode. Naime, uzevši u obzir primjenu detektora za upravljanje robotskom rukom, prihvatljiva doza informacije sastoji se od binarne informacije o stupnju zatvorenosti dlana i radij vektora lokacije dlana te eventualno orijentacija dlana u smjeru x-osi definiranog koordinatnog sustava.

Neki od ključnih postupaka u detekciji izvedeni su pomoću biblioteke OpenCV:

- **Segmentacija slike na osnovu praga** je postupak prilikom kojeg se s obzirom na željenu vrijednost praga može konstruirati binarna slika (maska) gdje jedna vrijednost pokriva sve elemente čija je vrijednost ispod praga, a druga vrijednost sve elemente iznad praga. Na slici 6.4 (b) vidimo binarnu sliku dobivenu segmentacijom po dubini slike (a) pri čemu je prag određen kao $110mm$ u dubinu od minimalne vrijednosti dubine na slici.

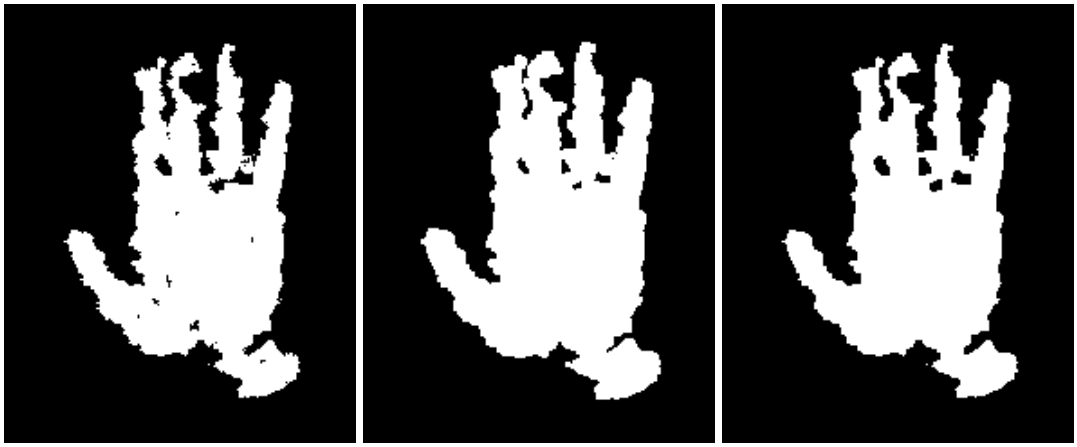


Slika 6.4: Segmentacija slike na osnovu praga, lijevo ulazna dubinska slika i desno generirana maska

- **Dilatacija i erozija binarne slike** su morfološke transformacije. Dilatacija je transformacija prilikom koje se generira nova slika iste veličine. Za svaki osvijetljeni slikovni element u početnoj slici se u ovoj slici osvjetljava element na toj lokaciji zajedno s n slikovnih elemenata u njegovoj okolini. Takvim postupkom smo povećali osvijetljene površine na slici. Erozija je slična dilataciji s razlikom što se postupak temelji na neosvijetljenim slikovnim elementima.

  Na slici 6.5 (c) je prikazana morfološka transformacija izvorne slike (a) operacijom zatvaranja (dilatacija pa erozija) uz djelovanje na okolnih 8 slikovnih elemenata. Kao što vidimo ovakav slijed transformacija pokriva moguće šumove unutar osvijetljenog dijela slike te može spojiti bliske skupine osvijetljenih slikovnih elemenata što je korisno ako na primjer šum *odsiječe* prst od ostatka ruke.

- **Određivanje obujmica** skupina osvijetljenih slikovnih elemenata koristan je način za aproksimaciju oblika na slici. Prvi korak u određivanju obujmica (engl. *convex hull*) je generiranje kontura oko skupina susjednih slikovnih elemenata. Jednom kada je kontura izgenerirana jednostavno se može naći obujmica kao minimalan konveksan poligon koji obuhvaća konturu.



Slika 6.5: Morfološka transformacija binarne slike operacijom zatvaranja, s lijeva na desno ulazna slika, međukorak te slika nakon transfomracije

Na slici 6.6 je običnoj binarnoj slici sivom bojom pridodana razlika u površini konture oblika i obujmice. Ta razlika se može iskoristiti kao dobar indikator zatvorenosti dlana.

Slika 6.6: Obujmica i kontura ruke na slici

6.2. Određivanje poze dlana

6.2.1. Slične metode

Različiti radovi predlažu različite metode određivanja poze u kojoj se nalazi ruka, ali i u različitom opsegu preciznosti.

Qian i suradnici u svom radu [**?**] su modelirali prosječnu ruku pomoću 48 sfera uz 26 stupnjeva slobode. Pretražuju ekstreme u području dlana te im oni služe kao kandidati za vrhove prstiju i zapešće. Pomoću inverzne kinematike nad modelom generiraju moguće položaje ruke te svaki od njih ocjenjuju s obzirom na sličnost s dubinskom slikom. Najbolji model je proglašen trenutnim položajem ruke.

Thompson i suradnici u svojem radu [**?**] koriste naučenu kaskadu konvolucijskih neuronskih mreža. Učenje je izvršeno na 70 000 označenih slika. Mreže na izlazu daju dvodimenzionalnu Gaussovu razdiobu vjerojatnosti da se na određenoj lokaciji nalazi jedna od 36 označenih lokacija na ruci (npr. vrh malog prsta). I u ovom radu inverznom kinematikom se traži model ruke koji najviše odgovara razdiobama. Pri tome je rabljen često korišten model ruke otvorenog koda *libhand* [**?**].

Valja napomenuti kako su navedene metode sposobne prepoznati ruku u jednoj dubinskoj slici te im ne treba niz slika.

6.2.2. Razvijena brza metoda određivanja zatvorenosti dlana

Zatvorenost dlana je apstraktan pojam te ga treba pobliže definirati. Prilično je intuitivan pojam otvoreni dlan koji predstavlja dlan u planarnom položaju, pri čemu kutevi između članaka teže nuli (Slika **??** a). Isto tako je jasan je pojam zatvorenog dlana (Slika **??** d). Teško je pak definirati stupanj zatvorenosti prilikom neke geste, pri kojoj dlan nije ni potpuno otvoren ni zatvoren, pogotovo ako je svaki prst pod svojim kutom. Tako se u ovom radu stupanj zatvorenosti određuje binarno pri čemu *0* predstavlja zatvoren dlan, a *1* sve ostale položaje.

Navedeni pristup sasvim je dovoljan pri kontroli željenog robota jer takav klasifikator nam određuje hoće li se prsti robotske ruke skupiti ili ne. Naime, iako robot posjeduje prste, nezgrapno bi bilo odrediti parametar zatvorenosti za svaki prst kako broj robotskih prstiju (3) ne korespondira broju ljudskih.

Klasifikator je napravljen jednostavno i empirijski. Izračuna se površina binarne slike zahvaćena konturom dlana ($P_k(\mathbf{s})$) i površina obujmice iste konture ($P_o(\mathbf{s})$). Generira se značajka koja predstavlja omjer površine konture i obujmice (6.5).

$$f_1(\mathbf{s}) = \frac{P_k(\mathbf{s})}{P_o(\mathbf{s})} \tag{6.5}$$

Potom se normalizira površina konture kako bi bila invarijantna na dubinu te se ista tretira kao druga značajka (6.6).

$$f_2(\mathbf{s}) = P_k{}'(\mathbf{s}) \tag{6.6}$$

Određivanje zatvorenosti dlana vrši se određivanjem predznaka linearne kombinacije dvije navedene značajke te još ostaje samo odrediti parametre k i l (6.7).

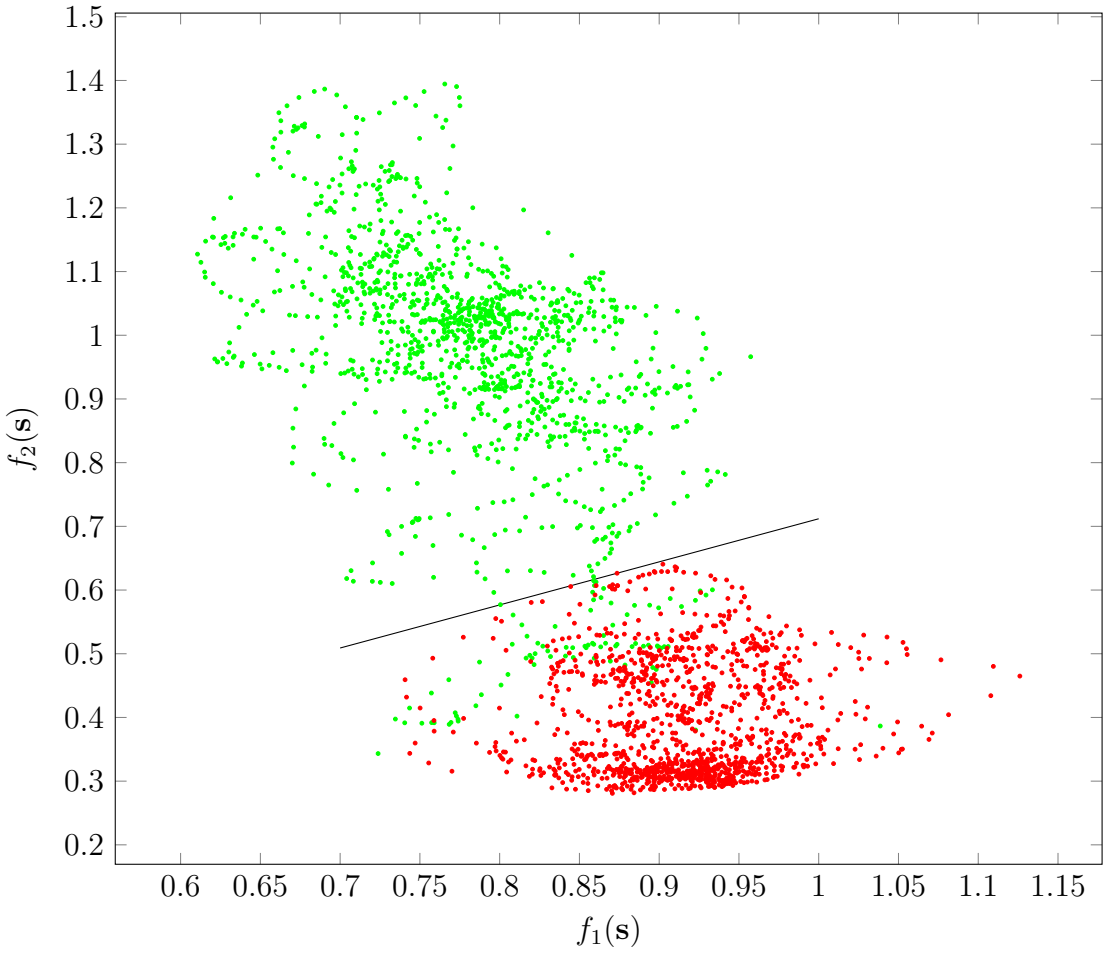
$$Z(\mathbf{s}) = \begin{cases} 1 & ako \quad kf_1(\mathbf{s}) + lf_2(\mathbf{s}) > 0 \\ 0 & ako \quad kf_1(\mathbf{s}) + lf_2(\mathbf{s}) \leq 0 \end{cases} \tag{6.7}$$

Učenje klasifikatora

Nakon što su definirane značajke, generirano je 2559 ispitnih primjera od kojih 1239 s otvorenim dlanom i 1320 sa zatvorenim dlanom. Grafikon **??** prikazuje točke koje predstavljaju slike u ravnini određenoj značajkama f_1 i f_2 s time da zatvoreni dlanovi su prikazani zelenom bojom, a otvoreni crvenom.

U ravnini značajki grafičkom metodom je aproksimiran pravac koji razdvaja dvije klase, zatvorene i otvorene dlanove s greškom od 3.4%. Težilo se preciznijoj detekciji zatvorenog dlana, što je uzrokovalo gubitak u preciznosti detekcije otvorenog dlana. Takav pristup je vođen pretpostavkom da prilikom upravljanja robotskom rukom zatvoreni dlan podrazumijeva kako ruka drži neki objekt te želimo biti sigurni kako ga neće olako ispustiti bez naše volje. Iako za neki teži problem ovakav pristup detekciji otvorenosti dlana ne bi bio dobar te bismo se trebali okrenuti naprednijim metodama, u ovom slučaju rezultat je sasvim zadovoljavajuć s obzirom na poprilično jasnu distinkciju između dviju klasa.

Dobiveni pravac ima parametre $k = 0.67635$ i $l = 0.035558$ koji se koriste u detektoru prilikom određivanja zatvorenosti dlana.



Grafikon 6.7: Skup za učenje (crveno - slike zatvorenih dlanova, zeleno - slike otvorenih dlanova)

6.2.3. Razvijena metoda primjenom dubokih konvolucijskih neuronskih mreža

Inicijalna estimacija lokacije dlana u slici ne traje dugo te daljnja procjena poze dlana može trajati puno duže. Već i gore navedena brza metoda određivanja zatvorenosti dlana i na najslabijim uređajima ostavlja poprilično procesorsko vrijeme neiskorišteno. Odlučeno je dakle iskoristiti do tada neiskorišteno procesorsko vrijeme kako bi detektor dao bolji opis poze u kojoj se dlan nalazi.

Prethodna metoda se temelji na jednostavnim, ali vrlo uspješnim heuristikama. Ipak, za opisivanje poze dlana s više parametara takav pristup je puno zahtjevniji. Jedan od alternativnih skupova pristupa su tzv. pristupi vođeni samim podacima (engl. *data-driven approach*). Za takav pristup valja prikupiti dovoljnu količinu željenih (u

konkretnom slučaju i označenih) podataka te nad njima naučiti model. Model bi opisivao pozu dlana koji mu je predan kao slika na ulazu.

U nedostatku bolje opreme (konkretno žiroskopa kakav se nalazi u većini mobilnih uređaja) koja bi omogućila brže i preciznije prikupljanje željenih podataka s detaljnijim opisom poze dlana, valjalo je osmisliti efikasan način prikupljanja što većeg broja raznovrsnih podataka uz njihovo precizno označavanje.

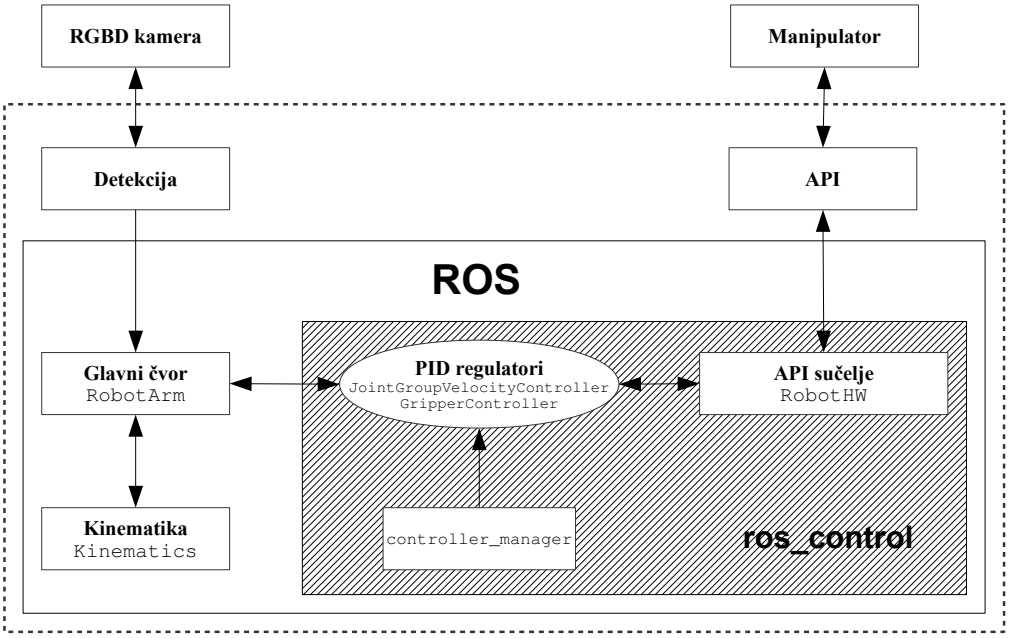
Klasifikatora poze ima dva izlaza. Prvi određuje stupanj otvorenosti dlana kao i u prethodnoj metodi, dok drugi određuje rotaciju dlana oko x osi. Takva rotacija omogućuje robotskoj ruci da uhvati nešto iznad, ispred ili ispod dlana te zbog toga je odabrana kao najkorisnija u usporedbi s rotacijama oko y ili z osi. Podaci su prikupljani u 15 različitih klasa kao kartezijev produkt tri stupnja zatvorenosti dlana$0, 0.5, 1$ te 5 različitih rotacija oko x osi $-90\,\mathrm{deg}, -45\,\mathrm{deg}, 0\,\mathrm{deg}, 45\,\mathrm{deg}, 90\,\mathrm{deg}$. Kada bismo u istu proceduru uključili i druge dvije rotacije, tada bi ukupan broj klasa bio $3{*}5{*}5{*}5 = 375$ što uz dane resurse ne bi bilo izvedivo.

Ukupno je prikupljeno 61395 označenih slika dlana. Prikupljene slike su dubinske slike dlana dobivenog detekcijom segmentirane na osnovu praga opisanog u postupku detekcije te binarizirane tako da slikovni elementi dlana imaju vrijednost 1, a ostali 0.

Razlog binarizaciji i segmentaciji leži u tome što pri učenju modela želimo reducirati količinu informacija koje klasifikator dobiva na ulaz u postupku učenja kako ih ne bi prenaučio. Naime, uklanjanjem pozadine klasifikator neće biti opterećen time što je u pozadini dlana već morati naučiti heuristike isključivo temeljem slikovnih elemenata dlana. Također, iz sličnog razloga ne želimo klasifikatoru pružiti informaciju na kojoj udaljenosti se dlan nalazio u datoj slici kako se ne bi oslonio i na tu informaciju.

7. Upravljački algoritam

Ovo poglavlje pomoću koncepata i metoda objašnjenih u prethodnim poglavljima možemo detaljno prikazuje arhitekturu upravljačkog dijela sustava. U odlomku 7.1 opisujemo korake potrebne za inicijalizaciju sustava te detalje sklopovskih sučelja, glavnog čvora i kinematičkih izračuna. Odlomak 7.2 sadrži opis simulacijskih ispitaivanja odrađenih na sustavu i njihove rezultate. Rezultate zatim komentiramo i objašnjavamo njihov odraz na stvarni sustav.



Slika 7.1: Upravljačka shema

Na slici 7.1 prikazana je detaljna shema sustava u programskoj izvedbi. Iscrtkani pravokutnik predstavlja granicu između uređaja i računala dok žuti i sivi predstavljaju područje sustava unutar ROS, odnosno `ros_control`arhitekture, respektivno.

7.1. Arhitektura upravljačke petlje

Ulazni signal u sustav dolazi iz detekcijskog algoritma u obliku ROS poruke `ime poruke` čija je struktura : Kako bi ostvarili mogućnost udaljenog upravljanja, detekcijski dio ostvarujemo kao zasebni ROS čvor koji frekvencijom 30 Hz objavljuje poruke na temu `ime topica`. Ovime omogućujemo pokretanje detekcijskog algoritma na zasebnom računalu, uz uvjet da je na istoj mreži kao ostatak ROS čvorova.

Glavni čvor svaku novu poruku `ime topica` čita i obrađuje. Početnoj poziciji korisnikovog dlana (pri prvoj detekciji) pridružuje se željena pozicija manipulatora jednaka početnoj. Pomak dlana od početne pozicije primljen u poruci sustav zbraja s početnom pozicijom manipulatora koju smo ranije spremili, čime dobivamo novu željenu poziciju manipulatora. Trenutna pozicija manipulatora potom je oduzeta od željene, čime dobivamo nama bitnu potrebnu translaciju manipulatora. Zbog ograničenja pri detekciji orijentacije dlana, trenutnu orijentaciju (klasificiranu u diskretne slučajeve) direktno prenosimo u željenu orijentaciju manipulatora.

Algorithm 1 Računanje potrebne promjene vektora konfiguracije

Ulaz: P – odmak dlana od početne pozicije
R – diskretna orijentacija
M – početni vektor konfiguracije manipulatora
T – trenutni vektor konfiguracije manipulatora.
Izlaz: E – potrebna promjena vektora konfiguracije.
for $(i := 0; i < 6; i++)$ **do**
  if $(i < 3)$ **then**
    $nova_i := M_i + P_i$
  end if
  if $(i >= 3)$ **then**
    $nova_i := M_i + R_i$
  end if
  $E_i = nova_i - T_i$
end for

Vektor E_i predstavlja *vektor razlike konfiguracija* i koristi se u daljnjim koracima algoritma, koji se ovisno o dostupnim naredbama konfigurira u dva različita oblika. Prvi oblik se koristi pri dostupnoj naredbi Kartezijske brzine izvršnog člana, kojom možemo direktno narediti kretanje po x,y i z osima koordinatnog sustava baze. Drugi oblik koristi se ako je dostupno postavljanje brzine zglobova, što ga čini složenijim.

7.1.1. Inicijalizacija sustava

Prvi korak pri inicijalizaciji sustava sastoji se od pokretanja sklopovskih sučelja. Samo sučelje jest programska rutina koji ostvarujemo tako da API funkcije manipulatora "umotavamo" u `hardware_interface` klasu koja je dio `ros_control`paketa. Pri pokretanju cijelokupnog sustava , sučelje se pokreće prvo i čini granicu između algoritama i naredbi koje direktno pokreću manipulator.

Sučelje brzine izvršnog člana

U slučaju dostupnosti sklopovskog sučelja koje dopušta direktno postavljanje kartezijske brzine, algoritam se znatno pojednostavljuje. Na ovaj način izbjegavamo složen kinematički račun iz poglavlja 5.1 i oslanjamo se na interne algoritme samog manipulatora.

Pri primanju poruke tipa `ime_poruke` sa detekcijskog servera, funkcija `sendCartesianCommand` unutar glavnog čvora obrađuje podatke. Vektor razlike konfiguracija sklairamo za neki iznos α, ali daljnju regulaciju ostavljamo upravljačkom algoritmu samog manipulatora. Ovo znači da `ros_control` u ovom obliku koristimo isključivo za prosljeđivanje naredbi te čitanje stanja zglobova. Po završetku obrade, Kartezijska naredba šalje se u obliku poruke `ime_2` na temu `goal`, koja je rezervirana za referentnu veličinu kretanja manipulatora. Naredba vezana uz izvršni član sprema se u poruku `ime_3` te se šalje na temu `goal_2`, koja je rezervirana za referentnu veličinu operacije izvršnog člana te ima vlastiti regulator.

Pokazalo se da je ovaj pristup optimalan za brzi početak korištenja jer visoka radna frekvencija najčešće znači da se parametar α može postaviti proizvoljno, dok ostalu regulaciju provodi API.

Sučelje brzine/pozicije zglobova

Nakon inicijalizacije, sučelje registrira pojedinačne zglobove i "spaja" se na naredbe pisanja i čitanja stanja zglobova. Ovo spajanje vrši se tako da se spomenuti API naredbe umataju u funkcije čiji su pokazivači dostupni odgovarajućim `ros_control`regulatorima. Važno je spomenuti da svaki regulator ima odgovarajuću podvrstu `hardware_interface` klase i neće se spajati na neodgovarajuća sučelja.

Sljedeći korak je inicijalizacija sučelja, pokreće se regulator tipa `velocity_controllers/JointGroupVelocityController`. Regulator svoje parame-

tre učitava iz odgovarajuće konfiguracijske YAML[1] datoteke. U parametre između ostalog ubrajamo imena robota i zglobova, tip sučelja, pojačanja P, I i D komponenti, vremenska ograničenja na izvršenje naredbi, ograničenja zglobova i frekvenciju. Pokretanjem regulatora API naredbe manipulatora činimo mrežnim resursima unutar ROS "grafa", čime ovaj dio sustava postaje potpuno fizički odvojiv od ostatka. Svi podaci sa senzora postaju dostupni na temi `joint_states`, postavljanje referentne veličine vrši se slanjem odgovarajuće poruke na temu `\goal`, a parametri regulatora također su dostupni.

7.1.2. Glavni čvor

Inicijalizacijom glavnog čvora dovršavamo pokretanje sustava. Dok `ros_control` regulator osigurava stabilnu brzinu, glavni čvor čini nadređenu regulacijsku petlju koja osigurava odgovarajuću referentnu veličinu baziranu na razlici željenog i trenutnog vektora konfiguracija. Kako bi rješili problem kinematike opisan u poglavlju 5.1, pomoću metoda dostupnih u KDL bibliotekama učitavamo opis kinematičkog lanca iz `.urdf` datoteke. Iz učitanog se lanca pomoću funkcija u klasi `Kinematics` generira matrica direktne kinematike i Jakobijan sustava. Nakon toga poziva se funkcija koja obavlja množenje pseudoinverza jakobijana traženom promjenom vektora konfiguracija način sličan 5.22, pri tome dodajući članove koji osiguravaju numeričku stabilnost. Konačno, inicijaliziraju se odgovarajući ros pretplatnici i izdavači te sustav postaje spreman za uporabu.

Sada pri primanju nove naredbe iz kinecta sustav obrađuje ulazne podatke kao što je opisano na početku poglavlja. Dobiveni vektor razlike konfiguracije se pomoću funkcija `Kinematics` klase pretvara u vektor pogreške pozicije zglobova. Pogreške pozicija zglobova tretiraju se kao željene brzine zglobova, ali ih je prvo potrebno skalirati s obzirom na maksimalnu dozvoljenu brzinu. Ograničenja brzina upisana su u YAML datoteku koja se učitava pri pokretanju regulatora te ih `ros_control` stavlja na mrežne lokacije koje glavni čvor čita i sprema u varijable. Brzine je vrlo bitno skalirati na način kojim se zadržava njihov omjer, stoga prvo pronalazimo brzinu koja najviše prelazi ograničenje te sve brzine skaliramo prema njoj.

[1] YAML (engl. YAML Ain't Markup Language) - `www.yaml.org`

Algorithm 2 Skaliranje brzina

Ulaz: V – brzine bez ograničenja, O – ograničenja brzine.

Izlaz: Vs – skalirane brzine.

$m := 0$

for $(i := 0; i < brojzglobova; i++)$ **do**

  if $|V_i| > O_i \;\&\; \|V_i - O_i\| > m)$ **then**

    $m := \|V_i - O_i\|$

    $i_m := i$

  end if

end for

$s = O_{i_m}/V_{i_m}$

for $(i := 0; i < brojzglobova; i++)$ **do**

  $Vs_i := V_i * s$

end for

Visoka vrijednost akceleracije zglobova može uzrokovati oscilacije u kretanju pri određenim položajima. Akceleracije ograničavamo skaliranjem, na sličan način kao kod brzina. Pri ograničavanju potrebno je voditi računa o frekvenciji glavnog čvora jer se brzine osvježavaju 50 puta u sekundi te se male razlike pri svakom taktu mogu pretvoriti u veliku razliku.

Algorithm 3 Skaliranje brzina s obzirom na akceleraciju

Ulaz: V – brzine bez ograničenja, Vt – trenutna brzina, A – ograničenja akceleracije.

Izlaz: Vs – skalirane brzine.

$m := 0$

for $(i := 0; i < brojzglobova; i++)$ **do**

  if $\|V_i - Vt_i\| > A_i \;\&\; \|V_i - Vt_i\| > m)$ **then**

    $m := \|V_i - Vt_i\|$

    $i_m = i$

  end if

end for

$s = A_{i_m}/m$

for $(i := 0; i < brojzglobova; i++)$ **do**

  $Vs_i := V_i * s$

end for

Nakon ovih skaliranja referentna brzina spremna je za slanje na temu koju čita

regulator brzine zglobova. U slučaju sučelja pozicije zglobova, ovu veličinu možemo smatrati pomakom te je dodati trenutnim vrijednostima zakreta zglobova kako bi dobili željenu poziciju. Vrijednosti vezane uz izvršni član neovisne su o sučelju i šalju se direktno.

7.1.3. Kinematika

Biblioteka KDL nije jedina, a vjerojatno ni najkvalitetnija dostupna metoda rješavanja općeg kinematičkog problema. Kako bi ostavili prostora za kvalitetnijeg nasljednika ili moguće vlastite implementacije naprednijih metoda, kinematiku smo odvojili u diskretnu komponentu sustava. Kao što je ranije spomenuto, sve funkcije vezane uz kinematičke kalkulacije sadržane su u `Kinematics` klasi.

Pri inicijalizaciji kalse unutar glavnog čvora prvo je potrebno konstruirati kinematički lanac od semantičkog opisa sadržanog u URDF datoteci. Čitanje se obavlja tako da pri pokretanju sustava datoteku učitavamo u ROS parametar `robot_description`, koji klasa pri inicijalizaciji dohvaća i sprema kao varijablu tipa `string`. Ovaj opis zatim parsiramo pomoću KDL funkcije `treeFromString` koja od opisa generira kinematičko stablo tipa `Tree`. Kinematički lanac koji opisuje manipulator iz stabla dohvaćamo pomoću funkcije `getChain` i spremamo u varijablu tipa `Chain`.

Pomću kinematičkog lanca inicijaliziramo kalsu `ChainFkSolverPos_recursive`, koja iz zakreta zglobova pomoću funkcije `JntToCart` daje poziciju izvršnog člana u koordinatnom sustavu baze. Za kalkulacije potrebnih brzina zglobova inicijaliziramo klasu `ChainIkSolverVel_wdls`. Pri pozivu funkcije `CartToJnt` generira Jakobijan te računa izraz sličan 5.22, uz određena poopćenja i osiguravanje numeričke stabilnosti (ref). Funkcija `getFK` sadrži proceduru za rješavanje direktne kinematike, dok `getIKvel` sadrži računa vezan uz dobivanje potrebnih pomaka zglobova.

Poznavajući teorijsku pozadinu iza ovakvog sustava, vlastita implementacija algoritma također je moguća. Unatoč tome, za KDL biblioteke odlučili smo se radi već rješenih tehničkih problema kao što su upravljanje memorijom i optimiranje performansi. Sustav mora vršiti izračune relativno visokom frekvencijom i stoga bilo kakvo *curenje* memorije može ostaviti trag na performansama.

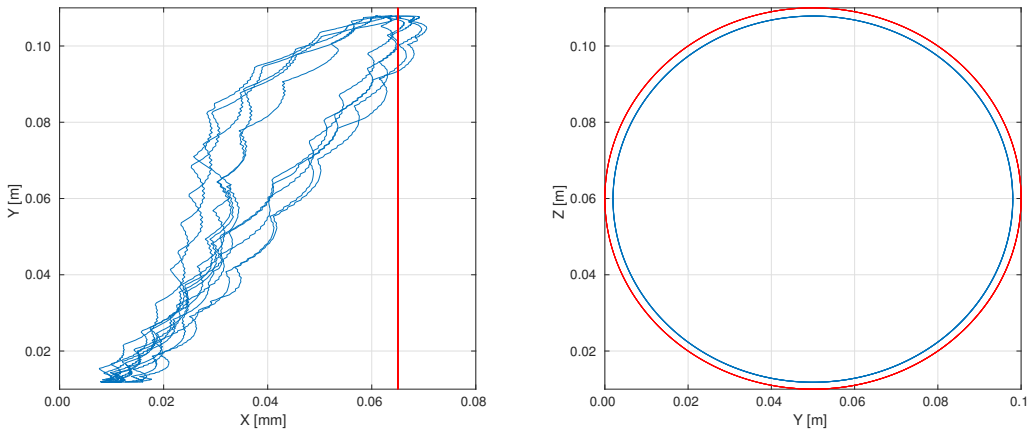
7.2. Simuliranje sustava

Ispravnost sustava bilo je potrebno testirati na simulaciji kako pri stvarnoj primjeni ne bi došlo do oštećenja korištenog manipulatora. Za postizanje simulacije manipulatora koristili smo Gazebo programski paket.

Kao što je prije spomenuto, Gazebo ima ugrađenu kompatibilnost sa `ros_control` paketom te nudi opciju stvaranja "lažnog" sklopovskog sučelja za simulaciju. Velika prednost Gazeba leži upravo u navedenoj kompatibilnosti jer pri simuliranju možemo koristiti identičan kod kao i za stvarni manipulator. Ovakvim pristupom znatno smanjujemo vrijeme potrebno za testiranje jer uočene probleme ne moramo popravljati na dvije različite varijante koda.

Simuliramo Jaco robotsku ruku, čija URDF datoteka sadrži i definirana sučelja zglobova te 3d modele koje korespondiraju pojedinim člancima. Ovakve opise možemo pronaći za većinu komercijalnih manipulatora, stoga je testiranje na isti način moguće vršiti i na drugim modelima. U simulaciju također uključujemo dinamičko ponašanje ruke jer opisna datoteka također sadrži tenzore inercije pojedinih članaka. Pri pokretanju, Gazebo automatski generira sklopovsko sučelje prema podacima iz opisa i inicijalizira regulator. Za inicijalizaciju sustava koristimo iste korake kao i prije, pri tome isključujući korak inicijalizacije sklopovskog sučelja za stvarni manipulator.

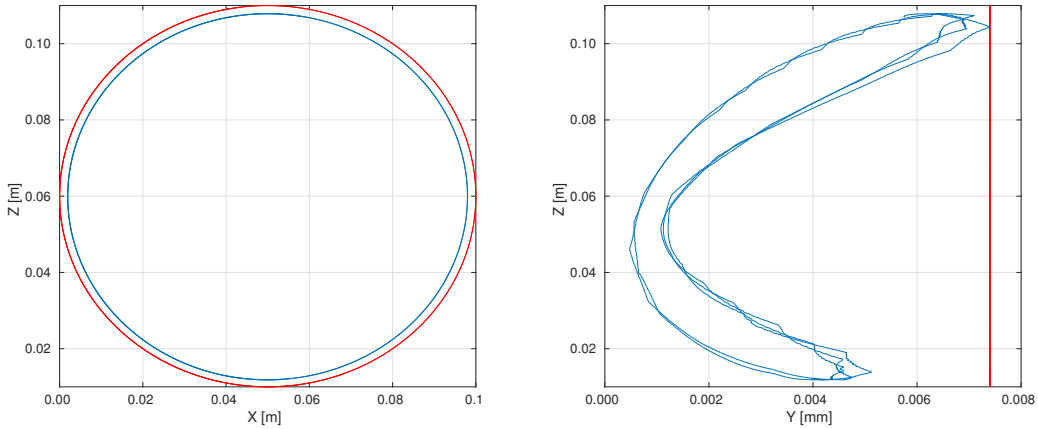
Testiranje provodimo zadajući kružne, isključivo translacijske putanje koje manipulator treba periodički izvoditi. Prva putanja prostire se samo u yz-ravnini te bi se u idealnom slučaju trebalo događati samo dvodimenzionalno kretanje. Na slici 7.2



Slika 7.2: Prikaz zadane putanje u yz-ravnini (crveno) i njene izvedbe (plavo). **Lijevo**: prikaz u xz-ravnini, **Desno**: prikaz u yz-ravnini.

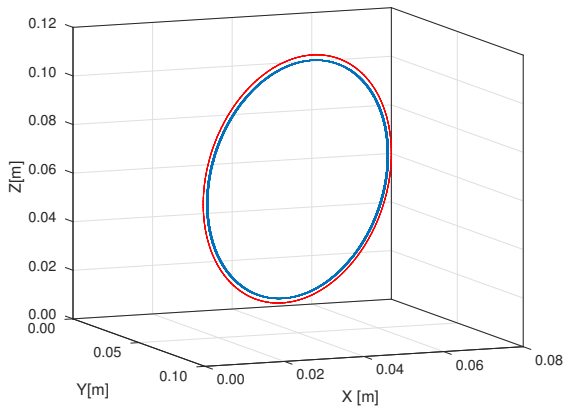
primjećujemo kako za planarno gibanje model ima naočigled zadovoljavajuće prefor-

manse. Prikaz u xz-ravnini otkriva kako ostvarujemo skoro savršeno kružnu putanju. Greška u stacionarnom stanju uzrokovana je relativno malim promjerom kruga čija zakrivljenost zahtjeva veliku brzinu sustizanja. Na prikazu gibanja u yz-ravnini primjećujemo odstupanja reda veličine manjeg od milimetra koja ne predstavljaju problem jer industrijski precizno kretanje nije ciljana karakteristika našeg sustava. Postavljamo



Slika 7.3: Prikaz zadane putanje u xz-ravnini (crveno) i njene izvedbe (plavo). **Lijevo**: prikaz u xz-ravnini, **Desno**: prikaz u yz-ravnini.

krug duž xz-ravnine i zabilježavamo rezultate na slici 7.3. Rezultati su slični kao i kod slučaja sa slike 7.2, te su pogreške u praćenju zanemarive kada uzmemo u obzir zamišljene aplikacije sustava. U praksi zadana putanja nikada neće biti planarna, stoga zadajemo putanju u obliku kružnice postavljene dijagonalno u prostoru.

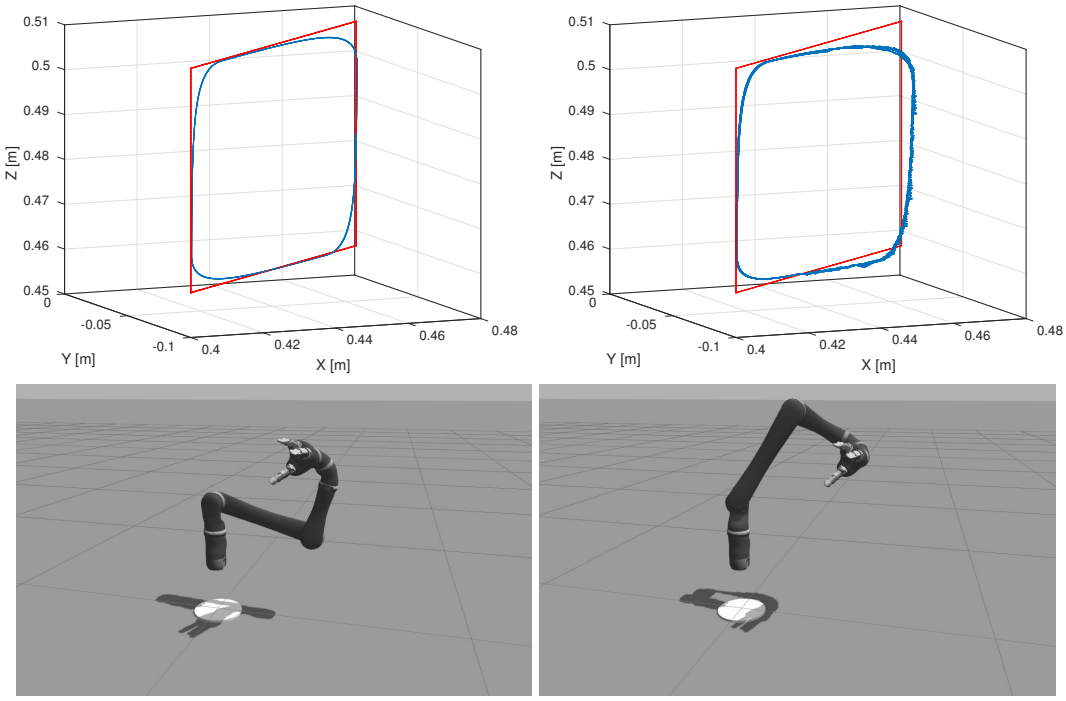


Slika 7.4: Prikaz zadane prostorne putanje (crveno) i njene izvedbe (plavo) u Kartezijevom koordinatnom sustavu.

Putanja sa slike 7.4 zahtjeva kretanje po sve 3 osi i predstavlja izazov za stabilnost algoritma. Iz rezultata možemo zaključiti kako je u općem slučaju pri translacijskom

gibanju po neprekinutoj krivulji kinematičko rješenje zadovoljavajuće. Potrebno je uzeti u obzir da je kružna putanja najpogodnija za izvedbu manipulatorima, dok će u praksi putanje često sadržavati nagle promjene smjera.

Pravokutne ili isprekidane putanje mogu predstavljati izazov jer savršeno pravocrtno gibanje manipulatora u praksi nije moguće. U većini slučajeva linearne se putanje uspješno izvode sa određenom pogreškom pri promjeni nagloj promjeni smjera. Na slici 7.5 lijevo primjećujemo kako je pravokutno gibanje ostvareno sa pogreškom pri skretanju reda veličine milimetra. Ovo je očekivano višestruko veća pogreška od one dobivene gibanjem po kružnici, ali i dalje potpuno prihvatljiva s obzirom na primjenu.



Slika 7.5: Prikaz izvedbe iste putanje sa dvije različite početne konfiguracije. Gornji red sadrži željenu (crveno) i izvedenu (plavo) putanju . Donji red sadrži prikaz početne konfiguracije putanje iznad.

Na slici 7.5 desno vidimo izvođenje iste putanje pri drugačijoj početnoj konfiguraciji manipulatora. Rezultat je očito drastično lošiji jer se na dijelu putanje pojavljuju značajne oscilacije koje uzrokuju oscilatorno gibanje i pogrešku u praćenju. Ovo je posljedica činjenice da manipulatori imaju tzv. singularne konfiguracije u kojima Jacobijeva matrica manipulatora nije regularna te se efektivno gubi stupanj slobode. U ovakvim konfiguracijama male kartezijske promjene zahtjevaju neizvedivo brze rotacije zglobova, čime sustav postaje nestabilan.

Metoda provođenja kinematičkog računa unutar KDL biblioteka "ublažava" singularitete tako da spriječava ulazak vrijednosti u beskonačnost te osigurava kontinuiran rad sustava. Pronalazak svih singularnih konfiguracija zahtjeva pronalaženje svih zakreta zglobova koji zadovoljavaju identitet 7.1:

$$\det \mathbf{J} = \mathbf{0} \tag{7.1}$$

Iako pronalaženje svih jedinstvenih rješenja ovog identiteta programski nije pretjerano složeno, izbjegavanje ovakvih konfiguracija složeniji je problem. Potrebno je izvesti algoritam zaobilaženja kritičnih konfiguracije (u općem slučaju) na način da se ne gubi smjer kretanja. Ovo zahtjeva dublje ulaženje u problematiku i veliku količinu testiranja te za sada moguća oscilatorna stanja prihvaćamo kao nedostatak sustava.

8. Rezultati

9. Rasprava

10. Zaključak