Planiranje gibanja robotske ruke imitacijom pokreta čovjeka praćenog monokularnom kamerom

Jambrešić, Ana

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 63

PLANIRANJE GIBANJA ROBOTSKE RUKE IMITACIJOM POKRETA ČOVJEKA PRAĆENOG MONOKULARNOM KAMEROM

Ana Jambrešić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 63

PLANIRANJE GIBANJA ROBOTSKE RUKE IMITACIJOM POKRETA ČOVJEKA PRAĆENOG MONOKULARNOM KAMEROM

Ana Jambrešić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 63

Pristupnica:	**Ana Jambrešić (0036518990)**
Studij:	Informacijska i komunikacijska tehnologija
Profil:	Automatika i robotika
Mentorica:	prof. dr. sc. Marija Seder

Zadatak: **Planiranje gibanja robotske ruke imitacijom pokreta čovjeka praćenog monokularnom kamerom**

Opis zadatka:

Teleoperacija robota bitan je alat za obavljanje zadataka čija složenost nadilazi mogućnosti postojećih algoritama. Postojeće metode upravljanja na daljinu često nisu intuitivne za ljudske operatere ili zahtijevaju posebne senzore i opremu, što ih čini neisplativim i nepraktičnim u mnogim scenarijima. Cilj je ovoga diplomskog rada predložiti metodu teleoperacije robotske ruke koja se zasniva na slikama iz monokularne kamere. Predložena metoda prvo koristi jednostavne neuronske mreže za procjenu položaja ljudskog tijela i prepoznavanje čovjekove geste rukom. Učinkovit algoritam inverzne kinematike zatim pronalazi željenu konfiguraciju ruke robota, postižući kretanje izvršnog člana robota koje oponaša kretanje šake operatera. Predložena metoda za teleoperaciju izvršavat će se na prijenosnom računalu pomoću ugrađene kamere te korištenjem Robotskog Operacijskog Sustava (ROS). Metoda će biti testirana u simulaciji te u laboratorijskom okruženju s robotskom rukom Kinova Jaco.

Rok za predaju rada: 28. lipnja 2024.

Najprije želim izraziti duboku zahvalnost svojoj mentorici, prof. dr. sc. Mariji Seder, i asistentu dr. sc. Luki Petroviću na stručnom vodstvu, podršci i savjetima tijekom cijelog procesa izrade ovog diplomskog rada.

Također, želim zahvaliti svojim roditeljima na neizmjernoj podršci i razumijevanju kroz cijeli proces studija. Iznimno sam zahvalna i svojim prijateljima na njihovoj pomoći, ohrabrenju i veselim trenucima koje smo dijelili. Posebnu zahvalu upućujem svojoj sestri blizanki Luciji, koja mi je napisala zadaću iz matematike u prvom razredu osnovne škole, što je bio moj prvi korak prema ovom trenutku.

Sadržaj

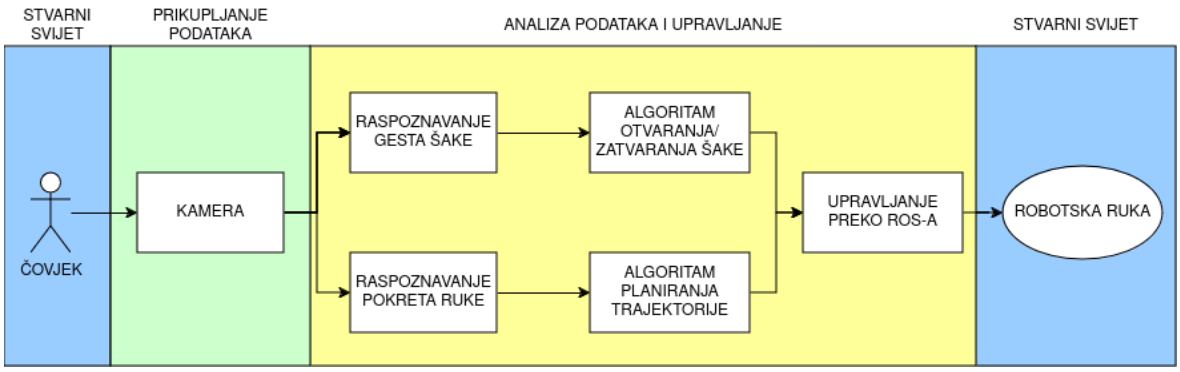
1. Uvod

U posljednjem desetljeću, napredak u području robotike značajno je transformirao industriju, medicinu, ali i naš svakodnevni život. Roboti su postali nezamjenjivi u izvršavanju opasnih i teških zadataka. Jedan od ključnih aspekata te transformacije jest mogućnost da roboti imitiraju ljudske pokrete, što otvara nove pristupe u interakciji između čovjeka i stroja. U ovom radu, fokusiramo se na pristup u području teleoperacije robotskih ruku, gdje se teži postizanju intuitivnih, prirodnih ljudskih pokreta robotske ruke kroz imitaciju, koristeći monokularnu kameru za praćenje pokreta čovjeka. Tradicionalne metode teleoperacije često nisu intuitivne za ljudske operatere, zahtijevajući složenu obuku i prilagodbu na specifične kontrole, što može biti nepraktično i skupo. S druge strane, imitacija ljudskih pokreta predstavlja prirodniji i intuitivniji pristup, ali je često otežana zbog potrebe za dodatnom opremom. Predloženi sustav u ovom radu rješava te izazove kroz razvoj teleoperacijskog okvira koji omogućava precizno prepoznavanje i rekonstrukciju ljudskih pokreta koristeći samo monokularnu kameru, te ih transformira u naredbe za pokretanje robotske ruke. Osnovna ideja ovog diplomskog rada temelji se na korištenju naprednih algoritama inverzne kinematike i jednostavnog otvorenog izvornog modela neuronske mreže za detekciju cjelokupnog stava tijela i gestikulacije ruke operatora. Ovo omogućuje sustavu da intuitivno i u realnom vremenu imitira pokrete zapešća operatora na robotskoj ruci, čime se omogućuje prirodna i efikasna teleoperacija. Korištenjem opće dostupne monokularne kamere, ovaj pristup postaje izuzetno troškovno učinkovit i primjenjiv na širok spektar uređaja, uključujući osobna računala i mobilne uređaje. Integracijom s Robot Operating System (ROS), te MediaPipe okvirom za računalni vid, ovaj rad predstavlja značajan doprinos u pojednostavnjivanju teleoperacijskih sustava. Cilj ovog diplomskog rada jest demonstrirati kako se kroz primjenu postojećih tehnologija i algoritama može postići visoka razina sinergije između čovjeka i robota, omogućujući robotskoj ruci da intuitivno i precizno slijedi ljudske pokrete. Kroz

simulacijske primjere u okruženju Gazebo i praktične eksperimente s robotskom rukom Kinova Jaco, ovaj rad potvrđuje performanse predloženog sustava, naglašavajući njegovu primjenjivost i potencijal za buduća istraživanja i razvoj u području robotike.

2. Teleoperacija

Teleoperacija predstavlja tehniku upravljanja sustavima s udaljene lokacije korištenjem komunikacijskih kanala [1] . U okviru ovog diplomskog rada, fokusiramo se na upravljanje robotskom rukom, koristeći se analizom snimke ljudskog pokreta dobivene monokularnom kamerom. Takav pristup omogućava precizno i intuitivno upravljanje robotskom rukom, oponašajući ljudske pokrete u stvarnom vremenu. To dovodi do visoke razine sinkronizacije i fluidnosti između ljudskog operatera i robotskog sustava. Kako bi ostvarili taj zadatak, prvi korak u ovom procesu jest snimanje ljudskog pokreta. Nakon toga, analiziramo dobivene snimke s ciljem estimacije položaja osobe, pokreta i gestikulacija njezinih ruku. Oslanjajući se na prikupljene podatke, kreiramo trajektoriju koja se zatim implementira u upravljanje robotskom rukom. Detaljan prikaz programskog okvira, koji je predložen u ovom radu, ilustriran je na slici 2.1.



Slika 2.1. Programski okvir

2.1. Praćenje pokreta čovjeka

Snimanje i prepoznavanje pokreta je proces detekcije pokreta nekog objekta ili čovjeka te pretvaranje tog pokreta u digitalni oblik. Tehnologiju snimanja pokreta zbog svoje raznolikosti moguće je koristiti u raznim područjima kao što je robotika, medicina, sport,

multimedija, video igre i slično [2]. Proces prepoznavanja pokreta koristi razne senzore i kamere koje prate i snimaju gibanje željenog objekta ili osobe. Dobivena snimka se zatim mapira kao 3D model izvedenog gibanja. Glavni cilj snimanja pokreta je hvatanje pokreta, obrada dobivenih podataka te njihova pohrana. Dobivene podatke možemo kasnije koristiti i integrirati u razne aplikacije.

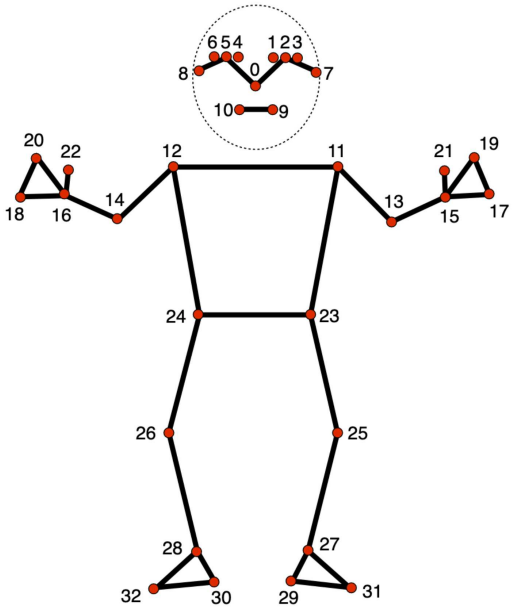
Postoje nekoliko metoda za prepoznavanje pokreta. Najčešće korištena je **optička pasivna metoda**, ona koristi infracrvene kamere za detekciju pasivnih markera. Sljedeća metoda, **optička aktivna metoda**, koristi posebne kamere koje prate LED markere koji emitiraju svijetlo i na temelju kašnjenja emisije svijetla estimiraju dubinu praćenog objekta. Također, koristi se i **inercijska metoda**. Kod korištenja inercijske metode praćeni objekt ima inercijske senzore koji šalju podatke računalu dok kamera služi samo za lokalizaciju. [2] Zadnji način prepoznavanja pokreta je preko **video snimke bez markera**, ovakve tehnologije se oslanjaju samo na softversko rješenje. Snimanje pokreta bez potrebe za postavljanjem markera na tijelu je korisno u situacijama gdje je potrebno snimati pokrete u stvarnom vremenu što nam daje prednost u stvaranju više prirodnog i neometanog korisničkog iskustva. Takva tehnologija eliminira potrebu za fizičkim markerima ili posebnom odjećom. To ne samo da pojednostavljuje proces snimanja, već također omogućava veću fleksibilnost i mobilnost sudionika. Uz to, ova metoda omogućava prikupljanje podataka u različitim okruženjima, od kontroliranih laboratorijskih uvjeta do stvarnih, nekontroliranih, vanjskih lokacija, čime se proširuje opseg istraživanja i primjena.

2.1.1. Analiza snimljenog pokreta

U ovom radu kao glavni alat za obradu snimljenog pokreta koristimo programski okvir MediaPipe, razvijen od strane Googlea. MediaPipe je alat otvorenog koda namijenjen za stvaranje složenih modela strojnog učenja za percepciju, koji omogućava brzu i efikasnu analizu podataka u stvarnom vremenu. Ovaj okvir uključuje raznovrsne module koji se mogu primijeniti u skladu s funkcionalnim zahtjevima projekta. U sklopu ovog rada potrebno je detektirati poziciju ruke čovjeka i geste šake. Kako bi detektirali ključne točke pozicije čovjeka, a samim time i ruke, koristimo "Pose Landmark Detection" rutinu dok za prepoznavanje gesta šake čovjeka koristimo "Hand Gesture Recognition" rutinu. Upravo ta modularna struktura MediaPipe alata omogućuje raspoređivanje modela stroj-

nog učenja na različitim platformama.

MediaPipe Pose Landmark Detection rutina omogućuje detekciju ključnih točaka ljudskog tijela na videu ili slici. Koristi modele strojnog učenja koji rade s jednom slikom ili videom. Na temelju unaprijed snimljene statične slike ili dobivenih video kadrova preko monokularne kamere generiraju se normalizirane koordinate pozicija čovjeka sa slike i vrijednosti pozicije čovjeka u koordinatama svijeta. Za predikciju, ovaj alat koristi niz modela strojnog učenja. Prvi model detektira prisustvo ljudskog tijela u okviru slike, dok drugi locira ključne točke ljudskog tijela. Oba modela su zajedno zapakirana i koriste konvolucijsku neuronsku mrežu BlazePose arhitekture. [3]

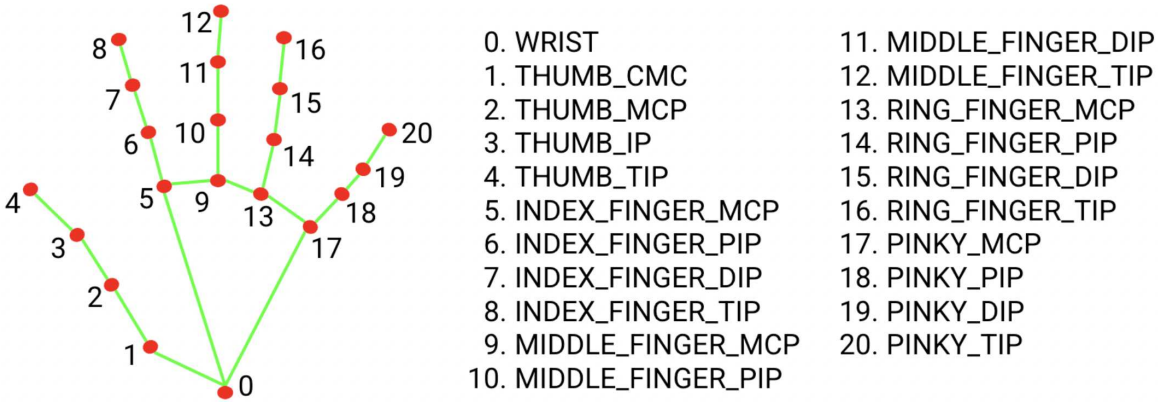


Slika 2.2. 33 ključne točke ljudskog tijela
[3]

BlazePose je jednostavna konvolucijska neuronska mreža za estimaciju pozicije čovjeka u stvarnom vremenu. Generira 33 ključnih orijentira ljudskog tijela, omogućujući detaljan uvid u ljudsko tijelo. Građa neuronske mreže inspirirana je "stacked-hourglass" arhitekturom i sastoji se od enkoder-dekoder mrežne arhitekture za predikciju glavnih zglobova. Osnovna struktura sastoji se od detektora i mehanizma praćenja. Detektor prvo identificira glavni okvir tijela, a zatim, prema detekciji glave čovjeka se radi predikcija drugih dodatnih parametara ljudskog tijela. Za praćenje svih pozicija tijela, rotaciju estimiramo kao liniju između dobivenih kukova i ramena ljudskog tijela. Jedna od ključnih značajki BlazePose neuronske mreže je sposobnost praćenja orijentira i kada su oni

sakriveni. U slučaju kada je dio osobe sakriven omogućeno je praćenje zbog uvođenja parametra koji mijenja vrijednost ovisno je li točka sakrivena. Tijekom faze treniranja, neuronska mreža se izlaže simulacijama situacija u kojima su dijelovi tijela blokirani ili sakriveni, što omogućuje razvoj robusnog modela. [4]
Kako bi model estimirao položaj, definirane su 33 točke ljudskog tijela kao orijentiri. Navedene točke prikazane su na slici 2.2. [3]. Ključne točke ljudskog tijela označene su indeksima od 0 do 32. Tako na primjer, desna ruka definirana je indeksima 12, 14, 16, 18, 20 i 22. U okviru ovog rada, koristi se pozicija lijevog zgloba, izdvojena indeksom 15.

MediaPipe Hand Gesture Recognition rutina predstavlja napredan sustav koristeći se dubokim učenjem i računalnim vidom. Zahvaljujući svojoj sposobnosti da precizno detektira položaj i pokrete šake u stvarnom vremenu, ovaj alat nalazi primjenu u različitim područjima, uključujući interakciju čovjeka i računala. Za precizno prepoznavanje šake koristi dva međusobno zapakirana modela. Prvi model, model ključnih orijentira šake, koristi se za detekciju 21 ključne točke na šaci (2.3.) [5]. Ove točke obuhvaćaju sve značajne dijelove šake, od vrhova prstiju do zgloba, omogućavajući detaljan prikaz geometrije šake. Detekcija dlanova i prstiju zajedničkim radom ovih modela omogućava efikasno praćenje i analizu pokreta šake. Drugi model, model klasifikacije gesta



Slika 2.3. 21 ključnih točaka ljudske šake
[5]

šake, ima mogućnost razlikovati 7 različitih gesta kao što su zatvoren/otvoren dlan, palac gore/dolje, pokazivanje prstom i druge često korištene geste. Sposobnost prepoznavanja ovih gesta otvara mogućnost upravljanju robotske ruke kako bi uhvatili i podigli neki predmet. [5]

2.2. Kinematika robotske ruke

Kinematika predstavlja osnovni aspekt u razumijevanju i projektiranju robotskih sustava. Ovaj dio robotike bavi se proučavanjem geometrije pokreta, opisuje i predviđa gibanja, bez razmatranja sila koje ih uzrokuju. Razumijevanje i korištenje kinematičkih jednadžba omogućuje nam precizno kontroliranje i upravljanje robotskim sustavima kako bi obavili kompleksne zadatke.

2.2.1. Direktna kinematika

Robotski manipulator može se modelirati kao lanac krutih članka međusobno povezani raznim vrstama zglobova. Zglobovi mogu biti rotacijski i linearni te takvi zglobovi imaju jedan stupanj slobode. Nepomična baza robota je na početku serije a završni mehanizam ili alat se nalazi na kraju. [6] Uz pretpostavku da svaki zglob ima jedan stupanj slobode, djelovanje svakog zgloba može se opisati jednim realnim brojem: kutem rotacije u slučaju zakretnog zgloba ili pomakom u slučaju prizmatičnog spoja. Pošto svaki zglob povezuje dva članka robota, manipulator s n zglobova ima $n + 1$ članak [7].
Direktna kinematika je jedan od temeljnih alata u robotici koja se bavi određivanjem pozicije i orijentacije vrha alata (na primjer, robotske ruke) poznavajući vrijednosti zakreta svih zglobova robotskog manipulatora. Cilj direktne kinematike je izračun koordinata vrha alata u prostoru alata na temelju zakreta zglobova u prostoru zglobova. Direktna kinematika omogućava precizno upravljanje robotom pri izvođenju raznih zadataka.
Međutim, matematička analiza manipulatora s n članaka može biti iznimno složena. Jedna od najpopularnijih metoda za rješavanje problema direktne kinematike u robotici je **Denavit-Hartenberg (DH) metoda**. Ova metoda pruža sistematičan pristup za kreiranje matematičkog modela robota koristeći relativno jednostavan skup parametara. DH metoda koristi četiri parametra za opisivanje svake veze između zglobova robota [7].
Kinematički parametri članka:

1. θ - **kut zgloba** - kut rotacije oko z-osi prethodnog zgloba robota tako da x-os trenutnog i x-os prethodnog zgloba budu paralelne,

2. d - **pomak članka** - pomak dobiven translacijom z-osi prethodnog zgloba tako da im x-osi budu kolinearne, reprezentira udaljenost između ishodišta $i - tog$ koordi-

natnog sustava i presjeka x_{i+1}-osi s z_i-osi mjerene duž z_i-osi,

3. α - **zakret članka** - kut rotacije oko x-osi prethodnog zgloba tako da z-osi budu paralelne,

4. a - **duljina članka** - pomak dobiven translacijom x-osi prethodnog zgloba tako da im z-osi budu kolinearne, reprezentira duljinu između z_i i z_{i+1} osi i mjeri se po $x_i - osi$.

Koraci primjene DH metode

1. **Postavljanje koordinatnih sustava** - svaki zglob ima svoj koordinatni sustav tako da z-os gleda u smjeru djelovanja, x_i-os se određuje prema z_i i z_{i-1} koordinatnim sustavima, dok y-os postavimo prema pravilu desne ruke uvažavajući prethodno postavljene x i z osi

2. **Određivanje DH parametara** - svaka veza između dva uzastopna zgloba se opisuje koristeći gore navedena četiri DH parametra

3. **Izračun transformacijske matrice** - za svaki par uzastopnih zglobova formiramo transformacijsku matricu prema formuli (2.1), ovako postavljena matrica predstavlja rotaciju i translaciju od jednog koordinatnog sustava zgloba do drugog, sljedećeg koordinatnog sustava zgloba

$$T_{i-1}^{i} = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i)\sin(\theta_i) & \sin(\alpha_i)\sin(\theta_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & -\sin(\alpha_i)\cos(\theta_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

4. **Množenje transformacijski matrica** - konačna pozicija i orijentacija vrha alata dobi se množenjem svih transformacijski matrica s desna kako je navedeno u (2.2), počevši od baze robota do vrha alata

$$T_0^n = T_0^1 * T_1^2 * ...T_{n-1}^n \tag{2.2}$$

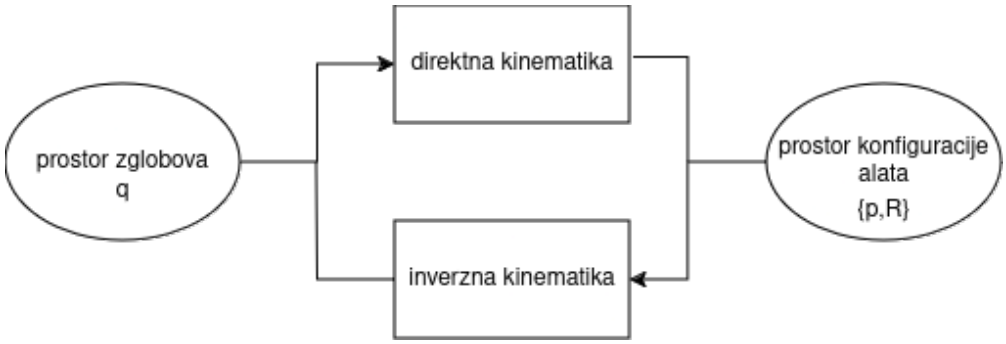
$$T_0^n = \begin{bmatrix} R_0^n & p_0^n \\ 0 & 1 \end{bmatrix} \tag{2.3}$$

Množenjem svih matrica homogene transformacije (2.2) dobiva se matrica složene homogene transformacije (2.3), koja koordinate ortonormiranog koordinatnog sustava alata transformira u koordinate ortonormiranog koordinatnog sustava baze. Iz te matrice određujemo konačnu orijentaciju R_0^n i poziciju u prostoru alata p_0^n vrha alata robotskog manipulatora. [6] Korištenjem DH metode, možemo sistematično i efikasno modelirati kompleksne robotske sustave, što čini tu metodu vrlo popularnim alatom u industrijskoj robotici i automatizaciji. Također, navedeni pristup transformacije homogenih koordinata moguće je koristiti i u odnosu prema drugim, dodatnim, koordinatnim sustavima koji se mogu vezati na robota. To su na primjer koordinatni sustavi raznih senzora, kao što su kamere, sonari i slično [6].

2.2.2. Inverzna kinematika

Kako bi robot izvršio zadani zadatak, potrebno mu je zadati točke u prostoru koje vrh alata mora proći. A to znači da je potrebno zadati položaj i orijentaciju alata u tim točkama prema koordinatnom sustavu baze robota [6]. Inverzna kinematika je postupak u robotici koji se bavi određivanjem potrebnih parametara zakreta zglobova robota kako bi se postigao željeni položaj i orijentacija vrha alata robotskog manipulatora. Za razliku od direktne kinematike, koje koriste poznate parametre zakreta zglobova da bi se izračunao položaj vrha alata, inverzna kinematika na temelju željenog položaja vrha alata traži odgovarajuće vrijednosti zakreta zglobova robotskog manipulatora koje vode ka tom položaju. Slika 2.4. [6] ilustrativno prikazuje odnos i povezanost direktne i inverzne kinematike.

Rješenja problema inverzne kinematike često su kompleksnija od direktne kinematike jer jednadžbe mogu biti nelinearne i može postojati više rješenja ili čak beskonačna rješenja, također mogu se pojaviti nedopustiva rješenja koja izlaze iz prostora robota. [8] Za rješavanje problema inverzne kinematike postoje razne metode. Jedna od metoda, je **analitička metoda** gdje direktnim izvođenjem matematičkih formula računamo vrijednosti zakreta zglobova robotskog manipulatora. Analitički pristup je brz i precizan no primjenjiv je samo kada postoji jasna matematička formulacija rješenja, što nije uvijek



Slika 2.4. Diretkna i inverzna kinematika

slučaj. Također, postoje **geometrijske metode** koje se koriste za jednostavnije konfiguracije robota i oslanjaju se na geometrijske odnose i trigonometriju kako bi se izračunali željeni zakreti zglobova robotskog manipulatora. Kada analitički ili geometrijski pristup nije moguć, moguće je koristiti **iterativne numeričke metode**. Prednost numeričkih postupaka je izrada univerzalnog programa za različite konfiguracije robota. No problem dolazi kod neprepoznavanja singularnih stanja robota i moguće divergencije postupaka računanja [6]. Ako postoji rješenje inverznog kinematičkog problema, ono ne mora biti jedinstveno. To znači da isti položaj i orijentaciju vrha alata je moguće postići s više raznih kombinacija vrijednosti zakreta zglobova robota.

Inverzna kinematika je izuzetno važna u industrijskoj robotici, pogotovo u primjenama gdje roboti moraju postići točno definirane i precizne pozicije kako bi izvršili razne zadatke kao što su zavarivanje, montaža ili neka manipulacija proizvodnim dijelovima. Također, koristi se u animaciji i virtualnoj stvarnosti za kontrolu pokreta likova i virtualnih objekta na prirodan način.

2.2.3. Diferencijalna kinematika

Opisivanje odnosa između brzina zglobova i linearne i kutne brzine vrha alata opisano je **diferencijalnom kinematikom**. Jedna od iterativnih metoda je izračun rješenja pomoću **Jacobijeve matrice članka**. Jacobijeva matrica predstavlja vezu između infinitezimalnih promjena varijabli zglobova te infinitezimalnih linijskih i kutnih pomaka alata [6]. Ona ovisi o konfiguraciji robota i moguće ju je odrediti analitički. Jacobijeva matrica $J(q)$ jedan je od najvažnijih alata za pronalaženje singulariteta, analizu redundantnosti, određivanje inverzne kinematičke jednadžbe i opisivanje brzina i sila upravljivih elipsoida. [8]. Linearna brzina vrha alata $\dot{q}$ i kutna brzina ω opisuju se kao funkcija

brzina zglobova robota $\dot{q}$ povezane Jacobijevom matricom $J(q)$ prema (2.4) [8].

$$v = \begin{bmatrix} \dot{p} \\ \omega \end{bmatrix} = \begin{bmatrix} J_P \\ J_O \end{bmatrix} * \dot{q} \tag{2.4}$$

Gdje J_P označava matricu doprinosa brzina zglobova linearnoj brzini vrha alata dok J_O doprinos kutnoj brzini vrha alata. Vrijednost Jakobijan matrica se razlikuje ovisno je li zglob prizmatičan ili rotacijski. Doprinos **kutnoj** brzini ako je zglob i **prizmatičan** iznosi:

$$\dot{q} * J_{Oi} = 0 \tag{2.5}$$

a ako je zglob i **rotacijski**:

$$\dot{q} * J_{Oi} = \dot{(\theta_i)} * z_{i-1} \tag{2.6}$$

Također, doprinos **linearnoj** brzini ako je zglob i **prizmatičan** iznosi:

$$\dot{q} * J_{Pi} = \dot{(d_i)} * z_{i-1} \tag{2.7}$$

i ako je zglob i **rotacijski**:

$$\dot{q} * J_{Pi} = \dot{(\theta_i)} * z_{i-1} x (p - p_{i-1}) \tag{2.8}$$

Također, vidljivo je iz navedenih jednadžba da Jacobijeva matrica mijenja vrijednost ovisno o koordinatnom sustavu u kojem se izražava. U formulama (2.5), (2.6), (2.7) i (2.8), vrijednost z_{i-1} je dana u trećem stupcu rotacijske matrice R^0_{i-1}, dok su p i p_{i-1} vrijednosti iz četvrtog stupca transformacijske matrice T^0_n odnosno T^0_{n-1} [8]

2.2.4. Planiranje putanje i trajektorije

Planiranje putanje je ključni aspekt u navigaciji robotskih ruku, omogućavajući im da izvršavaju zadatke s visokom preciznošću i efikasnošću. **Putanja** je skup točaka u prostoru alata koje robotska ruka mora proći kako bi postigla željeno ponašanje. To jest, putanja je krivulja u prostoru konfiguracije alata [6]. Za razliku od putanje, **trajektorija** je krivulja u prostoru po kojoj robot putuje u točno zadanim vremenskim trenutcima. Pod pojmom planiranje trajektorije spada proračun brzina i usklađivanje brzina te izračun

položaja zglobova svih osi gibanja sa zadanom putanjom gibanja vrha alata robota [6]. Postoji mnogo metoda za planiranje putanje.

Jedna od metoda je **interpoliranje**, ona je jednostavna za implementaciju i korisna kod kratkih udaljenosti no može rezultirati neočekivanim pokretima zglobova robotske ruke. Takvu metodu koristimo kada su zadane samo spojne točke, to jest, početak i završetak te krivulje. Između spojnih točaka provodimo interpolaciju, to jest, izračunavanje novih točaka na temelju zadanih, kako bi dobili glatku putanju. Jedan od načina interpolacije jest pomoću kubnih polinoma tipa (2.9) [6].

$$w(t) = A * t^3 + B * t^2 + C * t + D, 0 \leq t \leq T \tag{2.9}$$

Također, ako želimo ostvariti glatke putanje kontroliranjem brzina i ubrzanja u svakoj točci kako bi dobili fluidan pokret robotske ruke, moguće je koristiti **metodu Splineova**. U navedenoj metodi, segmente krivulje opisujemo spline-funkcijama. Takve funkcije zadovoljavaju svojstvo kontinuiteta i pogodne su za proračunavanje na standardnim računalima [6]. No, navedena metoda zahtjeva složenije izračune.

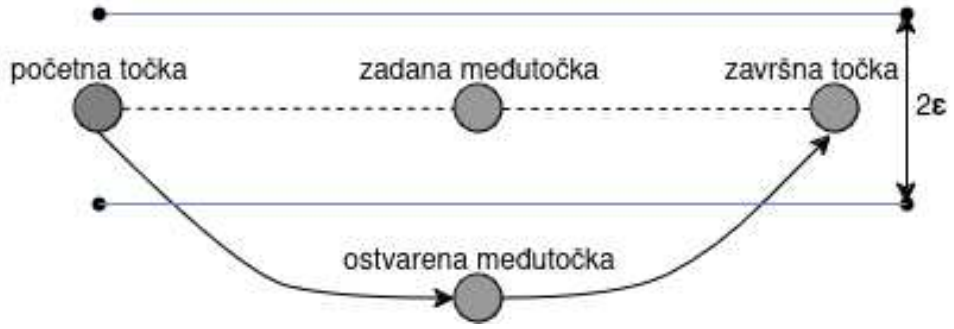
Isto tako, za planiranje putanje, moguće je koristiti neki od **"Sampling-based" algoritama** kao što su Rapidly-exploring Random Tree (RRT) ili Probabilistic Roadmaps (PRM). Takve metode nasumično uzrokuju okoliš u potrazi za adekvatnom putanjom. Korisne su za planiranje putanje u složenim okruženjima s preprekama, no složena je implementacija i mogu biti vremenski zahtjevne.

Još jedna od metoda za planiranje putanje je **Taylorova metoda**. Ona omogućava stvaranje pravocrtnog kretanja i relativno je jednostavna za implementaciju, no često je teško postići visoku preciznost u složenijim putanja ili s preprekama.

2.2.5. Taylorova metoda

Taylorovu metodu koristimo kako bi dobili pravocrtno, što pravilnije i fluidnije gibanje robotske ruke od točke do točke. Pravocrtno gibanje je kretanje vrha alata robota po krivulji koja ima najmanju duljinu između dviju točaka u radnom prostoru [6]. Ona se temelji na definiranom početnom i krajnjom točkom u prostoru alata i dodavanju potrebnih među-točka između njih. Taylorovom metodom dobivamo optimalan broj spojnih točaka na krivulji. Ova metoda se zasniva na pretpostavci da je odstupanje trajektorije

stvorene pravocrtnim gibanjem od zadane pravocrtne trajektorije najveće u blizini točke koja određuje sredinu trajektorije, takozvanoj, sukcesivnoj aproksimaciji 2.5. [6].



Slika 2.5. Taylorova metoda

Koraci primjene Taylorove metode [6]

1. Izabrati parametar $\epsilon > 0$ kao vrijednost tolerancije koji određuje granicu iza koje odbacujemo dobivene međutočke

2. Izračunati međutočku u prostoru zglobova $q^m = (q^0 + q^1)/2$ kao sredinu između početne i završne točke

3. Uz pomoć inverzne kinematike odrediti vrijednost pozicije vrha alata za dobivenu međutočku w^m

4. Izračunati točnu međutočku u prostoru konfiguracije alata kao $w^M = (w^0 + w^1)/2$

5. Usporediti odstupanje $||w^m - w^M|| \leq \epsilon$, te ako je ono zadovoljeno postupak je završen, a ako nije, dodati međutočku w^M

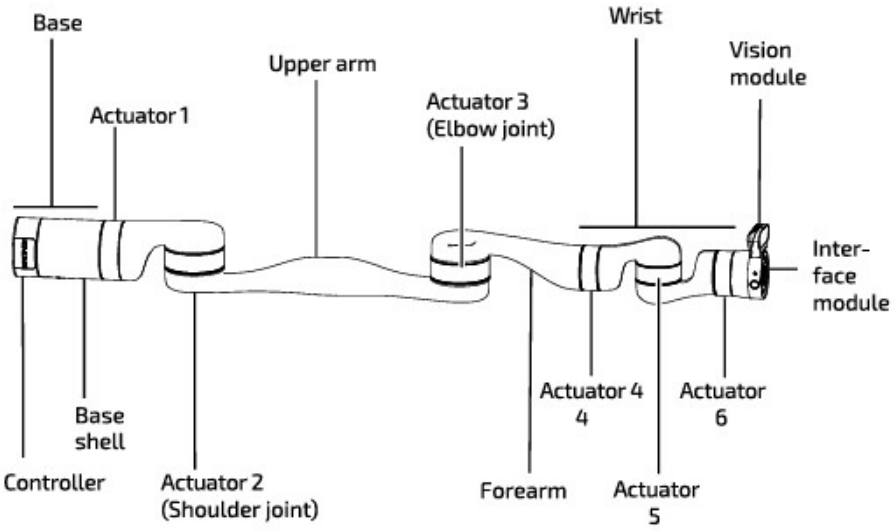
6. Rekurzivno se vratiti na postupak koristeći dva novodobivena segmenta w^0, w^M i w^M, w^1

Taylorova metoda može postati neučinkovita u oštrim vrhovima nekog lika zbog naglih promjena derivacije [6]. Također, mogućnost je dobivanja velikog broja međutočka, a nad svakom među-točkom primjenjujemo postupak inverzne kinematike. Stoga dolazi do velike računske zahtjevnosti. Zato Taylorova metoda ona nije optimalna za korištenje u slučajevima kada želimo izvesti gibanje robotske ruke u stvarnom vremenu.

2.3. Upravljanje robotskom rukom

2.3.1. Kinova Jaco robotska ruka

Za izvođenje pokreta u stvarnom svijetu kao robotski manipulator koristimo robotsku ruku Jaco (slika 2.6.) razvijenu od Kinova Robotics. Ova robotska ruka ima šest stupnjeva slobode što joj omogućuje veliku pokretljivost i preciznost, stoga je moguće njome vrlo dobro rekonstruirati pokret čovjeka. Kinova Jaco robotska ruka sastoji se od baze robota, raznih aktuatora i vrha alata. Za potrebe ovog diplomskog, kao vrh alata, korištena je hvataljka s tri prsta. Ruka ima hvataljku s tri prsta i tri stupnja slobode koji oponašaju ljudsku šaku. Svaki od tri prsta mogu se zasebno kontrolirati što omogućuje precizne, nalik ljudima, pokrete šake.

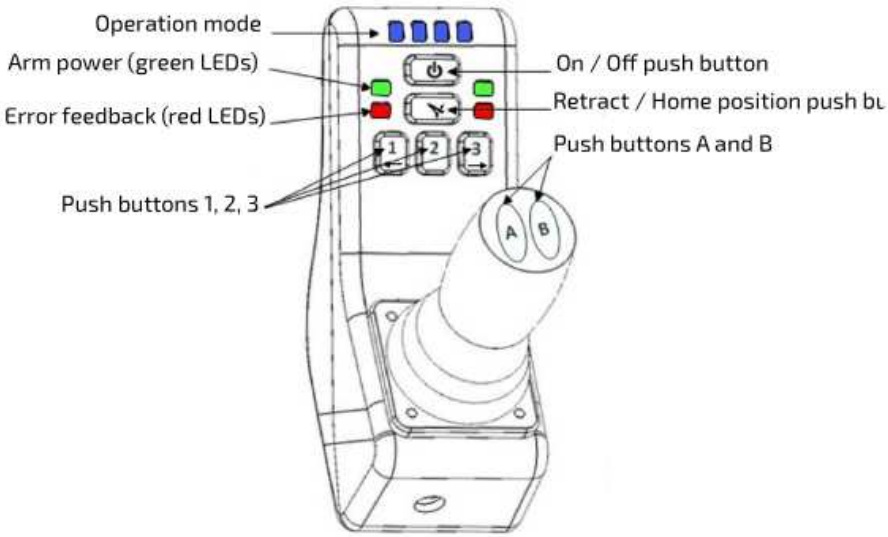


Slika 2.6. Kinova Jaco 6 DoF model robotske ruke [9]

Baza robota služi kao "mozak" robota, omogućuje montiranje robota na razne površine te napajanje i komunikaciju s drugim komponentama. Robotska ruka opremljena je aktuatorima koji omogućuju rotacijske pokrete. Aktuatori imaju senzore koji osiguravaju povratnu vezu robotu, što omogućuje sigurnost i prilagodbu pokreta ruke ovisno o okolišu. Senzori se koriste kako bi robotska ruka izbjegla prepreke u okolišu ili znala svoj trenutni položaj.

Navedenom robotskom rukom moguće je upravljati na više načina. Najviše intuitivan ljudima način je preko vanjskog kontrolera, odnosno joysticka prikazanog na slici 2.7. Standardni Kinova joystick ima mogućnost pomicanja u tri osi, lijevo/desno, naprijed/nazad i rotaciju te također uključuje pet nezavisnih gumba i četiri vanjska pomoćna pri-

ključka [9].



Slika 2.7. Vanjski kontroler Kinova Jaco robotske ruke [9]

Dodatno, razvijeno je sučelje koje omogućava uspostavljanje veze i komunikaciju s robotskom rukom kroz okruženje Robot Operating System, što dodatno proširuje mogućnosti upotrebe i integracije u različite aplikacije.

2.3.2. Robot Operating System

Robot Operating System (ROS) predstavlja besplatnu i otvorenu platformu namijenjenu za razvoj softverskih aplikacija za robote. Ovaj sustav posjeduje širok spektar alata, biblioteka i sposobnosti koje programerima omogućuju razvoj željenih aplikacije te time upravljanje i efikasno implementiranje željenog ponašanje robota. Zahvaljujući strukturi otvorenog koda, ROS nudi veliku fleksibilnost koja se lagano prilagođava raznim korisničkim potrebama te se jednostavno integrira s drugim postojećim softverskim rješenjima. [10]

ROS se sastoji od mnogo različitih komponenti, uključujući bogat skup upravljačkih programa koji omogućuju čitanje podatke s raznih senzora ili slanje upravljačkih signala aktuatorima. Također, ROS platforma uključuje veliki broj osnovnih robotskih algoritama koji podupiru temeljne funkcije u robotici, kao što je izgradnja karte okoliša, navigacija u okolišu, manipulacija objektima i mnoge druge. Osim toga, ROS pruža veliki set alata koji korisnicima omogućuje jednostavnu vizualizaciju stanja robota, implementiranih

željenih algoritma te snimanje i analizu podatke dobivenih od senzora. [11]
ROS se sastoji od modularne strukture gdje se jedan softverski program zove čvor. Čvorovi međusobno komuniciraju putem raznih poruka. Upravo ta modularnost je jedna od glavnih prednosti ROS platforme, tj. razdvajanje složenog zadatka na više jednostavnih čvorova i mogućnost daljnjeg korištenja tih čvorova u izgradnji drugačijih složenih funkcionalnosti. Ova raznolikost alata i funkcija čini ROS iznimno popularnim izborom za istraživača i inženjere koji se bave robotikom.

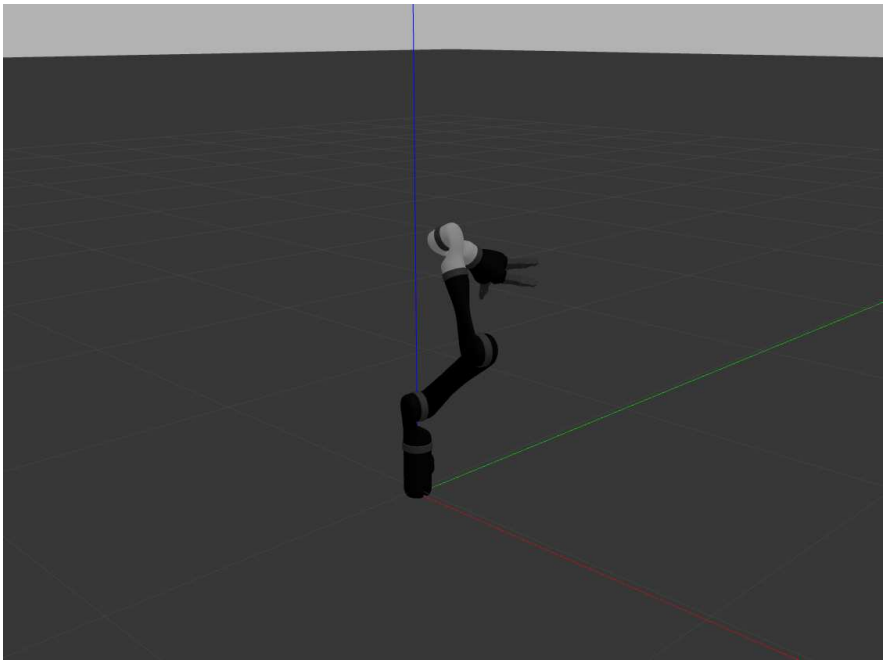
Kinova-ROS

Kinova-ROS predstavlja ROS sučelje za Kinova Robotics JACO, JACO2 i MICO robotske manipulatore [12]. Ono sadrži pakete koji omogućuju integraciju i upravljanje navedenim robotskim rukama kroz ROS sučelje.
Neki od ključnih paketa koji se nalaze unutar repozitorija su **kinova_bringup** koji se koristi za pokretanje potrebnih driver-a i postavljanje konfiguracija, **kinova_control** i **kinova_gezebo** koji se koriste za pokretanje Gazebo simulacije, **kinova_description** koji sadrži URDF opis modela robotske ruke te **kinova_msgs** koji sadrži prilagođene ROS poruke i servise.
Kod pokretanja robotske ruke potrebno je definirati tip ruke koji koristimo u formatu `{j|m|r|c}{1|2}{s|n}{4|6|7}{s|a}{2|3}{0}{0}` . Gdje `{j|m|r|c}` označava kategoriju ruke; Jaco, Mico, Roco ili prilagođeno. Zatim, `{1|2}` definira informaciju o korištenoj verziji robotske ruke, `{s|n}` označava tip zgloba šake koji može biti sferne konfiguracije ili ne-sferan. Sljedeće, `{s|a}` označava način rada robota koji može biti uslužni ili pomoćni. Dok `{2|3}` oznaka govori o vrhu alata i koliko ima prstiju [12] .
Robotskom rukom, unutar ROS-a je moguće upravljati na više načina. Možemo kontrolirati poziciju svakog zgloba u prostoru zglobova ili možemo upravljati pozicijom vrha alata u kartezijskom prostoru. **Kartezijski koordinatni prostor** je (x, y, z) prostor okoliša, dok **prostor zglobova** sadrži vrijednost zakreta svih zglobova robotske ruke $(q_1, q_2, ..., q_n)$. Također, moguće je upravljanje pozicijom prsta vrha alata robotske ruke. Isto tako, postoji mogućnost upravljanje brzinama u prostoru zglobova i u kartezijskom prostoru.

2.3.3. Gazebo Simulator

Gazebo je kolekcija biblioteka softvera otvorenog koda dizajnirana kako bi pojednostavila razvoj i performanse aplikacija [13]. Jedan od dijelova Gazeboa je Gazebo Simulator. To je 3D dinamički simulator koji omogućava točnu i učinkovitu simulaciju robota u raznim okruženjima. Glavne namjene Gazebo Simulatora uključuju testiranje algoritama korištenih u robotici, projektiranje robota te izvođenje regresijskog testiranja u realnim scenarijima. Zahvaljujući značajkama poput bogate biblioteke modela robota i okruženja, velikog izbora senzora i slično, Gazebo Simulator je odabran kao simulator unutar ovog diplomskog rada.

Također, Kinova-ros repozitorij dolazi s kinova_gezebo paketom. Kako bi upravljali robotom unutar Gazebo Simulatora koristimo se ros_control paketom. To je generični paket koji omogućuje pokretanje aktuatora robota na temelju danih upravljačkih veličina. Dostupna su tri načina upravljanja, upravljanje silom, pozicijom i brzinom [14].



Slika 2.8. Kinova Jaco 6 DoF unutar Gazebo Simulatora

2.3.4. Algoritam upravljanja robotskom rukom

Kao što je već napisano, robotskom rukom Kinova Jaco unutar ROS okruženja, moguće je upravljati kontroliranjem pozicija zglobova, kontroliranjem pozicije vrha alata ili kontroliranjem brzinama u prostoru zglobova ili kartezijskom prostoru.

Kontrola zakreta pojedinih zglobova u prostoru zglobova podrazumijeva relativno ili apsolutno zakretanje pojedinog zgloba robotske ruke za unaprijed definirani broj stupnjeva. Ova vrsta upravljanja posebno je korisna u situacijama kada vrh alata i ostali zglobovi moraju postići precizno određenu konfiguraciju i položaj. To je važno primjerice, kada ruka mora manevrirati kroz uske prostore ili prepreke, a pritom izvršavati specifične zadatke. U takvim okolnostima, mogućnost preciznog pozicioniranja svakog zgloba omogućava robotskoj ruci da efikasno i sigurno obavlja zadatak bez rizika od oštećenja objekata u svom radnom okruženju. Unutar kinova_ros paketa nalazi se čvorklijent naziva joints_action_client.py čijim pozivanjem omogućuje se navedeno pomicanje robotske ruke. Ovaj klijent prima tri parametara, tip robota (prema gore opisanom formatu), mjernu jedinicu izraženu u stupnjevima ili radijanima i vrijednosti zakreta svakog od n zglobova robota.

Također, unutar kinova_ros paketa nalazi se klijent naziva pose_action_client.py preko kojeg korisnici imaju mogućnost laganog upravljanja robotskom rukom na temelju kartezijskih pozicija. Ovaj klijent također prima tri parametra, tip robota, mjernu jedinicu izraženu u metrima i quaternionima ili metrima i stupnjevima ili metrima i radijanima, te željenu poziciju i orijentaciju vrha alata. On omogućuje apsolutno ili relativno pomicanje vrha alata na temelju zadanih parametra. U ovakvom slučaju, potrebno nam je znanje o inverznoj kinematici. Unutar svakog Kinova robota je implementiran algoritam inverzne kinematike koji automatski izbjegava singularitete i kolizije ruke same sa sobom. Stoga treba imati na umu da nije svaki zadani položaj moguće ostvariti na siguran način. Ovako upravljanje korisno nam je u situacijama kada moramo dovesti vrh alata u točno određeni položaj, neovisno o zakretu ostalih zglobova robotske ruke.

Kao što je to navedeno, cilj ovog diplomskog rada je pomicanje robotske ruke na temelju demonstracije čovjeka i dobivenih podataka detekcijom monokularnom kamerom. To jest, pomicanje robotske ruke želimo izvesti čim fluidnije i bliže stvarnom vremenu kako bi bilo što više nalik čovjekovom gibanju.

Predloženi sustav demonstriramo i testiramo najprije u simulaciji a nakon toga na stvarnom postavu.

Algoritam upravljanja unutar simulacije

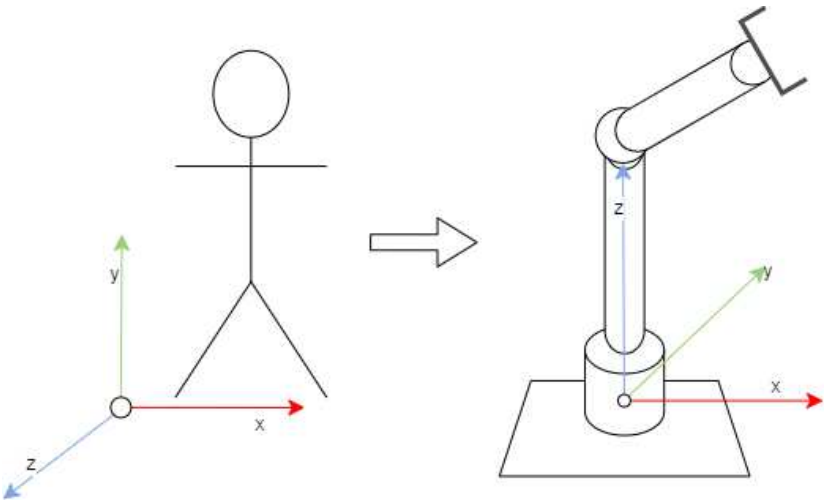
Kako bi simulirali pomicanje robotske ruke u Gazebo simulatoru, koristimo objavljivanje poruke tipa *JointTrajectory* na rostopic `/effort/joint/trajectory/controller/command`. Prema navedenom programskom okviru željene teleoperacije, iz monokularne kamere dobe se podaci o položaju lijeve šake praćenog čovjeka u 3D prostoru (vidi kod 2..1). Prikazani programski kod opisuje metodu definiranja kamere osobnog računala i hvatanje slike za obradu. Pomoću njega se dobe izlazni podaci modela za detekciju gesta šake (redak 30) i koordinate lijeve šake čovjeka (redak 24).

```
1  def cam(self):
2          #caputures video from a webcam and processes the frames to
       detect human pose and hand gestures
3          cap = cv2.VideoCapture(self.webcam_num)
4          time.sleep(1)
5          with mp_pose.Pose(
6              min_detection_confidence=0.5,
7              min_tracking_confidence=0.5) as pose:
8
9              while cap.isOpened():
10                 success, image = cap.read()
11                 if not success:
12                     print("Ignoring empty camera frame.")
13                     continue
14
15                 image.flags.writeable = False
16                 image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
17                 mp_image = mp.Image(image_format=mp.ImageFormat.SRGB,
       data=image)
18                 pose_results = pose.process(image)
                       # Pose results
19                 hand_gesture_recognition_result = recognizer.recognize
       (mp_image)          # hand_gesture_recognition_result
20
21                 # Taking the position of the desired joint
22                 if pose_results.pose_world_landmarks != None :
23                     cap.release()
24                     self.pose = pose_results.pose_world_landmarks.
       landmark[15]
```

```
                if str(hand_gesture_recognition_result.gestures) != '
[]':
                    #get hand gesture
                    score  = str(hand_gesture_recognition_result.
gestures).split()[1].split('=')[-1]
                    gesture_name = str(hand_gesture_recognition_result
.gestures).split()[-1].split('=')[-1]
                    score = score[:-1]
                    self.hand_gesture = gesture_name[:-3]
                if self.camview:
                    image.flags.writeable = True
                    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
                    cv2.imshow('MediaPipe Pose', cv2.flip(image, 1))
                    if cv2.waitKey(5) & 0xFF == 27:
                        #checks if the ESC key is pressed
                        break
        cap.release()
```

Listing 2..1: Metoda definiranja kamere i hvatanja slike

Na slici 2.9. ilustrativno je prikazan koordinatni sustav kamere i koordinatni sustav robotske ruke. U koordinatama dobivenih iz kamere, $z - os$ predstavlja dubinu položaja šake čovjeka u prostoru. Stoga, je navedene koordinate potrebno transformirati u koordinatni sustav robota. Zatim, na temelju razlike sljedeće to jest, željene pozicije i trenutne



Slika 2.9. Koordinatni sustavi kamere i robotske ruke

pozicije robotske ruke računa se potreba konfiguracija zakreta robotske ruke kako bi ju

doveli u željeni položaj. Ta konfiguracija se računa s pomoću Jakobijana dobivenog numerički. Pošto se radi samo o pozicijama, numerički dobiven Jakobijan nam je dovoljno točan. Kako bi ga odredili radi se numerička diferencijacija prema formuli (2.10) [15]. To jest, računamo ako se svaki zglob pomakne za neki mali pomak koliko se promijeni pozicija vrha alata.

$$f(\dot{x}) = \frac{f(x+h) - f(x-h)}{2h} \tag{2.10}$$

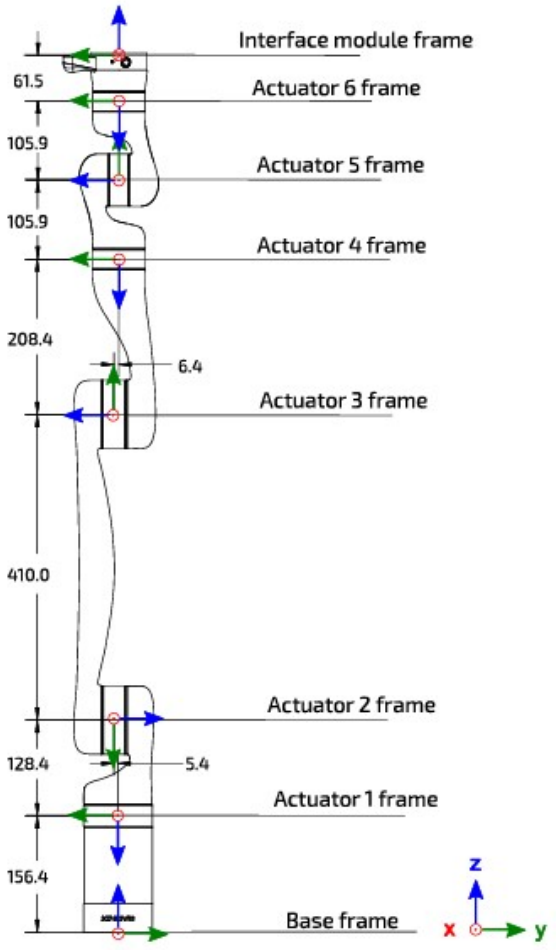
Trenutni zakreti zglobova robota dobiveni su s pomoću rostopic-a `/joint_states`. Kako bi znali trenutnu poziciju vrha alata korišten je algoritam direktne kinematike (vidi kod 2..2).

```
def forward_kinematics(self, joint_position):
        #forward kinemtics for kinova Jaco , from given joint positions calculates end effector position
        #DH parametars - fixed for Kinova Jaco
        d1 = 0.2755
        d2 = 0.41
        d3 = 0.2073
        d4 = 0.0741
        d5 = 0.0741
        d6 = 0.16
        e2 = 0.0098

        aa = ((30*math.pi)/180)
        sa = math.sin(aa)
        s2a = math.sin(2*aa)
        d4b = d3 + (sa/s2a) * d4
        d5b = (sa/s2a) * d4 + (sa/s2a) * d5
        d6b = (sa/s2a) * d5 + d6

        alpha = [math.pi/2, math.pi, math.pi/2, 2*aa, 2*aa, math.pi]
        a = [0, d2, 0, 0, 0, 0]
        d = [d1, 0, -e2, -d4b, -d5b, -d6b]
        theta = np.array([-joint_position[0], joint_position[1]-math.pi/2, joint_position[2]+math.pi/2, joint_position[3], joint_position[4]-math.pi, joint_position[5]+math.pi/2])

        transform_matrix = np.eye(4)
```

```
25          for i in range(len(theta)):
26              transform_matrix = np.dot(transform_matrix, self.
    compute_DH_matrix(theta[i], d[i], alpha[i], a[i]))
27          end_tool_position = transform_matrix[0:3, 3]
28          return end_tool_position
```

Listing 2..2: Metoda algoritma direktne kinematike

Direktna kinematika robota izvedena je koristeći parametre navedene na slici 2.10. Slika



Slika 2.10. Kinova Jaco 6 DoF dimenzije [9]

prikazuje dimenzije robotske ruke izražene u milimetrima.

Nakon dobivene konfiguracije, ona se objavljuje na rostopic `kinova_robotType/effort_joint_trajectory_controller/command` (vidi kod 2..3) kako bi pomaknuli robotsku ruku. Navedeni programski kod prikazuje algoritam upravljanja robotskom rukom u simulaciji.

```
1 def moveJoint(self, position):
```

```
        if position.visibility > 0.8:
            #moves robotic arm based on the position derived from detected pose
            #position is an (x,y,z) in  world coordinates from camera

            #transfer the coordinates to robot coordinate system
            next_position = [(1)*position.x,(1)*position.z,(-1)*position.y]
            #get current end effector position
            current_conf = self.joint_positions
            current_position = self.forward_kinematics(current_conf)
            #print(f'current position is {current_position}')

            vdot = next_position - current_position

            qdot = np.linalg.pinv(self.numerical_jacobian(current_conf)).dot(vdot)
            next_conf = current_conf + qdot

            jointCmd = JointTrajectory()
            point = JointTrajectoryPoint()
            jointCmd.header.stamp = rospy.Time.now() + rospy.Duration.from_sec(0.0);
            point.time_from_start = rospy.Duration.from_sec(5.0)
            for i in range(0, self.nbJoints):
                jointCmd.joint_names.append(self.prefix +'_joint_'+str(i+1))
                point.positions.append(next_conf[i])
                point.velocities.append(0)
                point.accelerations.append(0)
                point.effort.append(0)
            jointCmd.points.append(point)
            rate = rospy.Rate(100)

            count = 0
            while (count < 50):
                self.pub.publish(jointCmd)
                count = count + 1
```

```
35                  rate.sleep()
```

Listing 2..3: Metoda algoritma upravljanja robotskom rukom

Zatvaranje i otvaranje šake robotske ruke kontrolira se prema dobivenom rezultatu modela za raspoznavanje gesta šake. Ako je detektirana zatvorena šaka ili prst prema gore, zatvaramo šaku. (vidi kod 2..4).

```
1  def moveFingers(self):
2          if self.hand_gesture == "'Closed_Fist'" or  self.hand_gesture
       == "'Thumb_Up'" :
3              jointcmds = [1,1,1]
4          else :
5              jointcmds = [0,0,0]
6
7          if jointcmds != self.former_finger_jointcmds and self.
       hand_gesture != "'None'":
8              jointCmd = JointTrajectory()
9              point = JointTrajectoryPoint()
10             jointCmd.header.stamp = rospy.Time.now() + rospy.Duration.
       from_sec(0.0);
11             point.time_from_start = rospy.Duration.from_sec(5.0)
12             for i in range(0, self.nbfingers):
13                 jointCmd.joint_names.append(self.prefix +'
       _joint_finger_'+str(i+1))
14                 point.positions.append(jointcmds[i])
15                 point.velocities.append(0)
16                 point.accelerations.append(0)
17                 point.effort.append(0)
18             jointCmd.points.append(point)
19             rate = rospy.Rate(100)
20             count = 0
21             while (count < 500):
22                 self.finger_pub.publish(jointCmd)
23                 count = count + 1
24                 rate.sleep()
25             self.former_finger_jointcmds = jointcmds.copy()
```

Listing 2..4: Metoda algoritma upravljanja otvaranjem/zatvaranjem ruke

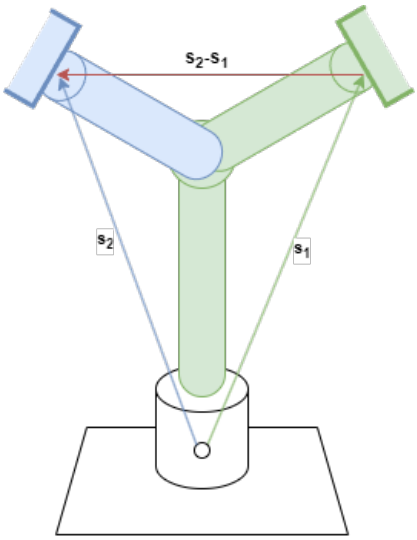
Algoritam upravljanja na stvarnom postavu

Također, unutar kinova_ros paketa, korisniku je dostupno upravljanje robotskom rukom po brzini u koordinatnom sustavu zglobova i u Kartezijskom koordinatnom sustavu. Upravljanje po brzini može biti korisno jer daje glatke i kontinuirane pokrete, bolja je kontrola nad dinamikom i lagana je prilagodljivost izvršavanju zadataka u stvarnom vremenu.

Kako bi ostvarili upravljanje po brzini u koordinatnom sustavu zglobova potrebno je objavljivati poruke tipa JointVelocity na rostopic `kinova_robotType_driver/in/joint_velocity`. Objavljivanje poruke na taj rostopic omogućuje pomicanje zglobova robotske ruke za zadanu vrijednost, izraženu u mjernoj jedinici stupnjevi/sekunda. Upravljanje u koordinatnom sustavu zglobova korisno je ako želimo vrlo detaljno i precizno upravljati svakim zakretom zglobova robota. No, ako želimo samo vrh alata dovesti u željenu točku dovoljno nam je precizno upravljanje po brzini u Kartezijskom koordinatnom sustavu.

Kako bi demonstrirali pomicanje robotske ruke na stvarnom postavu, korišten je ovaj tip upravljanja. Pošto je naš zadatak takav da se robotska ruka mora gibati kontinuirano, nalik ljudskom pokretu, izvedeno je upravljanje robotske ruke po brzini u Kartezijskom koordinatnom sustavu. To znači da upravljamo brzinom promjene pozicije vrha alata robotske ruke u koordinatnom sustavu vrha alata. U okviru ovog diplomskog rada, pošto iz slike kamere dobijemo vrijednost (x, y, z) pozicije ruke čovjeka u Kartezijskom koordinatnom sustavu, radimo upravljanje u tom istom sustavu bez potrebe za znanjem i korištenjem dodatnih algoritama inverzne kinematike. Također, potrebno je transformirati koordinate dobivene s kamere u koordinate robotske ruke (slika 2.9.). Slika 2.11. je ilustrativnog karaktera i prikazuje trenutni položaj robotske ruke, obojen zeleno i sljedeći, željeni, položaj robotske ruke, obojen plavo. Svaki od ta dva položaja možemo zapisati kao vektor pozicije. Na temelju ta dva vektora pozicije, to jest, njihove razlike, dobiven je vektor pozicije iz kojeg računamo potrebni vektor brzine kojeg dalje šaljemo na rostopic naziva `/kinova_robotType_driver/in/cartesian_velocity`.

$$\dot{s} = \frac{s_2 - s_1}{t} \tag{2.11}$$



Slika 2.11. Trenutna i sljedeća pozicija robotske ruke

Vektor brzine $\dot{s}$ dobiven je dijeljenjem razlike vektora pozicije sa vremenom, *time*, u kojem želimo da se navedeni pokret ostvari (2.11) (vidi kod 2..5). Programski kod prikazuje algoritam upravljanja i pomicanja robotske ruke na stvarnom postavu.

```
    def moveJoint(self, position):
        if position.visibility > 0.8:
            #moves robotic arm based on the position derived from
    detected pose
            #position is an (x,y,z) in  world coordinates from camera
            #transfer the coordinates to robot coordinate system
            next_position = [(1)*position.x,(1)*position.z,(-1)*
    position.y]
            #get current end effector position
            current_position = [self.tool_position.x, self.
    tool_position.y, self.tool_position.z]

            vel_msg = PoseVelocity()
            time = 2    #seconds, duration of movement
            #velocity we get
            vel = [(np-cp)/time  for np, cp in zip(next_position,
    current_position)]

            vel_msg.twist_linear_x = vel[0]
            vel_msg.twist_linear_y = vel[1]
            vel_msg.twist_linear_z = vel[2]
            vel_msg.twist_angular_x = 0.0
```

```
            vel_msg.twist_angular_y = 0.0
            vel_msg.twist_angular_z = 0.0

            #we are controling the robot arm tool velocity in
    cartesian space
            #unit is meter/second for linear velocity
            if abs(vel[0])<0.1 and abs(vel[1])<0.1 and abs(vel[2])
    <0.1:
                #stop publishing
                print("Point is reached.")
            else:
                rate = rospy.Rate(100)  #Hz
                count = 0
                while (count < int(time*100)):
                    self.pub.publish(vel_msg)
                    print(f'Published {vel_msg}')
                    count = count + 1
                    rate.sleep()
```

Listing 2..5: Metoda algoritma upravljanja pomicanjem robotske ruke na stvarnom postavu

Analizom ponašanja robotske ruke koristeći raznih vremena izabrano je 2 sekunde kao vrijednost parametra. Ako bi povećali parametar vremena, pokret bi bio sporiji i više kontroliran. To nam je korisno u situacijama kada se obavlja neki delikatan zadatak kada brzina pokreta nije bitna. No, ako ga previše povećamo pokreti se mogu činiti neprirodni i nekontrolirani. Parametar *rate* kontrolira fluidnost pokreta, ako ga smanjimo pokreti su isprekidani no ako ga previše povećamo moguće je opteretiti sustav. Ta vrijednost automatski je postavljena na 100Hz, korištenje veće vrijednosti neće utjecati na brzinu pomicanja [12]. Zadnje, imamo parametar *count* koji govori koliko se puta naša petlja izvršava, limit petlje je postavljen na umnožak *rate* i *time* parametara. Izračunata poruka se objavljuje do kad nije postignuta pozicija. Navedeni algoritam omogućuje pomicanje vrha alata dobivenom linearnom brzinom. Radi jednostavnosti, korištena je samo linearna brzina dok se zakret šake, odnosno kutna brzina zanemaruje.

Kako bi upravljali otvaranjem i zatvaranjem šake robotske ruke na stvarnom postavu koristimo klijent naziva fingers_action_client.py dostupan unutar kinova_ros pa-

keta. On prima tri parametra; tip robota, mjernu jedinicu i vrijednost otvaranja/zatvaranja. Vrijednost 0 ukazuje da je prst skroz otvoren. Ako se iz modela detekcija geste šake detektira zatvorena šaka ili samo jedan prst prema gore (vidi kod 2..6) na vrh robotske ruke šalje se vrijednosti [100, 100, 100] što ukazuju na zatvaranje robotske šake. Nadalje, zadane vrijednosti se šalju na klijenta koji zatvara ili otvara robotsku šaku.

```
35     def gripper_client(self,finger_positions):
36         """Send a gripper goal to the action server."""
37         action_address = '/' + self.prefix + '_driver/fingers_action/
   finger_positions'
38 
39         client = actionlib.SimpleActionClient(action_address,
   SetFingersPositionAction)
40         client.wait_for_server()
41 
42         goal = SetFingersPositionGoal()
43         goal.fingers.finger1 = float(finger_positions[0])
44         goal.fingers.finger2 = float(finger_positions[1])
45         # The MICO arm has only two fingers, but the same action
   definition is used
46         if len(finger_positions) < 3:
47             goal.fingers.finger3 = 0.0
48         else:
49             goal.fingers.finger3 = float(finger_positions[2])
50         client.send_goal(goal)
51         if client.wait_for_result(rospy.Duration(5.0)):
52             return client.get_result()
53         else:
54             client.cancel_all_goals()
55             rospy.logwarn('        the gripper action timed-out')
56             return None
57 
58     def moveFingers(self):
59         if self.hand_gesture == "'Closed_Fist'" or  self.hand_gesture
   == "'Thumb_Up'" :
60             jointcmds = [100,100,100]
61         else :
62             jointcmds = [0,0,0]
63 
```

```
64          if jointcmds != self.former_finger_jointcmds and self.
    hand_gesture != "'None'":
65              self.former_finger_jointcmds = jointcmds.copy()
66              finger_turn = [x/100.0 * self.finger_maxTurn for x in
    jointcmds]
67              positions_temp1 = [max(0.0, n) for n in finger_turn]
68              positions_temp2 = [min(n, self.finger_maxTurn) for n in
    positions_temp1]
69              positions = [float(n) for n in positions_temp2]
70              result = self.gripper_client(positions)
71              return result
```

Listing 2..6: Metode algoritma otvaranja/zatvaranja robotske ruke na stvarnom postavu

3. Rezultati i rasprava

U sljedećem poglavlju prikazani su rezultati testiranja rada predloženog teleoperacijskog okvira. Testiranje se provodi najprije unutar simulacije, koristeći Gazebo Simulator a zatim na stvarnom postavu koristeći Kinova Jaco robotsku ruku.

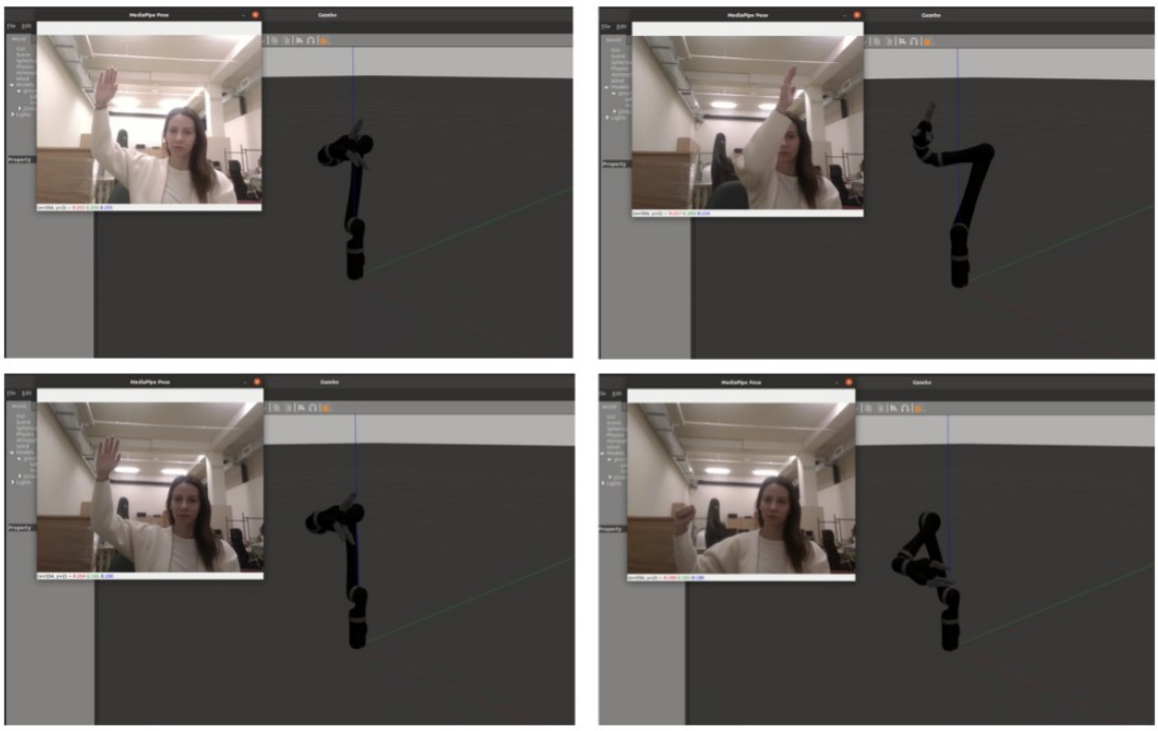
3.1. Simulacija u Gazebo Simulatoru

Kao početak testiranja, izvedeni su testovi u simulacijskom okruženju koristeći Gazebo Simulator. Testiranje u simulaciji omogućuje nam preciznu analizu ponašanja algoritma u kontroliranim uvjetima prije implementacije na stvarni postav. Na slici 3.1. vidimo rezultate testiranja u simulaciji. Na slici u gornjem lijevom kutu je prikazano što kamera vidi i snima. Unutar Gazebo Simulatora vidimo kako se ti podaci interpretiraju.

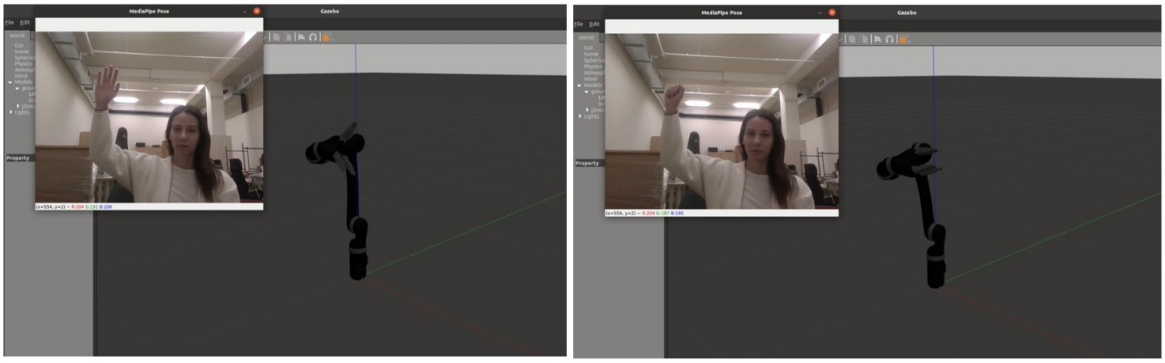
Najprije je testirano pomicanje robotske ruke. Željeno ponašanje robota bilo je definirano kroz nekoliko ključnih koraka: ruka u prvotnom položaju (slika gore lijevo), pomicanje ruke prema lijevo (slika gore desno) i vraćanje ruke u početni položaj (slika dolje lijevo) i pomicanje ruke dolje (slika dolje desno). Svaki od ovih koraka bio je precizno praćen i evaluiran kako bismo osigurali da algoritam ispravno interpretira i izvršava zadane naredbe. Prema rezultatima na slici 3.1. vidimo zadovoljavajuće ponašanje robotske ruke na temelju demonstracije čovjeka.

Zatim je proveden test otvaranja i zatvaranje šake robotske ruke, to jest njezinih prstiju. Na slici 3.2. prikazani su rezultati tog testiranja. Najprije je šaka otvorena te nakon toga čovjek zatvara svoju šaku te vidimo na slici desno da robotska ruka prati ponašanje čovjeka te zatvara vrh alata.

Prikazani rezultati simulacije potvrđuju da je algoritam sposoban izvršiti dane zadatke. Na temelju ovih simulacijskih rezultata, možemo zaključiti da je algoritam spre-



Slika 3.1. Pomicanje robotske ruke unutar simulacije



Slika 3.2. Zatvranje šake robotske ruke unutar simulacije

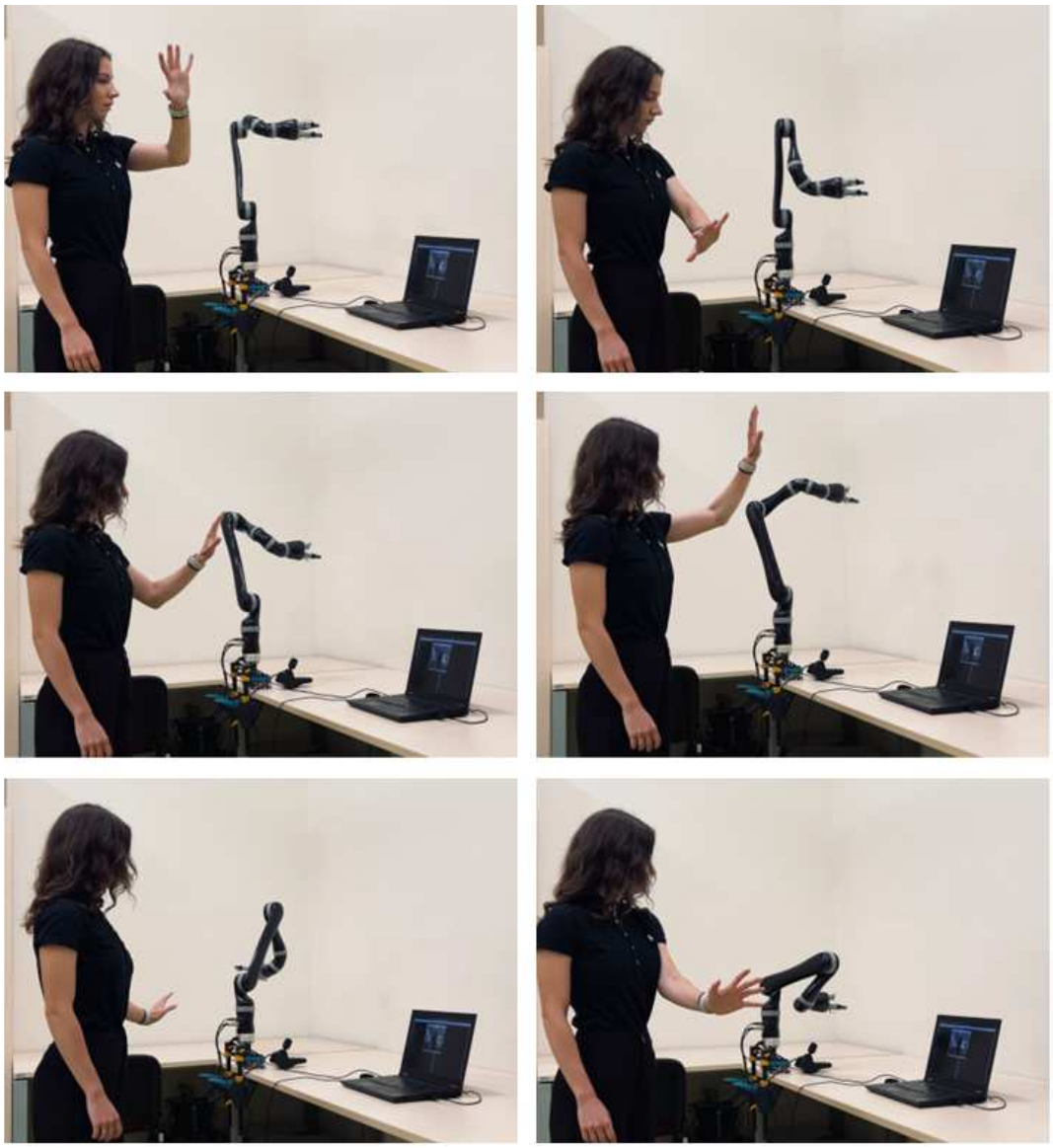
man za sljedeću fazu testiranja u stvarnom okruženju.

3.2. Stvarni postav s Kinova Jaco robotskom rukom

Nakon testiranja u simulaciji, upravljanje robotskom rukom testirano je na stvarnom postavu koristeći Kinova Jaco robotsku ruku.

Kao prvi test, promatrano je gibanje robotske ruke. Željeno gibanje sastojalo se od nekoliko ključnih koraka: pomicanje ruke lijevo ili desno, gore ili dolje, te dijagonalno. Na

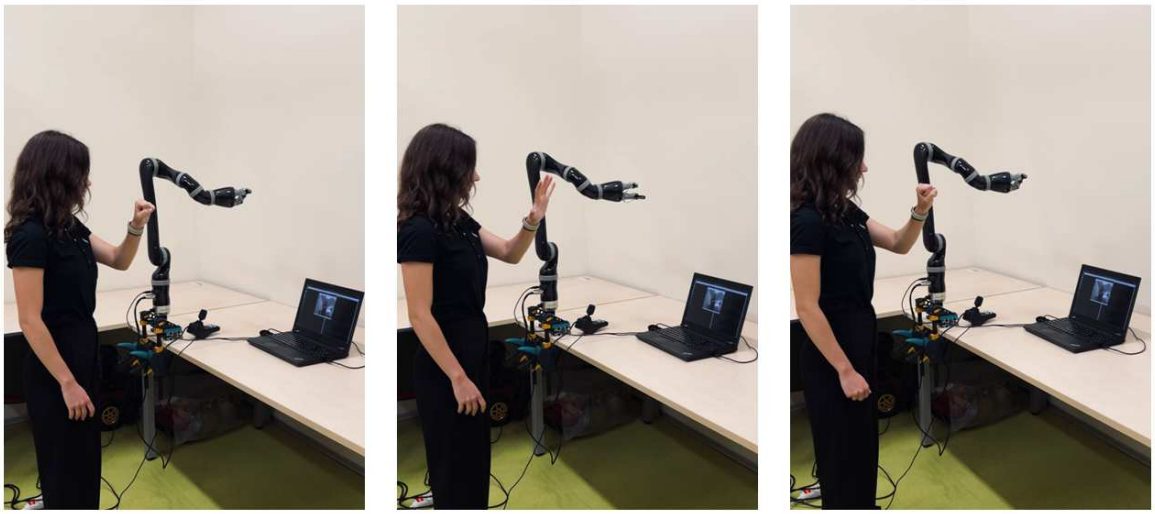
slici 3.3. vidljivi su rezultati ovog testiranja. Na temelju promatranja rada robotske ruke, uočeno je da ona precizno prati željeno ponašanje i giba se u skladu s pokretima čovjekove ruke. Ovi rezultati su izuzetno važni jer potvrđuju sposobnost sustava da replicira ljudske pokrete s zadovoljavajućom točnošću.



Slika 3.3. Gibanje robotske ruke

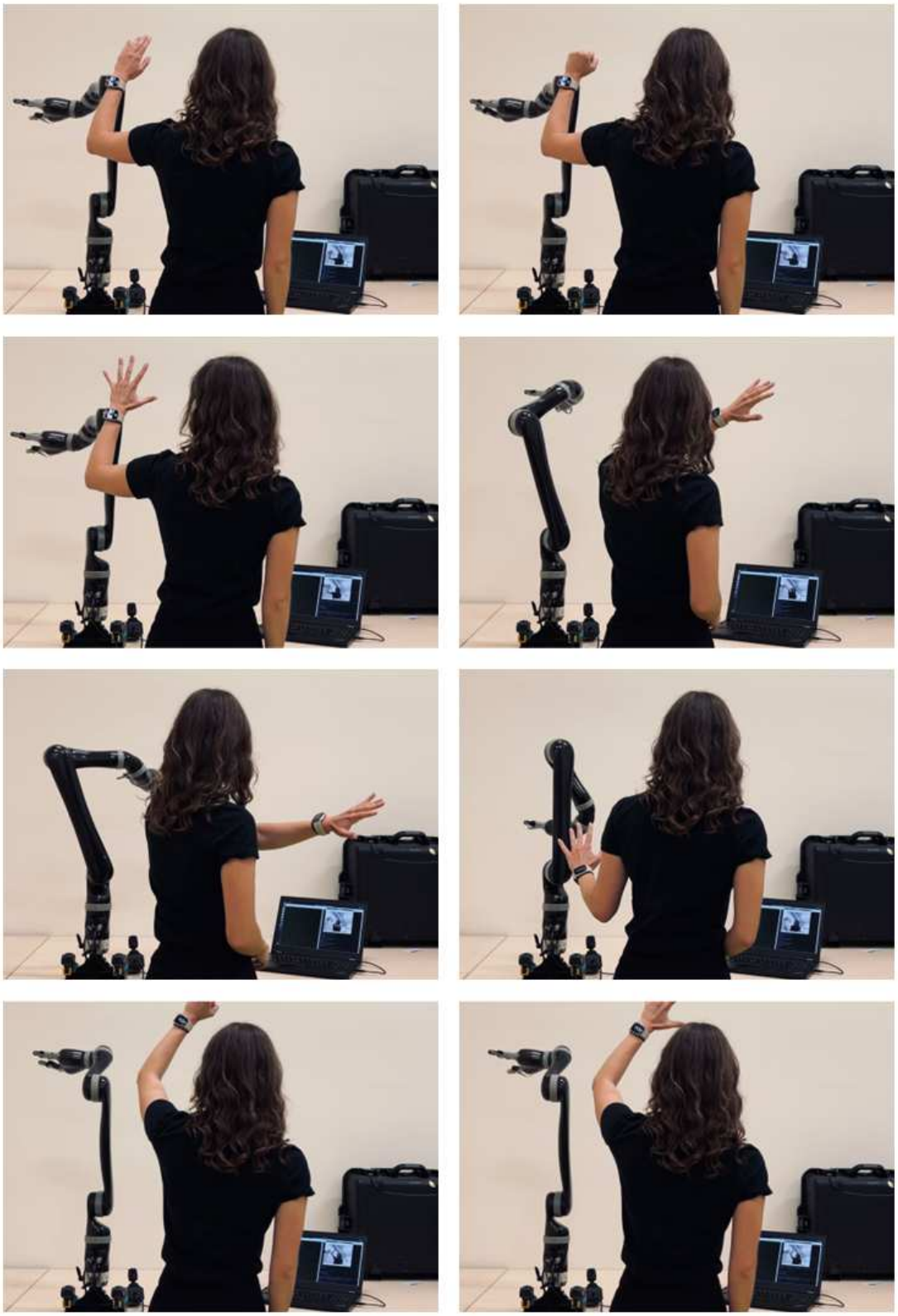
Zatim je testiran algoritam otvaranja i zatvaranja šake robotske ruke. Slika 3.4. prikazuje rezultate ovog testiranja. Na slici lijevo, vidimo zatvorenu šaku čovjeka, pri čemu su i prsti robotske ruke zatvoreni. Zatim se šaka otvara, a prsti robotske ruke prate taj pokret, te na slici skroz desno vidimo ponovo zatvorenu šaku, uz istovremeno zatvaranje prstiju robotske ruke. Ovi rezultati demonstriraju preciznost i sinkronizaciju robotske ruke s ljudskim pokretima, što je ključno za zadatke koji zahtijevaju finu motoričku kontrolu,

poput manipulacije malim objektima ili izvođenja delikatnih operacija.



Slika 3.4. Zatvranje i otvaranje šake robotske ruke na stvarnom postavu

Kao zadnji korak, promatrano je složeno gibanje koje uključuje pomicanje robotske ruke u svim smjerovima te otvaranje i zatvaranje šake. Rezultati ovog testiranja prikazani su na slici 3.5. Vidljivo je kako robotska ruka precizno prati pokrete čovjekove ruke i usklađeno otvara i zatvara šaku. Ova faza testiranja potvrđuje sposobnost sustava da izvede složene zadatke koji kombiniraju više pokreta, čime se dodatno ističe potencijal za primjenu u stvarnim scenarijima, uključujući medicinske postupke, industrijsku automatizaciju i svakodnevne asistivne radnje.



Slika 3.5. Složeno gibanje i otvaranje šake robotske ruke

4. Zaključak

U ovom radu predstavili smo algoritam teleoperacije robotske ruke zasnovan na imitaciji pokreta čovjeka praćenog monokularnom kamerom. Proces počinje snimanjem pokreta ljudske ruke monokularnom kamerom, čime se dobivaju podaci o poziciji šake čovjeka u prostoru. Ti podaci se zatim prenose putem ROS (Robot Operating System) okruženja na robotsku ruku Kinova Jaco. Predloženi teleoperacijski okvir je testiran u simulacijskom okruženju Gazebo Simulator, koristeći algoritme inverzne i direktne kinematike te pripadajuću Jakobijan matricu. Također, testiranje je provedeno i na stvarnom sustavu koristeći robotsku ruku Kinova Jaco i implementirano je upravljanje robotskom rukom po brzini u Kartezijskom koordinatnom sustavu. Korištenjem upravljanja robotskom rukom po brzini u Kartezijskom koordinatnom sustavu postignuto je zadovoljavajuće gibanje robotske ruke nalik čovjeku bez potrebe za poznavanjem algoritama direkte ili inverzne kinematike što ovaj proces čini intuitivan ljudskom operateru.
Dobiveni rezultati iz simualcije i u laboratorijskom okruženju pokazuju da robotska ruka vjerno prati pokrete čovjeka, uključujući otvaranje i zatvaranje šake u skladu s pokretima ljudske ruke. Ovaj rad demonstrira da predloženi algoritam uspješno omogućava intuitivnu kontrolu robotske ruke putem imitacije ljudskih pokreta, što otvara nove mogućnosti za primjene u područjima kao što su rehabilitacija, asistivna robotika i teleoperacija u složenim ili opasnim okruženjima. Također, korištenje samo monokularne kamere osobnog računala doprinosi jednostavnosti upravljanja robotskom rukom i prilagodljivosti bilo kojem okruženju i uređaju. Zaključno, navedenim teleopracijskim okrivom postignuta je sinergija između gibanja robotske ruke i poketa čovjeka praćenog monoukularnom kamerom.

Literatura

[1] I. Petrović, "Upravljanje robotskim sustavima", https://www.fer.unizg.hr/_download/repository/URS_Petrovic_2015-16_Predavanje_01.pdf, 2015., accessed: travanj 2024.

[2] M. Menolotto, D.-S. Komaris, S. Tedesco, B. O'Flynn, i M. Walsh, "Motion capture technology in industrial applications: A systematic review", *Sensors*, sv. 20, br. 19, str. 5687, 2020.

[3] Google Developers, "Pose landmarker", https://developers.google.com/mediapipe/solutions/vision/pose_landmarker, 2024., accessed: travanj 2024.

[4] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, i M. Grundmann, "Blazepose: On-device real-time body pose tracking", *arXiv preprint arXiv:2006.10204*, 2020.

[5] Google Developers, "Gesture recognizer", https://developers.google.com/mediapipe/solutions/vision/gesture_recognizer, 2024., accessed: travanj 2024.

[6] Z. Kovačić, S. Bogdan, i V. Krajči, "Osnove robotike", *Osnove robotike*, 2002.

[7] M. W. Spong, S. Hutchinson, i M. Vidyasagar, "Forward kinematics: the denavit-hartenberg convention", *Robot dynamics and control*, str. 57–82, 2004.

[8] L. Sciavicco i B. Siciliano, *Modelling and control of robot manipulators*. Springer Science & Business Media, 2012.

[9] K. Robotics, "Kinova™ ultra lightweight robotic arm user guide", 2022.

[10] "Why ros?" https://www.ros.org/blog/why-ros/, 2023., accessed: travanj 2024.

[11] M. Quigley, B. Gerkey, i W. D. Smart, *Programming Robots with ROS: a practical introduction to the Robot Operating System*. " O'Reilly Media, Inc.", 2015.

[12] "Kinoa-ros repository", https://github.com/Kinovarobotics/kinova-ros, 2024., accessed: travanj 2024.

[13] "About gazebo", https://classic.gazebosim.org/tutorials, 2024., accessed: travanj 2024.

[14] "Kinoa-gazebo repository", https://github.com/Kinovarobotics/kinova-ros/tree/noetic-devel/kinova_gazebo, 2024., accessed: travanj 2024.

[15] "Numerical differentiation", https://en.wikipedia.org/wiki/Numerical_differentiation, 2024., accessed: svibanj 2024.

Sažetak

Planiranje gibanja robotske ruke imitacijom pokreta čovjeka praćenog monokularnom kamerom

Ana Jambrešić

Razvoj sposobnosti robota da imitira ljudske pokrete znatno je transformirao industriju a tako i svakodnevni život. Ovaj diplomski rad fokusira se na razvoj sustava za praćenje pokreta čovjeka monokularnom kamerom te pretvorba tog pokreta u digitalni oblik i izvršavanje na robotskoj ruci Kinova Jaco. Korištenje samo monokularne kamere uklanja potrebu za složenom i skupom kamerom te čini proces teleoperacije intuitivnim i pristupačnim. U radu se primjenjuju algoritmi inverzne, direktne i diferencijalne kinematike za upravljanje robotskom rukom te model neuronske mreže otvorenog koda za detekciju i analizu pokreta ruku čovjeka. Integracija s ROS sustavom omogućava efikasno prenošenje ljudskog pokreta na robotsku ruku u stvarnom vremenu. Testiranje sustava je provedeno kroz simulacije u Gazebo okruženju te praktičnim eksperimentima s Kinova Jaco robotskom rukom.

Ključne riječi: teleoperacija; direktna kinematika; inverzna kinematika; diferencijalna kinematika; MediaPipe; Kinova Jaco; ROS

Abstract

Robot arm motion planning by imitating human motion tracked with a monocular camera

Ana Jambrešić

The development of robots capabilities to imitate human movements has significantly transformed the industry as well as everyday life. This thesis focuses on developing a system for tracking human movements using a monocular camera, converting these movements into digital form, and executing them on the Kinova Jaco robotic arm. The use of only a monocular camera eliminates the need for complex and expensive equipment, making the teleoperation process intuitive and accessible. The study applies algorithms for inverse, forward, and differential kinematics for controlling the robotic arm and an open-source neural network model for detecting and analyzing human hand movements. Integration with the ROS system enables efficient transfer of human motion to the robotic arm in real-time. System testing was conducted through simulations in the Gazebo environment and practical experiments with the Kinova Jaco robotic arm.

Keywords: teleoperation; forward kinematics; inverse kinematics; differential kinematics; MediaPipe; Kinova Jaco; ROS