SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 3962

Teleoperacija robotske ruke Kinova Jaco 3D kamerom

Filip Marić

Zagreb, ožujak 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 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.

Korisnost ovakvog sustava pronalazimo u širokom spektru područja, od vojne do zabavne industrije. U industriji zabave jedna moguća primjena je pomicanje kamere na kraju više-osnog manipulatora, gdje ovakvo upravljanje omogućuje intuitivniju kontrolu kadra od strane snimatelja. Ljudima s poteškoćama u kretanju robotska ruka upravljana ovakvom metodom može olakšati dohvaćanje objekata u okolini. Ovakvim upravljanjem dobivamo prirodnije kretanje robotskih manipulatora, što će postupnim ulaskom robota u svakodnevni život postati tražena karakteristika.

1.1. 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.
- Open source izvedba 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 trebalo uklopiti u strukturu upravljačke petlje visoke razine, pri tome vodeći računa o radnim frekvencijama korištenih sklopova.

[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

2. Izvedba

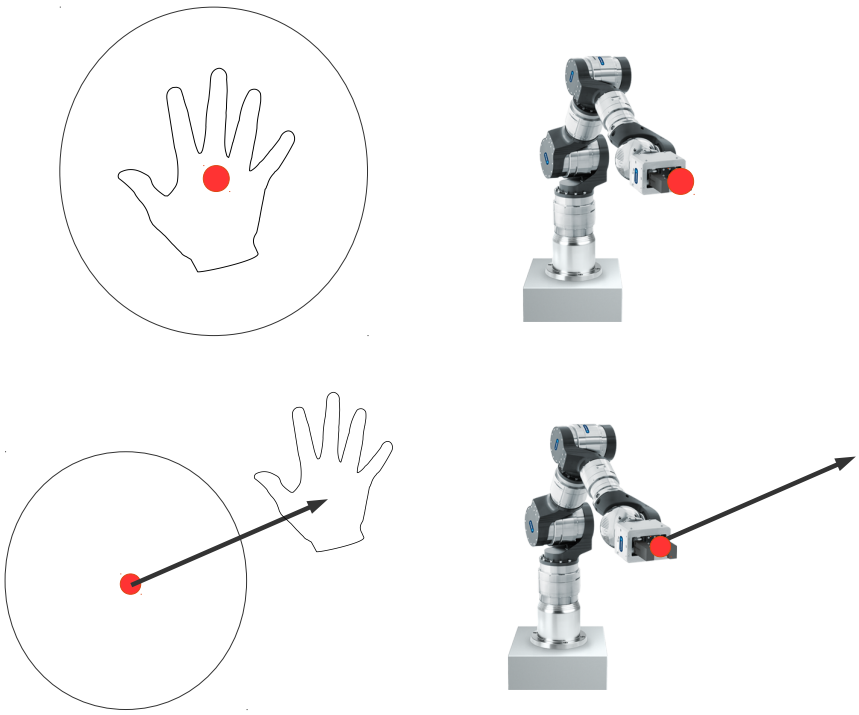
U ovom poglavlju opisujemo zamišljeno upravljačko sučelje. Prvo potpoglavlje opisuje dva načina na koje upravljački algoritam bilježi kretnje ruke korisnika. Drugo potpoglavlje sadrži pregled arhitekture cjelokupnog sustava i svrhe pojedinih komponenata.

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 svara upravljački signal kojim nastoji izjednačiti stvarnu i željenu poziciju manipulatora.

Druga izvedba oko početne pozicije korisnikove ruke formira kuglu. Zadržavanjem dlana korisnika unutar kugle, upravljački algoritam zadržava vrh manipulatora u trenutnoj poziciji. Pomicanjem dlana korisnika izvan kugle, upravljački algoritam pomiče vrh manipulatora u smjeru vektora udaljenosti dlana od središta kugle. Zajednička funkcija kod oba pristupa vezana je uz otvaranje i zatvaranje alata. 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 nije komplicirano radi strukture podataka koji ulaze u upravljački algoritam.



Slika 2.1: Prikaz *joystick* načina upravljanja.

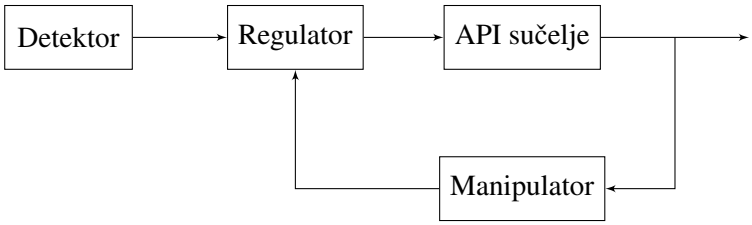
2.2. Struktura sustava upravljanja

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. Nakon toga se rezultat pretvara u potrban zakret (ili brzinu zakreta) zglobova manipulatora, nakon čega se šalje u API sučelje.
- **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.



Slika 2.2: Blok dijagram izvedbe sustava

2.2.1. 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. Minimalni zahtjev na sam detekcijski algoritam jest mogućnost pronalaženja koordinata ljudskog dlana u barem dvije dimenzije, ovime se omogućava kretanje manipulatora po ravnini. U našem radu izvedeno je trodimenzionalno translacijsko kretanje, dok je sustav potpuno osposobljen za detekciju punih 6 stupnjeva slobode gibanja.

2.2.2. 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 detaljnije 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 omogućuje pokretanje komponenti pojedinačno na različitim fizičkim računalima, što se pokazalo korisnim.

2.2.3. 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 "zamotati" u ponuđene klase, te ostatak sustava izvesti koristeći njih.

2.2.4. 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

3.1. Microsoft Kinect 3d kamera

3.2. Kinova Jaco robotska ruka

Kinova Jaco (3.1) 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 3.1: 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.

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 manipulatora, diferenciranjem zakreta dobivamo informaciju o brzini okretanja. Informacijom

Slika 3.2: 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

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 poz-

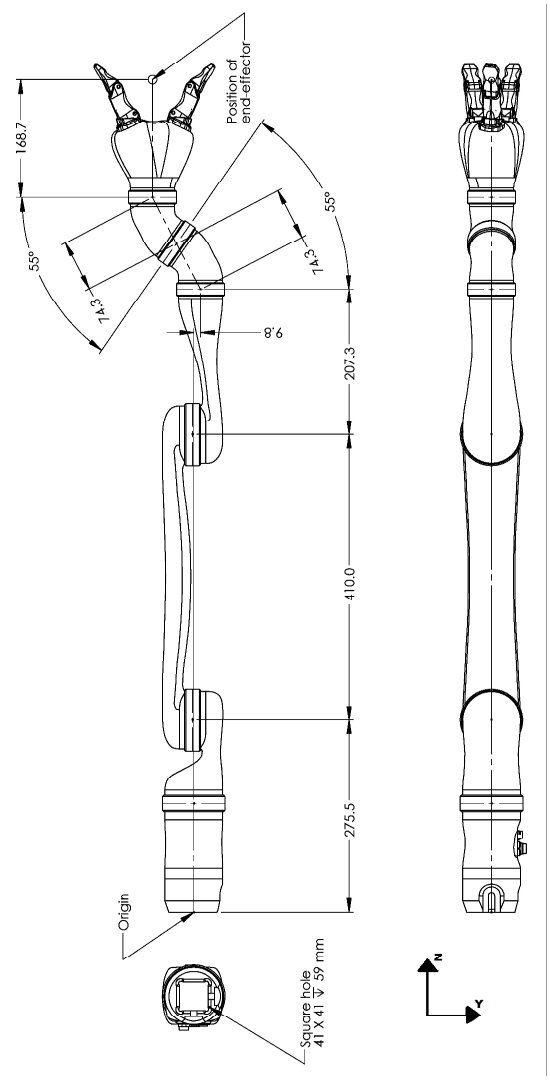
Maksimalna brzina	600 [RPM]
Ulazni napon	$18 - 29$ [VDC]
Nominalni moment	15 [mNm]
Maksimalni moment	30 [mNm]

Tablica 3.4: Prikaz tehničkih specifikacija aktuatora prstiju

natog 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.



Slika 3.3: Shema Kinova Jaco robotske ruke sa označenim duljinama i zakretima elemenata

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.

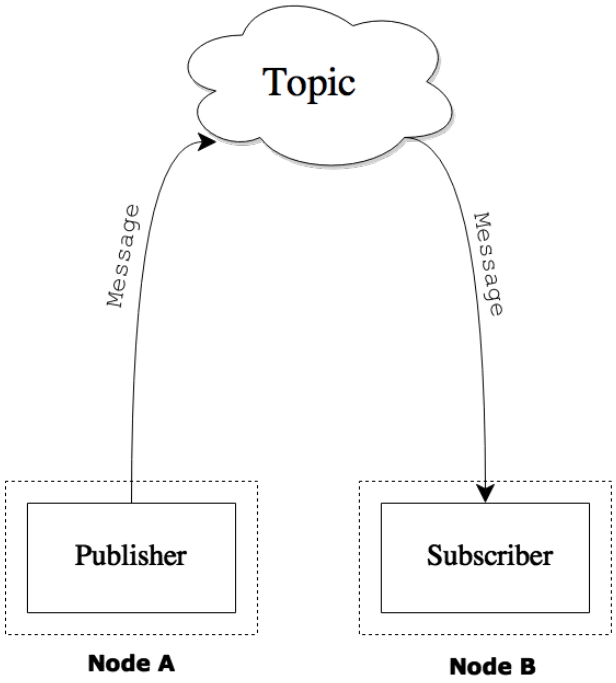
Glavna gradivna jedinica svake izvedbe u ROS-u je paket. Paketi u sebi sadrže C++/Python aplikacije koje nazivamo čvorovima (engl. *nodes*). Čvorove koristimo za komunikaciju s hardwareom (aktuatorima , senzorima ...), upravljanje simulacijama i prikazivanje podataka. Izvedbe se u stvarnosti najčešće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

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

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 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: Node A i Node 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. Objavljivač (en. *publisher*) je objekt koji unutar čvora šalje poruku na temu, dok pretplatnik (en. *subscriber*) čini njegov komplement i čita poruku s teme.

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.

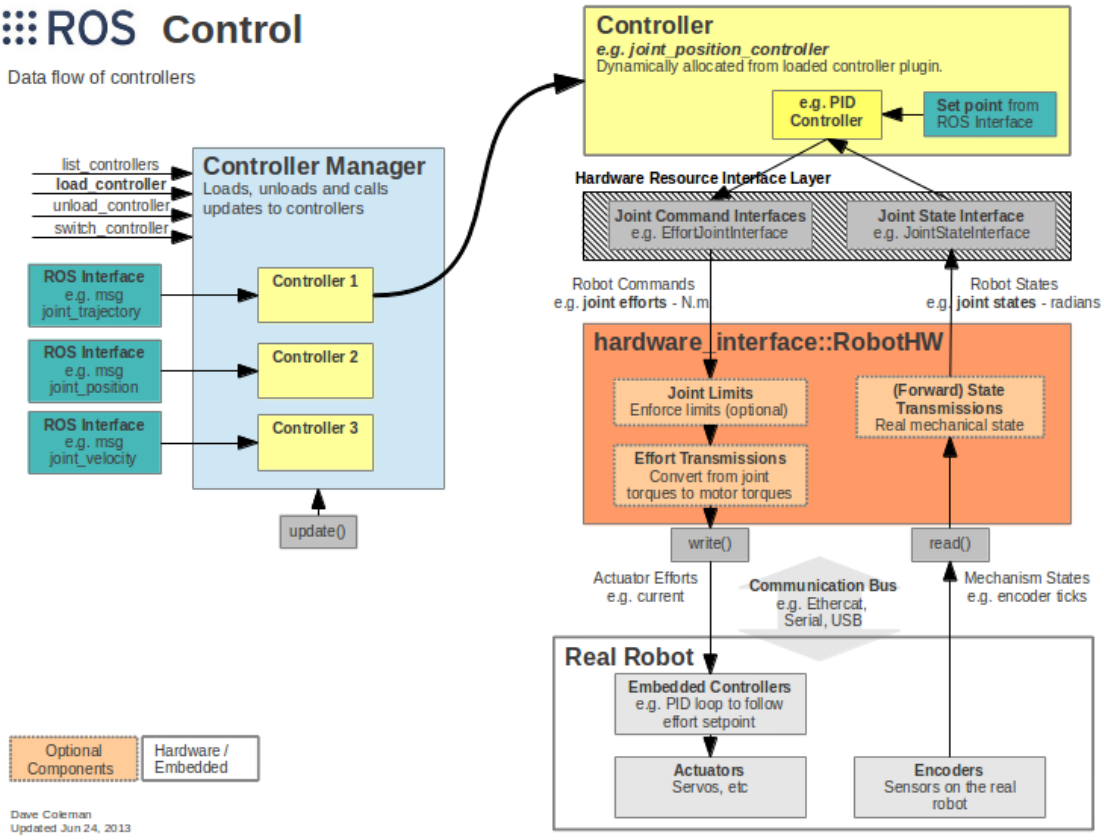
ROS nudi mogućnost učitavanja parametara s mrežnih lokacija čime se omogućuje jednostavno fino podešavanje ponašanja čvora pri testiranju. Ovakvim pristupom se izbjegava potreba ponovnim kompiliranje koda pri svakoj promjeni parametara.

4.1.3. ros control

ROS paket `ros_control` ostvaruje generičke PID regulatore 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 funkcija koje čine ROS-API sučelje. Umjesto da regulatori šalju i čitaju podatke koristeći API funkcije direktno, ove operacije obavljaju se na generičkim funkcijama unutar kojih su "umotane" API funkcije . Ovime korisnik kako bi primjenio postojeći sustav na drugog robota

mora samo uklopiti odgovarajuće API funkcije unutar generičkih. Za dobar dio robota ovo sučelje već postoji radi open-source karaktera ROS ekosustava.



Slika 4.3: Mathworks Matlab logo

4.2. Matlab

Slika 4.4: 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 robotske ruke.

4.3. OpenCV

4.4. KDL

KDL (engl. *Kinematics Dynamics Library*) je open-source C++ biblioteka razvijena za modeliranje i računanje kinematičkih lanaca. Pomoću ove biblioteke moguće je programski rješavati direktnu i inverznu kinematiku te dinamiku manipulatora sa do 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.

4.5. Gazebo

Slika 4.5: Gazebo logo

Gazebo je open-source programski paket za simulaciju robotskih sustava, njihovih senzora i okoline. Kako bi se postiglo simuliranje dinamike robota , moguće je izabrati jedan od nekoliko modula koje Gazebo podržava. Gazebo također omogućuje kvalitetno iscrtavanje 3d modela korištenog robota pomoću 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

aplikacijama. Gotovo svaki komercijalni robotski manipulator ima pripadajuću URDF datoteku i ta nam činjenica omogućuje izradu univerzalnog upravljačkog sučelja.

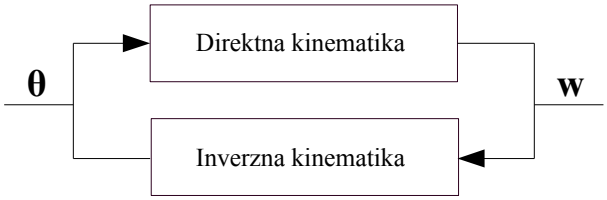
4.5.1. ROS integracija Gazebo paketa

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 koristeći `ros_control` sučelje, na isti način kao što upravljamo stvarnim sustavom. Ova kompatibilnost ostvaruje se kroz `gazebo_ros`, `gazebo_msgs`, `gazebo_plugins` ROS pakete. Ros control sučelje 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 se pokazalo vrlo korisnim.

5. Kinematika manipulatora

U idealnom slučaju, rješavanje problema kinematike manipulatora proizvoljnog broja osi 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.

Rješavanje problema direktne kinematike u općem je slučaju jednostavno: D-H postupkom postavljamo koordinatne sustave vezane uz zglobove ,a potom određivanjem matrica homogenih transformacija dobivamo jednoznačno rješenje. Problem inverzne kinematike znatno je složeniji jer zahtjeva rješavanje sustava nelinearnih jednadžbi s više nepoznanica, što je vrlo često netrivijalno. Još jedna otežavajuća okolnost nalazi se u činjenici da će dobivena rješenja često biti višestruka, a pronalaženje odgovarajućeg ovisi o konstrukciji i poziciji manipulatora.



Slika 5.1: Model Jaco robotske ruke sa označenim parametrima

Pronalaženje analitičkih rješenja inverzne kinematike pomoću računalnog algoritma vrlo je složen zadatak, pogotovo uzevši u obzir činjenicu da svako rješenje nije nužno fizički moguće. 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 potpoglavlju 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. Drugo potpoglavlje sadrži opis ideje rješenja inverzne kinematike koje koristimo u računalnom algoritmu. Rezultate zamišljenog postupka potom ispitujemo koristeći Matlab.

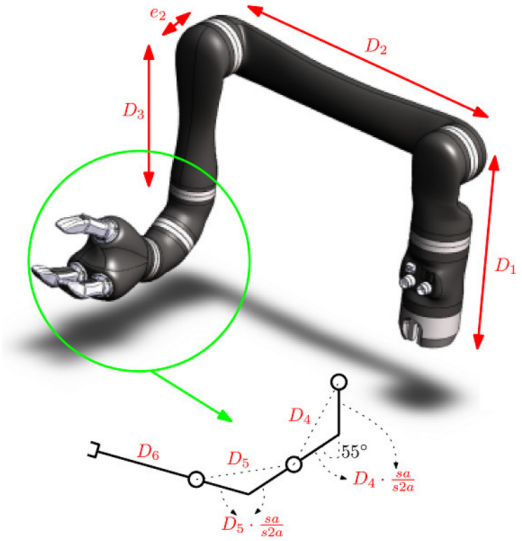
5.1. Direktna i inverzna kinematika Jaco robotske ruke

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.



Slika 5.2: Model Jaco robotske ruke sa označenim parametrima

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

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

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

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

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:

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

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 :

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

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

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

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

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

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

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.

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

Uvrštavajući podatke iz tablice 5.2 u izraz 5.10 dobivamo matrice homogenih transformacija za koordinatne sustave Jaco robotske ruke.

$$\mathbf{T^1_0} = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & 0 \\ \sin\theta_1 & 0 & -\cos\theta_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.11}$$

$$\mathbf{T^2_1} = \begin{bmatrix} \cos\theta_2 & \sin\theta_2 & 0 & a_2 \cos\theta_2 \\ \sin\theta_2 & -\cos\theta_2 & 0 & a_2 \sin\theta_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.12}$$

$$\mathbf{T^3_2} = \begin{bmatrix} \cos\theta_3 & 0 & \sin\theta_3 & 0 \\ \sin\theta_3 & 0 & -\cos\theta_3 & 0 \\ 0 & 1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.13}$$

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

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

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

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 $\mathbb{R}^3$ 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 $\boldsymbol{\theta_i}(\mathbf{t_i})$. 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 vrlo složen 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, razvili smo 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 [poglavlje].

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 manipulator, 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:

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

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:

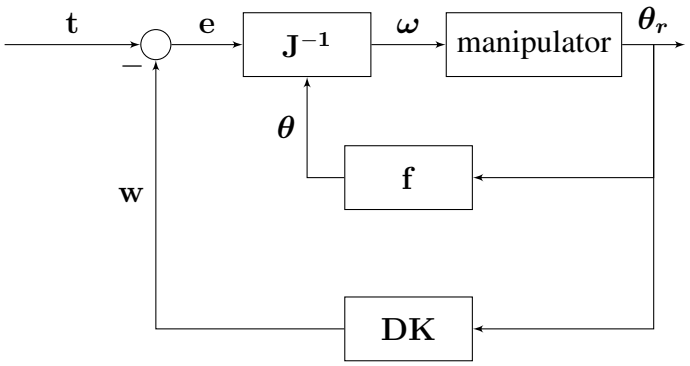
$$\mathbf{J} = \begin{bmatrix} J_{\boldsymbol{v}} \\ J_{\boldsymbol{\omega}} \end{bmatrix} \tag{5.27}$$

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

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

5.2.2. Upravljačka petlja

Razmatranjem izraza 5.22 možemo zaključiti da smo posredno došli do alata za aproksimaciju inverzne kinematike. Iterativnom metodom moguće je ostvariti kretanje ruke prema željenoj točki dokle god poznajemo $\mathbf{e}$. Ovom metodom pomoću već postojećih podataka možemo jednostavno dobiti rješenje za linearno kretanje bilo koje točke manipulatora.



Slika 5.3: Blok dijagram izvedbe upravljače petlje manipulatora

Definiramo vektor razlike konfiguracija kao:

$$\mathbf{e_i} = \mathbf{t_i} - \mathbf{w_i} \tag{5.29}$$

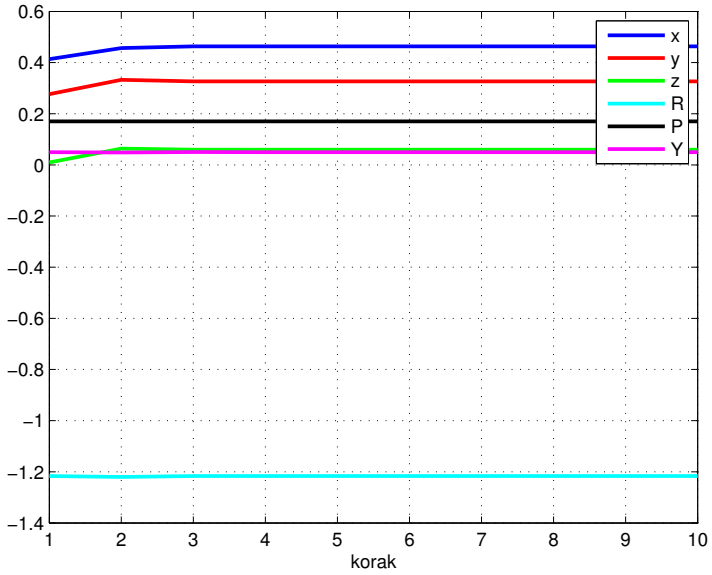
U daljnjem razmatranju iz izraza 5.29 ispuštamo indekse radi kompaktnosti. Razliku referentnog i stvarnog vektora konfiguracije alata sustav prema izrazu 5.22 množimo inverzom matrice $\mathbf{J}$. Kao rezultat množenja dobivamo potrebne zakrete zglobova $\Delta\mathbf{q}$. Pošto su rezultirajući referentni zakreti zglobova znatno manji od njihove nominalne kutne brzine, možemo ih tretirati i kao referentne brzine:

$$\Delta\mathbf{q} \approx \boldsymbol{\omega} \tag{5.30}$$

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

Shemu sustava predloženu shemom 5.3 provjeravamo u programskom paketu Matlab. 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 za 5cm u smjeru x,y i z osi. Orijentacija alata je konstantna.

Iz grafa 5.4 vidljivo je kako pri čistom translacijskom kretanju upravljačka petlja sa P regulatorom ostvaruje željenu konfiguraciju unutar 3 koraka. Graf 5.5 a) prikazuje ponašanje P upravljačke petlje pri složenijem gibanju koje obuhvaća i promjenu orijentacije. Visoka razina oscilatornosti navodi nas na zaključak kako je za složenija gibanja potreban složeniji oblik regulatora.

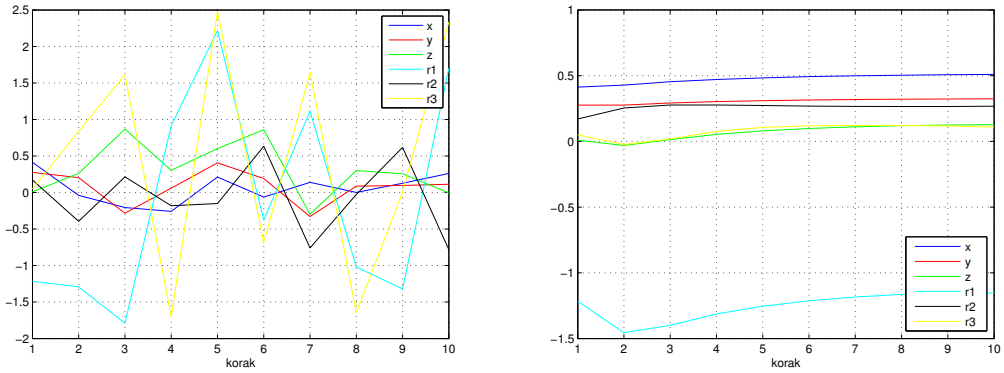
Proporcionalno-integralnim (PI) regulatorom možemo značajno smanjiti oscilatornost sustava. Pošto se regulator izvodi u kodu, potrebno ga je ostvariti rekurzivno:

$$\mathbf{u}(i) = K_I\mathbf{u}(i-1) + K_I[\mathbf{e}(i) - \mathbf{e}(i-1)] + K_P\mathbf{e}(i) \tag{5.31}$$

Parametriranje regulatora ostvarujemo pomoću optimizacije kriterijske funkcije 5.32 koja kažnjava oscilatornost i vrijeme potrebno za postizanje stacionarnog stanja.

$$crit(k) = crit(k-1) + 100k(\mathbf{t} - \mathbf{w}(k))(\mathbf{t} - \mathbf{w}(k))^T \tag{5.32}$$

Kako bi pronašli parametre K_I i K_P kojima ostvarujemo minimum ove funkcije koristimo Matlab funkciju `fminsearch`. Rezultirajući odziv matematičkog modela na grafu 5.5 b) znatno je stabilniji korištenjem PI regulatora sa parametrima



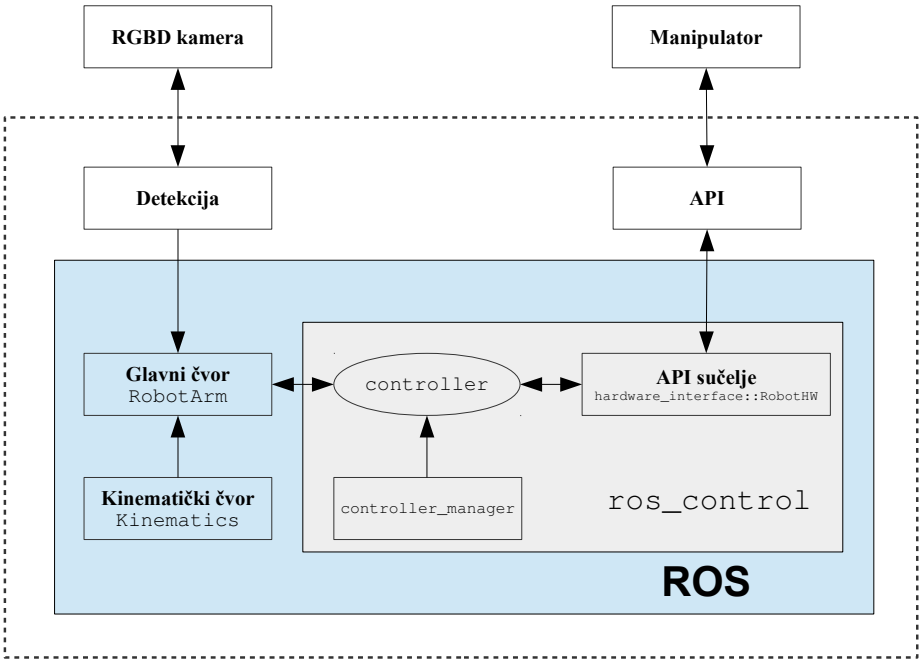
Slika 5.5: Odzivi upravljanja pri referentnoj konfiguraciji koja iziskuje regulaciju svih stupnjeva slobode. Slika a) (lijevo) sadrži odziv sustava sa P regulatorom, b) (desno) sadrži odziv sustava sa PI regulatorom.

$[K_I\ K_P] = [-0.2154\ 0.3197]$. Potrebno je uzeti u obzir činjenicu da u ovom procesu nismo modelirali ponašanje samih elektromotora u zglobovima jer pretpostavljamo da su kvalitetno regulirani od strane FPGA pločice na samom manipulatoru. Ovi parametri također ovise o Jakobijanu korištenog manipulatora, ali u ovom potpoglavlju dovoljno nam je bilo dokazati da predložena struktura radi.

6. Detekcijski algoritam

7. Upravljački algoritam

Koristimo ranije objašnjene koncepte i metode kako bi detaljno objasnili arhitekturu i izvedbu upravljačke petlje. U prvom potpoglavlju opisujemo sve uključene ROS čvorove, poruke i servise koji čine kostur upravljačkog algoritma. Drugo potpoglavlje sadrži opis programskog rješenja kinematike izvedenog u KDL-u.



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

8. Rezultati