Usklađeno rukovanje objektima pomoću dva robota korištenjem senzora sile i momenata

Pažanin, Ivan

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRANJE

DIPLOMSKI RAD

Ivan Pažanin

Zagreb, 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Prof. dr. sc. Bojan Jerbić

Student:

Ivan Pažanin

Zagreb, 2018.

Izjavljujem da sam ovaj rad izradio samostalno koristeći stečena znanja tijekom studija i navedenu literaturu.

Ivan Pažanin

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske radove studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment,
inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa:	
Ur. broj:	

DIPLOMSKI ZADATAK

Student: **IVAN PAŽANIN** Mat. br.: 0035169411

Naslov rada na hrvatskom jeziku: **Usklađeno rukovanje objektima pomoću dva robota korištenjem senzora sile i momenata**

Naslov rada na engleskom jeziku: **Coordinated handling of objects by two robots using force and torque sensor**

Opis zadatka:

Suradnju dvaju robota, koji međusobno nisu povezani klasičnim komunikacijskim sklopovljem, moguće je ostvariti putem međusobne fizičke interakcije. Tema ovog rada je interakcija dvaju robota koji moraju zajednički nositi predmet rada usklađujući se samo putem signala koje dobivaju s ugrađenih senzora sile i momenata. U sklopu rada potrebno je razviti odgovarajući upravljački program koji na osnovi podataka sa 6-osnog senzora omogućuje koordinaciju suradničkih robota u dvoručnoj konfiguraciji. Pri tome jedan od robota inicira kretanje, dok drugi robot temeljem podataka sa senzora prilagođava kretanje kako bi se obavio zadatak rukovanja zadanim objektom.

Razvijeni model potrebno je provesti i provjeriti na laboratorijskom modelu.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:
03. svibnja 2018.

Rok predaje rada:
05. srpnja 2018.

Predviđeni datum obrane:
11. srpnja 2018.
12. srpnja 2018.
13. srpnja 2018.

Zadatak zadao:

prof. dr. sc. Bojan Jerbić

Predsjednica Povjerenstva:

prof. dr. sc. Biserka Runje

SADRŽAJ

SADRŽAJ I
POPIS SLIKA II
POPIS TABLICA III
POPIS OZNAKA IV
SAŽETAK V
SUMMARY VI
1. Uvod 1
2. Kinematika robota 3
2.1. Matrice linearnih transformacija 4
2.2. Inverzna kinematika 9
2.3. Jacobijeva matrica 13
2.4. Inverzna kinematika UR5 robota [12] 15
2.5. Jacobijeva matrica UR5 robota 26
3. Postava tehničkog sustava 27
3.1. Universal Robots industrijski roboti 27
3.2. Robotiq FT 150 senzor sila i momenata [8] 30
3.2.1. Instalacija senzora 31
3.2.2. Kalibracija senzora 34
3.2.3. Niskopropusni filter (eng. lowpass) 35
3.3. Radna okolina 38
4. Programsko upravljanje UR5 robotom 40
4.1. Komunikacija s robotom 40
4.2. Upravljanje pozicijom ili upravljanje brzinom 43
5. Distribuirano upravljanje dvoručnim robotskim rukovanjem 46
5.1. PID regulacija 47
5.2. Primjena dvoručnog rukovanja 49
6. Zaključak 54
LITERATURA 55
PRILOZI 56

POPIS SLIKA

Slika 1.1. Dva manipulatora pridržavaju predmet guranjem 2
Slika 2.1. Trosegmentni robot s tri revolutna zgloba (3R) 3
Slika 2.2. Prikaz dvaju mogućih rješenja inverzne kinematike 3R robota 4
Slika 2.3. Transformacija koordinatnog sustava 5
Slika 2.4. Prikaz odnosa između dva segmenta [5] 6
Slika 2.5. Pravila postavljanja prvog (a) i zadnjeg (b) koordinatnog sustava 7
Slika 2.6. Slučaj kada z osi nisu koplanarne 7
Slika 2.7. Slučaj kada su z osi paralelne 8
Slika 2.8. Slučaj kada se z osi presijecaju 8
Slika 2.9. Primjer obilježavanja koordinatnih osi D-H konvencijom 8
Slika 2.10. Usporedba $\mathrm{atan}(x)$ i $\mathrm{atan2}(y,x)$ trigonometrijskih funkcija 10
Slika 2.11. Zadana orijentacija robota 11
Slika 2.12. Primjer dvostrukog rješenja za kut $\theta 1$ 12
Slika 2.13. Perspektiva koordinatnog sustava $\{x1, y1\}$ 12
Slika 2.14. Universal Robots UR5 robotska ruka 16
Slika 2.15. Pojednostavljeni prikaz UR5 robota 16
Slika 2.16. Dodjeljivanje koordinatnih osi 17
Slika 2.17. UR5 robot opisan D-H konvencijom 17
Slika 2.18. Pronalaženje ishodišta $\{x5, z5\}$ koordinatnog sustava 18
Slika 2.19. Rame lijevo ili rame desno 19
Slika 2.20. Posebni uvijet 20
Slika 2.21. Nalaženje kuta $\theta 5$ 20
Slika 2.22. Nalaženje kuta $\theta 6$ 21
Slika 2.23. Nalaženje $\theta 2, \theta 3$ 22
Slika 2.24. Osam mogućih rješenja inverzne kinematike robota 25
Slika 3.1. Dvoručno rukovanje predmetom 27
Slika 3.2. Postava robota proizvođača Universal Robots 28
Slika 3.3. Klasični industrijski roboti 29
Slika 3.4. Privjesak za učenje 29
Slika 3.5. Šesteroosni senzor sila i momenata FT 150 30
Slika 3.6. Spajanje FT 150 senzora u računalo robota 31
Slika 3.7. Dijagram ožičenja senzora 31
Slika 3.8. Instalirani URCaps paket unutar PolyScope sučelja 33
Slika 3.9. Funkcija snimanja puta 33
Slika 3.10. Hardverska instalacija senzora 34
Slika 3.11. Kalibracija senzora putem eksternog računala (laptopa) 35
Slika 3.12. Uzorkovanje senzora pri 100 Hz u trajanju od 10 sekundi 35
Slika 3.13. Karakteristika niskopropusnog filtera prvog reda 37
Slika 3.14. Primjena niskopropusnog filtera 38
Slika 3.15. VODAVODA [13] 38
Slika 3.16. Radna okolina 39
Slika 3.17. Prihvat predmeta 39
Slika 4.1. Program na privjesku za učenje (robot kao klijent) [14] 41
Slika 4.2. Funkcije za čitanje stanja senzora [11] 42
Slika 4.3. Uporaba speedl naredbe 44
Slika 5.1. Prihvat predmeta 46
Slika 5.2. Shema PID regulacije [16] 47
Slika 5.3. Program UR3 robota 50
Slika 5.4. Prikaz očitane sile i odziva PID regulatora za z os 50
Slika 5.5. Primjer rukovanja uz PID regulaciju 53

POPIS TABLICA

Tablica 3.1. D-H parametri trosegmentnog robota 9
Tablica 3.2. Članovi za računanje Jacobijeve matrice 14
Tablica 4.1. D-H parametri UR5 robota 18
Tablica 2.1. Usporedba karakteristika UR robota 28
Tablica 2.2. Karakteristike FT 150 senzora 30
Tablica 2.3. Kompatibilnost URCaps platforme s verzijama UR računala 32
Tablica 5.1. Utjecaj parametara PID regulatora na sustav [17] 48
Tablica 5.2. Parametri PID regulatora 49

POPIS OZNAKA

a		parametar filtra
a_i	m	duljina segmenta
D		broj dimenzija prostora
d_i	m	duljina segmenta
e		greška između referentne i izlazne veličine
$\mathbf{F}_S$		vektor senzorom očitanih sila
F_i	N	sila
f_b	Hz	širina frekvencijskog pojasa filtra
g	$\mathrm{m/s^2}$	gravitacijsko ubrzanje
$\mathrm{H(s)}$		prijenosna funkcija sustava
$\mathbf{J}$		Jacobijeva matrica
K_d		derivacijsko pojačanje
K_i		integralno pojačanje
K_p		proporcionalno pojačanje
k_S		faktor pretvorbe sile
k_M		faktor pretvorbe momenta
l_i, L_i	m	duljina članka
$\mathbf{M}_S$		vektor senzorom očitanih momenata
M	Nm	moment
m	kg	masa
N, n		broj zglobova
$\mathbf{P}_j^i$		vektor translacije
p_x, p_y, p_z		položaj robota
$\mathbf{R}_j^i$		matrica rotacije
R_x, R_y, R_z		orijentacija robota
r		referentna (željena) veličina
$\mathbf{r}$		vektor vanjskih koordinata
$\mathbf{q}$		vektor upravljanih koordinata
s		Laplaceov operator
$\mathbf{T}_j^i$		matrica transformacije
T_f	s	vremenska konstanta filtera
T_s	s	vrijeme uzorkovanja
t	s	vrijeme
t_k		sadašnji trenutak vremena
t_{k-1}		bivši trenutak vremena
$\mathrm{U(s)}$,$\mathrm{u(t)}$		ulaz sustava
X_c, Y_c, Z_c		koordinate centra alata
$\mathrm{Y(s)}$,$\mathrm{y(t)}$		izlaz sustava
α_i	°	zakrivljenost segmenta
θ_i	°	kut zakreta članka
α, γ, ϕ	°	kut
ω_b	rad/s	širina frekvencijskog pojasa filtra
ω	rad/s	kutna brzina

SAŽETAK

U ovom radu realizirano je dvoručno usklađeno rukovanje predmetom u prostoru pomoću šesteroosnog senzora sila i momenata. Korištene su robotske ruke UR3 i UR5 kompanije Universal Robots tako da manja UR3 ruka, upravljana programom sa privjeska za učenje, zadaje trajektoriju, a veća UR5 ruka, uz pomoć filtriranog signala senzora i upravljana programski na daljinu preko TCP protokola, pridržava predmet i prati trajektoriju manje ruke. Pridržavanje i praćenje trajektorije ostvaruje se proporcionalno-integralno-derivacijskom (PID) regulacijom brzina zakreta pojedinih zglobova UR5 robota uz upotrebu inverzne Jacobijeve matrice.

Koristeći *Denavit-Hartenberg* konvenciju, analizirana je geometrija robota i razrađen kinematički model, opisan je postupak nalaženja direktne i inverzne kinematike robota, kao i Jacobijeve i inverzne Jacobijeve matrice.

SUMMARY

Using a six-axis force torque sensor, a coordinated two-arm spatial object handling method is realized in this paper. The two industrial robot arms used, UR3 and UR5 from Universal Robots, are used such that the smaller UR3 arm, being controlled by its teach pendant, sets the trajectory, while the larger UR5 arm, using a filtered sensor signal and being controlled by a remote computer over TCP, holds the object in place and follows the trajectory of the smaller arm. Holding of the object and trajectory following is done with proportional-integral-derivative (PID) control of individual angular joint speeds of the UR5 arm using an inverse Jacobian matrix.

The geometry of the robot is analyzed and a kinematic model is worked out using the Denavit-Hartenberg convention, the process of finding the direct and inverse kinematics, as well as finding the Jacobian and inverse Jacobian matrix, is elaborated.

1. Uvod

Najranijeg poznatog industrijskog robota dizajnirao je Bill Griffith P. Taylor većinom koristeći dijelove edukativnog kompleta za izradu modela zvanog *Meccano* 1937. godine, a njegov rad objavljen je u časopisu *Meccano Magazine* 1938. godine. Unatoč kompleksnosti i inovativnosti, njegov trogodišnji projekt ostao je slabo poznat [1]. Prvi patent za industrijskog robota prijavio je George Devol 1954. godine (koji je odobren 1961. godine) i u suradnji sa Josephom F. Engelbergerom 1958. izradio je prvog robota. Zajedno su osnovali kompaniju *Unimation* i 1961. godine prodali svog prvog komercijalnog industrijskog robota kompaniji *General Motors*, a ubrzo nakon kompanijama *Chrysler* i *Ford Motor Company* [2]. Uvođenje robota ubrzo je i zauvijek promijenilo automobilsku i proizvodnu industriju.

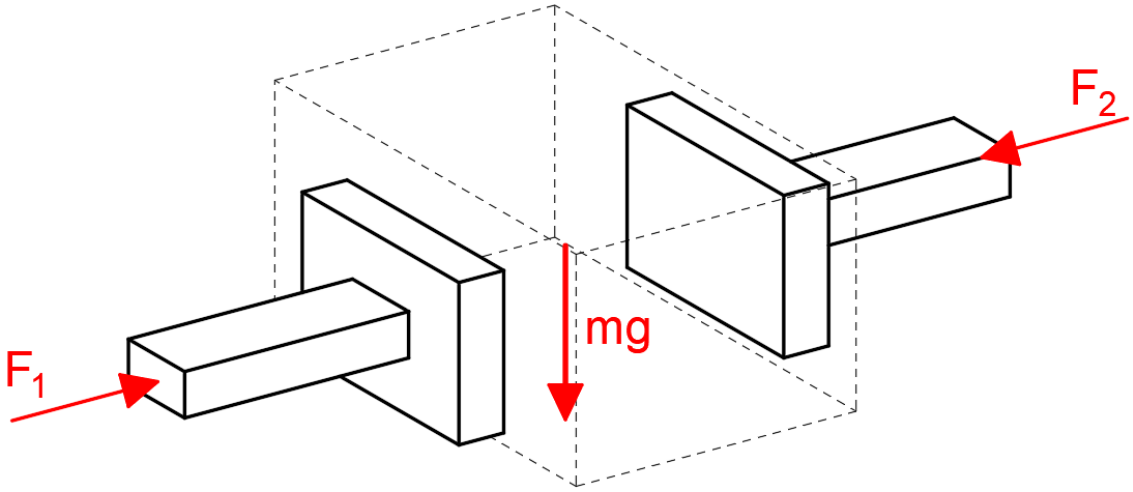
Roboti su danas zastupljeni u mnogim industrijama, a ponajviše u automobilskoj industriji [3]. Općenito, roboti se mogu grupirati u četiri skupine [4]:

1. roboti za opsluživanje – umeću, vade i prenose obratke i alate, otkivke, odljevke i kokile, limove; stavljaju umetke u kalupe prije lijevanja ili ubrizgavanja,
2. tehnološki roboti – opremljeni su alatkama kao što su kliješta za zavarivanje, pištoljima za raspršivanje, brusilicama,
3. montažni roboti – najizazovnije su područje primjene robota jer su uz brzinu rada potrebni i točnost, ponovljivost, različitost pokreta i sofisticirane hvataljke,
4. mjerni roboti – najčešći su takozvani trokoordinatni mjerni uređaji, umjesto hvataljke imaju mjernu glavu koja preko ticala sa safirnom glavom ostvaruje dodir s mjerenim predmetom.

Većina zadataka u industriji može se obavljati jednom robotskom rukom gdje su predmet obrade ili montažna stanica nepomični ili su na pokretnoj traci za vrijeme obavljanja tehnološke obrade ili montaže (npr. vrata automobila stoje na postolju za vrijeme lakiranja). Korištenjem dvoručnih i višeručnih sustava omogućuje se montaža ili obrada predmeta kompliciranijih geometrija na način da jedna ili više ruku pridržava i orijentira predmet obrade dok drugi tehnološki robot izvršava obradu (npr. kontinuirano zavarivanje karoserije automobila). Višeručni sustavi također imaju prednost nad jednoručnim pri nošenju velikih ili teških predmeta i predmeta bez jasno definiranog prihvata (npr. kutije raznih veličina).

Cilj ovog rada je razraditi jedan takav sustav gdje se uz pomoć dva koordinirana industrijska robota rukuje predmetom, ali na način da roboti međusobno ne komuniciraju. Jedan robot zadaje

trajektoriju koju drugi robot treba pratiti koristeći očitanja sa senzora sila i momenata. Korišteni su industrijski roboti UR3 i UR5 proizvođača *Universal Robots* u kombinaciji sa FT 150 senzorom sila i momenata proizvođača *Robotiq*, a rukovanje predmetom ostvareno je guranjem predmeta s dvije strane kao što je prikazano slikom 1.1.



Slika 1.1. Dva manipulatora pridržavaju predmet guranjem

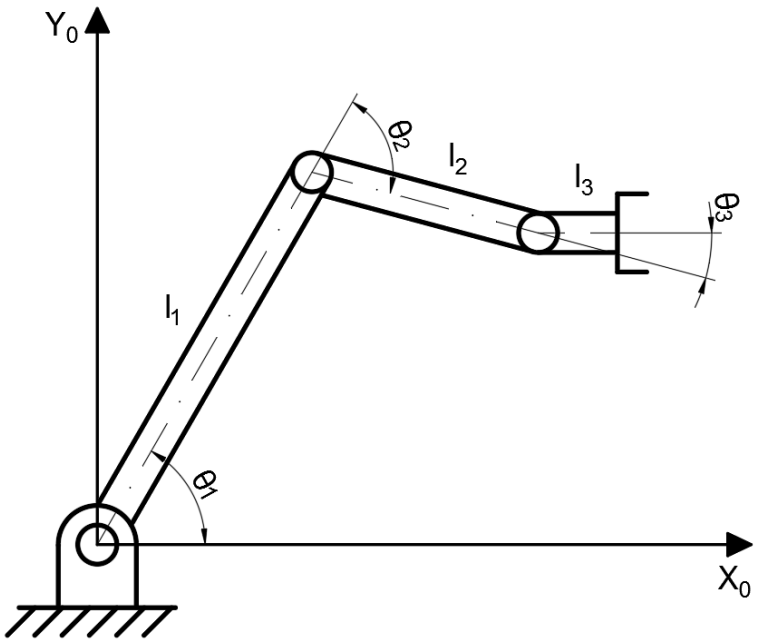
Robot se može modelirati kao lanac krutih tijela (članaka) međusobno povezanih zglobovima gdje se na početku lanca nalazi nepomična baza robota, a na kraju lanca prihvatnica robota. Da bi se robot kretao u trodimenzionalnom prostoru potrebno je upravljati pozicijom i orijentacijom prihvatnice. Zbog toga je neophodno odrediti vezu između parametara zglobova robota te pozicije i orijentacije prihvatnice.

U daljnjem tekstu opisana je kinematička analiza robota i, u svrhu računalnog upravljanja, izveden je kinematički model robota koji senzorom prati trajektoriju predmeta. Nadalje, detaljnije je opisana problematika i navedeni su sastavni dijelovi sustava i opisane su njihove značajke. Prikazan je postupak ožičenja, kalibracije i uporabe senzora, a opisana je i komunikacija između eksternog računala i robota. Razmotreno je nekoliko pristupa upravljanju robotom i elaboriran je konačni izbor. Na kraju je objašnjeno djelovanje i implementacija PID regulacije te su prikazani rezultati primijenjenog rješenja.

2. Kinematika robota

Kinematika je područje znanosti koje se odnosi na kretanje subjekta neovisno o silama koje utječu njegovo kretanje. U sklopu kinematike razmatraju se pozicija, brzina, akceleracija kao i sve derivacije pozicijskih varijabli (u odnosu na vrijeme ili druge varijable). Kinematika robota važna je kod programskog upravljanja robotom, odnosno poznavanja pozicije, brzine ili akceleracije pojedinih segmenata robota u vremenu.

Svaki robot sastoji se od određenog broja segmenata (eng. *link*) koji pomoću raznih vrsta zglobova (prizmatični, revolutni, sferni, vijčani, itd.) dovode vrh robota, ili prihvatnicu (eng. *end effector*), u određeni položaj. Robot prikazan slikom 2.1. ograničen je na kretanje u $\{x_0, y_0\}$ ravnini, a sastoji se od triju segmenata duljina l_1, l_2 i l_3, segmenti su povezani revolutnim zglobovima i zakrenuti kutovima θ_1, θ_2 i θ_3.



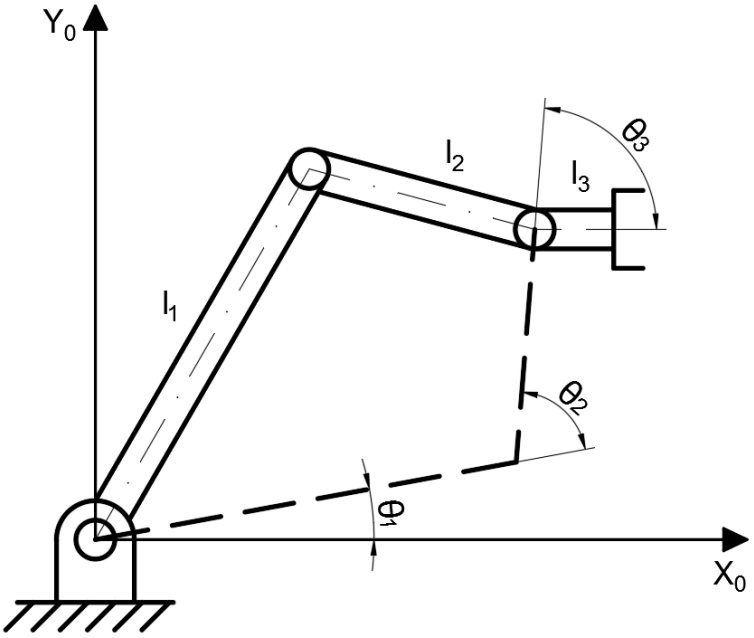
Slika 2.1. Trosegmentni robot s tri revolutna zgloba (3R)

Ako su kutovi svih zglobova poznati (u ovom slučaju θ_1, θ_2 i θ_3), moguće je pomoću jednostavnih trigonometrijskih funkcija izračunati poziciju i orijentaciju prihvatnice robota, to je takozvani direktni kinematički problem (eng. *forward kinematics*). Rješenje direktnog kinematičkog problema je jednoznačno, odnosno postoji samo jedna moguća pozicija i orijentacija prihvatnice sa zadanim kutovima zakreta zglobova.

Direktna kinematika u praksi se rijetko primjenjuje jer operater ili programer uglavnom ne podešava kutove zglobova robota pojedinačno. Prihvatnica robota, s odgovarajućim alatom, obavlja glavnu funkciju robota (zavarivanje, *pick and place*, montaža, itd.), stoga je direktno

upravljanje njenom pozicijom i orijentacijom puno korisnije. Kutovi zakreta zglobova u tom slučaju nisu poznati i potrebno ih je računski odrediti unazadno metodom inverzne kinematike (eng. *inverse kinematics*).

Ukoliko robot posjeduje više zglobova (ili stupnjeva slobode gibanja) nego dimenzija prostora u kojemu mu je ograničeno gibanje ($N = 3 > D = 2$), za robot se smatra da je kinematički redundantan. Znači da postoji više od jednog rješenja za inverzni kinematički problem gdje je jedino poznata pozicija i orijentacija prihvatnice (Slika 2.2).



Slika 2.2. Prikaz dvaju mogućih rješenja inverzne kinematike 3R robota

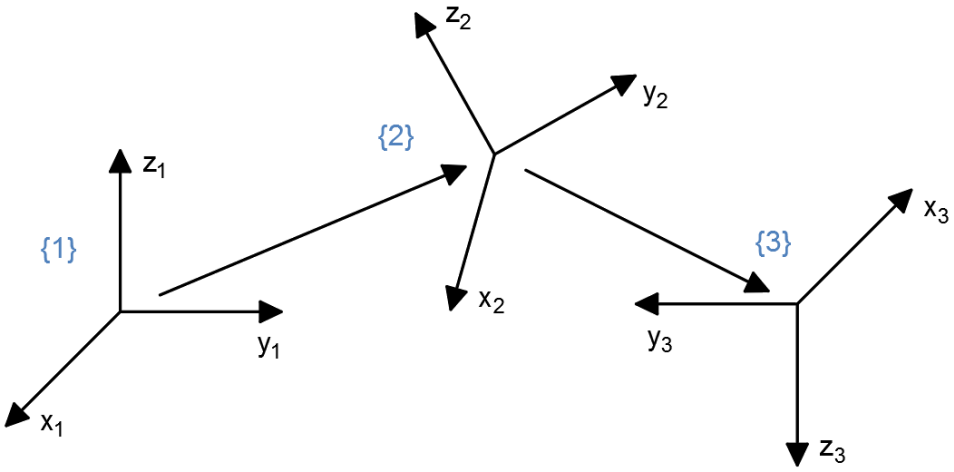
2.1. Matrice linearnih transformacija

Radi računanja direktne ili inverzne kinematike robota potrebno je postaviti koordinatni sustav za svaki segment robota. Za prijelaz linearnim transformacijama iz jednog koordinatnog sustava u drugi koriste se matrice transformacija **T**, a zapisuju se prema relaciji:

$$\mathbf{T}_j^i = \begin{bmatrix} \mathbf{R}_j^i & \vec{\mathbf{P}}_j^i \\ 0 & 1 \end{bmatrix} = \mathbf{T}_j^i = \begin{bmatrix} x_x & y_x & z_x & (P_j^i)_x \\ x_y & y_y & z_y & (P_j^i)_y \\ x_z & y_z & z_z & (P_j^i)_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \textbf{(2.1)}$$

gdje svaki stupac $3x3$ matrice rotacije $\mathbf{R}_j^i$ opisuje preslikavanje osi koordinatnog sustava j na os koordinatnog sustava i, a $\vec{\mathbf{P}}_j^i$ predstavlja $3x1$ vektor translacije iz koordinatnog sustava i u sustav j. Matrice transformacija mogu se lančati množenjem, tako da se transformacija iz nekog koordinatnog sustava {3} u neki koordinatni sustav {1} (Slika 2.3) može izraziti na sljedeći način:

$$\mathbf{T}_3^1 = \mathbf{T}_2^1 \mathbf{T}_3^2. \quad \textbf{(2.2)}$$



Slika 2.3. Transformacija koordinatnog sustava

Pojedinačne rotacije koordinatnog sustava oko svake od triju osi (x, y i z) za neki kut θ definirane su prema(2.3), a translacije u smjerovima triju osi prema (2.4):

$$\begin{aligned}
\mathrm{Rot}_{x,\theta} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_\theta & -s_\theta & 0 \\ 0 & s_\theta & c_\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\
\mathrm{Rot}_{y,\theta} &= \begin{bmatrix} c_\theta & 0 & s_\theta & 0 \\ 0 & 1 & 0 & 0 \\ -s_\theta & 0 & c_\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\
\mathrm{Rot}_{z,\theta} &= \begin{bmatrix} c_\theta & -s_\theta & 0 & 0 \\ s_\theta & c_\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},
\end{aligned} \tag{2.3}$$

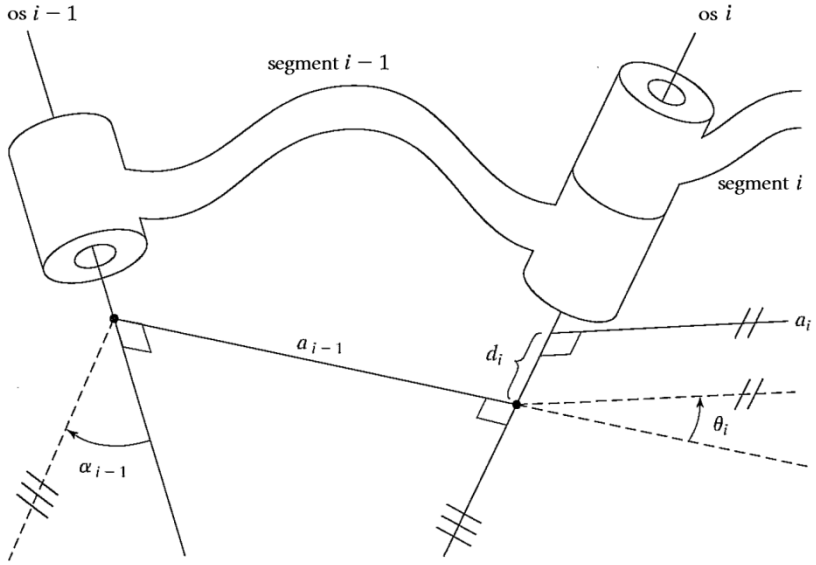
$$\mathrm{Trans}_{x,y,z} = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.4}$$

gdje c_θ i s_θ predstavljaju kraći zapis za $\cos(\theta)$ i $\sin(\theta)$, odnosno. Ukoliko je jedan koordinatni sustav $\{2\}$ zakrenut za neki kut θ oko osi z i translatiran za neku udaljenost y i z u odnosu na početni koordinatni sustav $\{1\}$, tada matrica transformacija iz $\{2\}$ u $\{1\}$ glasi:

$$\begin{aligned}
\mathbf{T}_2^1 = \mathrm{Rot}_{z,\theta}\mathrm{Trans}_{0,y,z} &= \begin{bmatrix} c_\theta & -s_\theta & 0 & 0 \\ s_\theta & c_\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_\theta & -s_\theta & 0 & -s_\theta y \\ s_\theta & c_\theta & 0 & c_\theta y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}.
\end{aligned} \tag{2.5}$$

Vidljivo je da je za svaku linearnu transformaciju potrebno poznavati šest parametara, tri kuta rotacije **R** i tri udaljenosti translacije **P** što postaje zahtjevno za kompleksnije konfiguracije robota.

Koristeći standardiziranu D-H konvenciju obilježavanja koordinatnih sustava robota (eng. *Denavit-Hartenberg convention*) broj parametara može se svesti na samo četiri (Slika 2.4).



Slika 2.4. Prikaz odnosa između dva segmenta [5]

Parametri koji opisuju odnos između segmenata i i $i-1$ su:

a_i – duljina segmenta, predstavlja udaljenost između osi z_{i-1} i z_i po x_i osi
α_i – zakrivljenosti segmenta, predstavlja kut između osi z_{i-1} i z_i po x_i osi
d_i – odmak segmenata, predstavlja udaljenost između osi x_{i-1} i x_i po z_{i-1} osi
θ_i – zakret zgloba, predstavlja kut između osi x_{i-1} i x_i po z_{i-1} osi

Parametri duljine i zakrivljenosti segmenata (a_i, α_i) ovise o konfiguraciji robota i nepromijenjivi su neovisno o gibanju robota tokom rada. Parametri odmaka i zakreta (d_i, θ_i) mogu biti promjenjivi ovisno o vrsti aktuiranog zgloba, prizmatični zglob utječe na vrijednost parametra d_i, dok revolutni utječe na vrijednost parametra θ_i. Svaka transformacija koordinatnog sustava može se predstaviti kao produkt četiriju osnovnih transformacija:

$$\mathbf{T}_i^{i-1} = \mathrm{Rot}_{z,\theta_i}\mathrm{Trans}_{z,d_i}\mathrm{Trans}_{x,a_i}\mathrm{Rot}_{x,\alpha_i}$$
$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \textbf{(2.6)}$$

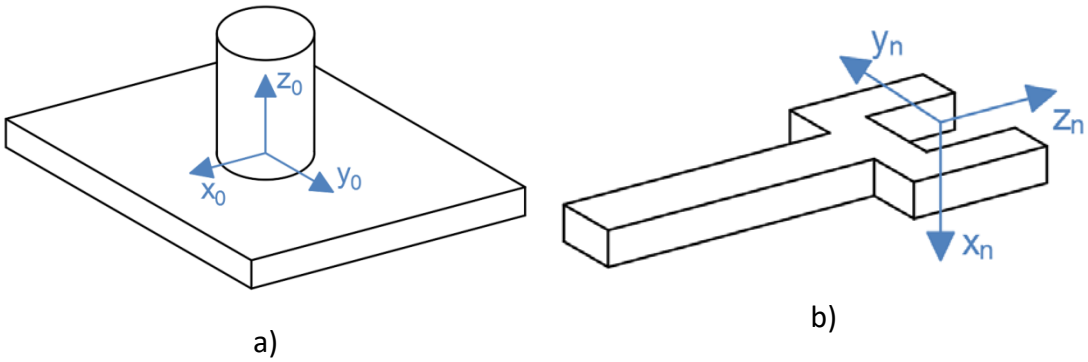
$$\mathbf{T}_i^{i-1} = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad \textbf{(2.7)}$$

Korištenje D-H parametara pretpostavlja poštivanje sljedećih pravila kod određivanja koordinatnih sustava robota [6]:

- koordinatni sustav postavlja se na spojište segmenata robota, odnosno na lokaciju zgloba, na način da je z_i os pozitivno usmjerena u pozitivnom smjeru translacije ili rotacije zgloba i (poštivajući pravilo desne ruke),
- os x_i mora presjecati i biti okomita na os z_{i-1},
- smjer osi y_i određuje se pravilom desne ruke s obzirom na osi x_i i z_i.

Prvi, ili bazni, koordinatni sustav robota označava se kao nulti i obično se postavlja na neku korisnu referentnu točku, kao što je vrh postolja na kojemu se robot nalazi. Orijentacija osi x_0 i y_0 proizvoljna je dok god je zadovoljeno pravilo desne ruke (Slika 2.5.a).

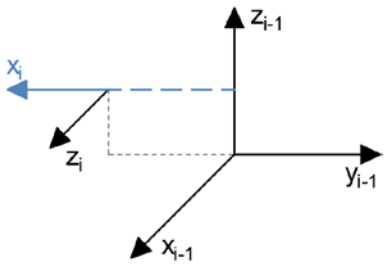
Zadnji, ili n-ti, koordinatni sustav za robot sa n zglobova postavlja se na vrh prihvatnice, gdje je os z_n usmjerena u smjeru prilaza alata, os y_n usmjerena je u pravcu klizanja alata, a os x_n poštuje pravilo desne ruke (Slika 2.5.b).



Slika 2.5. Pravila postavljanja prvog (a) i zadnjeg (b) koordinatnog sustava

Pri izboru x_i osi treba obratiti pažnju na tri moguća slučaja kada:

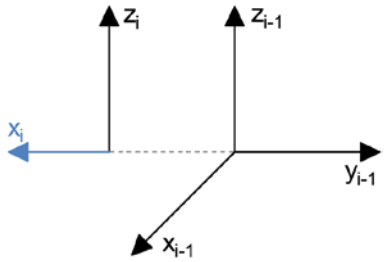
- z_{i-1} i z_i osi nisu koplanarne i ne presijecaju se, x_i os tada jedino može presijecati obje z osi kako je prikazano slikom 2.6., ishodište koordinatnog sustava O_i nalazi se na presjecištu x_i i z_i osi,



Slika 2.6. Slučaj kada z osi nisu koplanarne

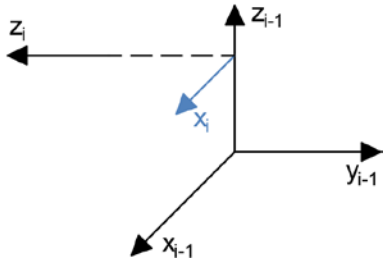
- z_{i-1} i z_i osi su paralelne. Kada su z osi paralelne postoji beskonačno puno x_i osi koje presijecaju obje z osi, poželjno je tada da x_i os prolazi kroz ishodište O_{i-1} jer će onda jedan

od parametara biti $d_i = 0$ (Slika 2.7). U slučaju kada su z osi paralelne također uvijek vrijedi $\alpha_i = 0$,



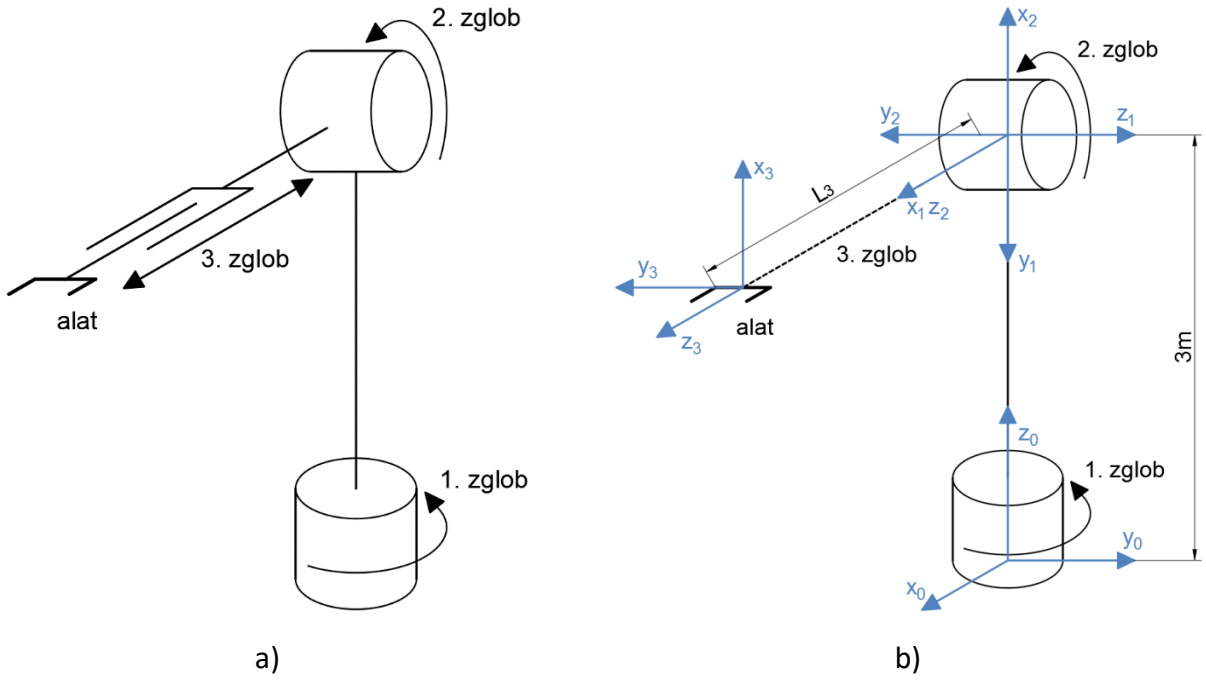
Slika 2.7. Slučaj kada su z osi paralelne

- z_{i-1} i z_i osi se presjecaju. Os x_i se tada nalazi na presjecištu osi z_{i-1} i z_i, a parametar a_i uvijek je $a_i = 0$ (Slika 2.8).



Slika 2.8. Slučaj kada se z osi presijecaju

Kao primjer obilježavanja koordinatnih sustava ispod je skicirana pojednostavljena shema robota s dva revolutna i jednim prizmatičnim zglobom, na kraju kojeg se nalazi alat (Slika 2.9.a).



Slika 2.9. Primjer obilježavanja koordinatnih osi D-H konvencijom

Poštujući pravila D-H konvencije obilježeni su svi koordinatni sustavi robota (Slika 2.9.b) i u tablicu upisani parametri za segmente robota (Tablica 2.1).

Tablica 2.1. D-H parametri trosegmentnog robota

Segment	a_i	α_i	d_i	θ_i
1	0	$-90°$	3 m	0
2	0	$-90°$	0	$-90°$
3	0	0	L_3	0

Nakon uvrštavanja nepromjenjivih parametara a_i i α_i svakog segmenta u (2.7) dobiju se tri transformacijske matrice $\mathbf{T}_1^0$, $\mathbf{T}_2^1$ i $\mathbf{T}_3^2$ dane izrazima (2.8), (2.9) i (2.10), čiji produkt daje (2.12):

$$\mathbf{T}_1^0 = \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.8}$$

$$\mathbf{T}_2^1 = \begin{bmatrix} c_2 & 0 & -s_2 & 0 \\ s_2 & 0 & c_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.9}$$

$$\mathbf{T}_3^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.10}$$

$$\mathbf{T}_2^0 = \mathbf{T}_1^0\mathbf{T}_2^1 = \begin{bmatrix} c_1c_2 & s_1 & -c_1s_2 & 0 \\ s_1c_2 & -c_1 & -s_1s_2 & 0 \\ -s_2 & 0 & -c_2 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

$$\mathbf{T}_3^0 = \mathbf{T}_2^0\mathbf{T}_3^2 = \begin{bmatrix} c_1c_2 & s_1 & -c_1s_2 & -L_3c_1s_2 \\ s_1c_2 & -c_1 & -s_1s_2 & -L_3s_1s_2 \\ -s_2 & 0 & -c_2 & 3 - L_3c_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.12}$$

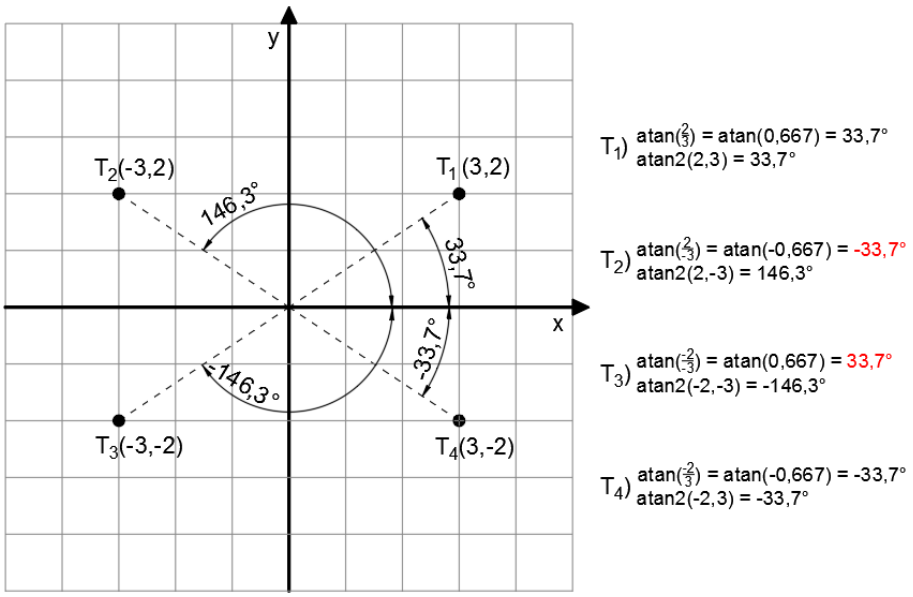
Konačna matrica transformacije $\mathbf{T}_3^0$ može se koristiti za računanje direktne kinematike robota, ukoliko su poznati promjenjivi parametri θ_i i d_i, odnosno ako su poznati svi kutevi zakreta i sve translacije zglobova robota. Uvrste li se vrijednosti iz primjera (Tablica 2.1) u (2.12), matrica transformacije $\mathbf{T}_3^0$ opisuje rotaciju i translaciju prihvatnice u odnosu na bazni koordinatni sustav robota kao:

$$\mathbf{T}_3^0 = \begin{bmatrix} 0 & 0 & 1 & L_3 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.13}$$

2.2. Inverzna kinematika

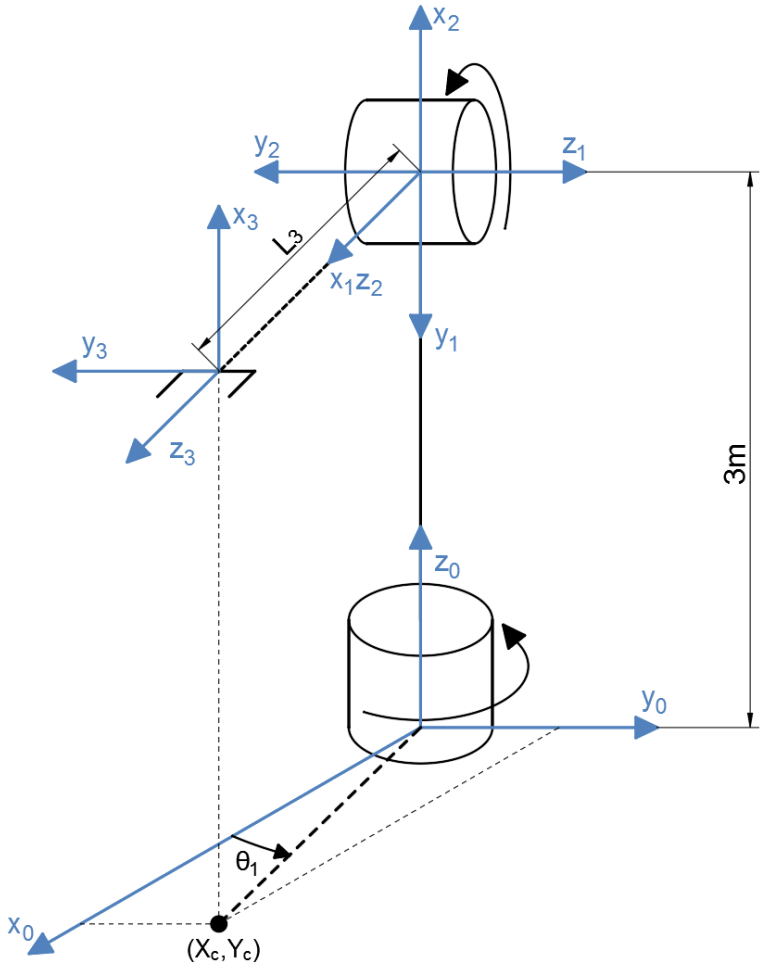
Ne postoje lagani univerzalni algoritmi za izračun inverzne kinematike jer je konfiguracija svakog robota unikatna, potrebno je dobro analizirati geometriju robota i primijeniti znanje trigonometrije. Pri računalnom računanju kutova poželjno je izbjegavati korištenje $\mathrm{acos}(x)$ i $\mathrm{asin}(x)$

trigonometrijskih funkcija zbog numeričke pogreške pri kutovima oko 0 ili $\frac{\pi}{2}$. Također je poželjno izbjegavati uporabu $\text{atan}(x)$ funkcije jer se njenom uporabom gubi informacija o predznaku varijabli. Preporučuje se korištenje $\text{atan2}(y,x)$ funkcije koja umjesto jedne uzima u obzir dvije varijable i njihove predznake. Kao što je vidljivo iz slike 2.10., uporabom $\text{atan2}(y,x)$ funkcije proširuje se raspon rješenja kutova sa $[-\frac{\pi}{2},\frac{\pi}{2}]$ na $[-\pi,\pi]$ [4] [6].



Slika 2.10. Usporedba $\text{atan}(x)$ i $\text{atan2}(y,x)$ trigonometrijskih funkcija

Kao što je napomenuto, inverznom kinematikom računaju se kutovi zakreta svih zglobova kako bi robot mogao prihvatnicom obavljati neku radnju na poznatoj poziciji i orijentaciji. Za jednostavni primjer trosegmentnog robota sa slike 2.9., nepoznati su upravljački parametri θ_1 , θ_2 i d_3, dok su pozicija i orijentacija prihvatnice definirani funkcijom koju robot obavlja. Ishodište robota isto je poznato pa je matrica transformacije $\mathbf{T}_3^0$, koja povezuje početni koordinatni sustav s krajnjim, također definirana.



Slika 2.11. Zadana orijentacija robota

Za neku određenu orijentaciju robota (slika 2.11.) potrebno je izračunati kut zakreta θ_1 prvog zgloba. Ako se točka centra alata projicira na bazni koordinatni sustav $\{x_0, y_0\}$, vidljivo je da je:

$$\theta_1 = \text{atan2}(Y_c, X_c), \tag{2.14}$$

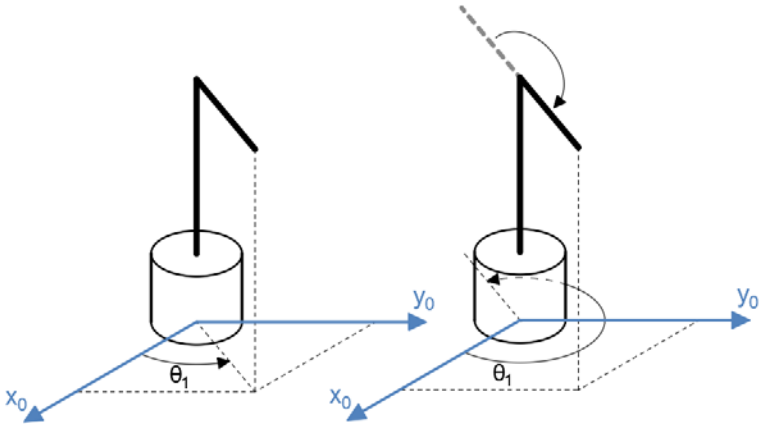
gdje su:

X_c, Y_c – koordinate centra alata.

Broj zglobova ovog robota jednak je broju dimenzija prostora $N = D$, stoga postoji samo jedno rješenje inverzne kinematike za svaki zglob. Kada bi robot imao $N > D$, kut θ_1 bi se također mogao izračunati i kao:

$$\theta_1 = \pi + \text{atan2}(Y_c, X_c), \tag{2.15}$$

jer bi se u tom slučaju ista pozicija i orijentacija alata mogla postići kao u (2.14) pod uvjetom da su zglob q_2 i neki novi četvrti revolutni zglob q_4 (primjerice na kraju prihvatnice, ali prije alata) također zakrenuti za 180° (Slika 2.12.).

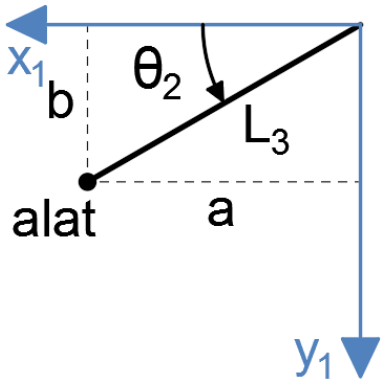


Slika 2.12. Primjer dvostrukog rješenja za kut θ_1

Razmotre li se preostali zglobovi iz perspektive koordinatnog sustava $\{x_1, y_1\}$ (Slika 2.13), vidljivo je da se varijable θ_2 i L_3 mogu izraziti kao:

$$\theta_2 = \text{atan2}(b, a), \tag{2.16}$$

$$L_3 = \sqrt{a^2 + b^2}. \tag{2.17}$$



Slika 2.13. Perspektiva koordinatnog sustava $\{x_1, y_1\}$

Promatrajući prostorno sliku robota 2.11., varijabla a može se izraziti kao horizontalna komponenta L_3 duljine u odnosu na bazni koordinatni sustav $\{x_0, y_0\}$:

$$a = \sqrt{X_c^2 + Y_c^2}, \tag{2.18}$$

dok se varijabla b može prikazati kao vertikalni odmak od visine robota od 3 m u obrnutom smjeru od osi z_0 tako da je:

$$b = 3 - Z_c. \tag{2.19}$$

Nakon uvrštavanja (2.18) i (2.19) u (2.16) i (2.17) dobiveni su:

$$\theta_2 = \text{atan2}(3 - Z_c, \pm\sqrt{X_c^2 + Y_c^2}), \tag{2.20}$$

$$L_3 = \sqrt{X_c^2 + Y_c^2 + (3 - Z_c)^2}, \tag{2.21}$$

gdje negativno rješenje varijable θ_2 iz (2.20) vrijedi za slučaj da je θ_1 izračunat relacijom (2.15).

2.3. Jacobijeva matrica

Rješenje direktnog i inverznog kinematičkog problema povezuje upravljane koordinate s vanjskim koordinatama robota i poznavanjem kinematičkog modela robota moguće je iz poznatih upravljanih koordinata $\mathbf{q}$ izračunati vanjske nepoznate koordinate $\mathbf{r}$ i obratno. Vektor upravljanih ili unutrašnjih koordinata $\mathbf{q}$ dan je izrazom:

$$\mathbf{q} = [q_1\ q_2\ \dots\ q_n]^T, \tag{2.22}$$

gdje je:

q_i – upravljana koordinata, predstavlja rotaciju ili translaciju pripadajućeg zgloba,
n – broj zglobova,

dok se vektor vanjskih koordinata $\mathbf{r}$ predstavlja kao:

$$\mathbf{r} = \left[p_x\ p_y\ p_z\ R_x\ R_y\ R_z\right]^T, \tag{2.23}$$

gdje:

p_x, p_y, p_z – predstavljaju položaj robota,
R_x, R_y, R_z – predstavljaju orijentaciju robota.

Pošto robot nije stacionaran dok obavlja neki zadatak često je korisno poznavati i njegovu brzinu u prostoru kao i vezu između brzina upravljanih i brzina vanjskih koordinata. Budući da je vektor vanjskih koordinata složena funkcija upravljanih koordinata (rješenje direktnog kinematičkog problema), može se pisati da je:

$$\mathbf{r} = \mathrm{f}(\mathbf{q}). \tag{2.24}$$

Utjecaj diferencijalne promjene upravljanih koordinata $\Delta\mathbf{q}$ na diferencijalnu promjenu vanjskih koordinata $\Delta\mathbf{r}$ može se predstaviti Jacobijevom matricom $\mathbf{J}$:

$$\mathbf{J} = \frac{\partial \mathbf{r}}{\partial \mathbf{q}}, \tag{2.25}$$

$$\Delta\mathbf{r} = \mathbf{J}(\mathbf{q})\Delta\mathbf{q}, \tag{2.26}$$

gdje $\mathbf{J}(\mathbf{q})$ označava da je Jacobijeva matrica funkcija upravljanih koordinata $\mathbf{q}$. Dijeljenjem jednadžbe (2.26) s diferencijalnom promjenom vremena Δt dobije se:

$$\dot{\mathbf{r}} = \mathbf{J}\dot{\mathbf{q}}, \tag{2.27}$$

što znači da Jacobijeva matrica povezuje brzine upravljanih i vanjskih koordinata. Daljnjim deriviranjem po vremenu dobije se i poveznica ubrzanja:

$$\ddot{\mathbf{r}} = \frac{\mathrm{d}\mathbf{J}}{\mathrm{d}t}\dot{\mathbf{q}} + \mathbf{J}\ddot{\mathbf{q}}. \tag{2.28}$$

Relacijom (2.27) vidljivo je da je moguće iz poznatih brzina upravljanih varijabli $\dot{\mathbf{q}}$ izračunati nepoznate brzine vanjskih koordinata $\dot{\mathbf{r}}$ uporabom Jacobijeve matrice $\mathbf{J}$. Kao što je to slučaj kod

direktne i inverzne kinematike, robotom je intuitivnije upravljati u okviru njegovih vanjskih koordinata nego koordinata njegovih pojedinih zglobova. Ako se obje strane relacije (2.27) pomnoži s invertiranom Jacobijevom matricom $\mathbf{J^{-1}}$ dobije se izraz:

$$\dot{\mathbf{q}}=\mathbf{J}^{-1}\dot{\mathbf{r}}, \tag{2.29}$$

kojim se iz poznatih brzina vanjskih koordinata $\dot{\mathbf{r}}$ mogu izračunati nepoznate brzine upravljanih koordinata $\dot{\mathbf{q}}$. Jacobijeva matrica $\mathbf{J}$ dimenzija je $m \times n$, gdje broj redova m predstavlja broj vanjskih koordinata, a broj stupaca n predstavlja broj upravljanih koordinata (broj zglobova robota), za slučaj primjera vrijedi:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{12} & J_{22} & J_{23} \\ J_{13} & J_{23} & J_{33} \\ J_{14} & J_{24} & J_{43} \\ J_{15} & J_{25} & J_{53} \\ J_{16} & J_{26} & J_{63} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{d}_3 \end{bmatrix} \tag{2.30}$$

Elemente Jacobijeve matrice moguće je izračunati koristeći pripadajuće članove za svaki zglob iz tablice 2.2.:

Tablica 2.2. Članovi za računanje Jacobijeve matrice

zglob	prizmatičan	revolutan
translacija	$\mathbf{R}_{i-1}^{0}\begin{bmatrix}0\\0\\1\end{bmatrix}$	$\mathbf{R}_{i-1}^{0}\begin{bmatrix}0\\0\\1\end{bmatrix} \times (\mathbf{P}_{n}^{0} - \mathbf{P}_{i-1}^{0})$
orijentacija	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\mathbf{R}_{i-1}^{0}\begin{bmatrix}0\\0\\1\end{bmatrix}$

gdje je i broj zgloba.

Za prvi i drugi zglob, koji su revolutni, koristi se desni stupac tablice 2.2., dok se za treći prizmatični zglob koristi lijevi stupac. Jacobijeva matrica $\mathbf{J}$ tada glasi:

$$\mathbf{J} = \begin{bmatrix} \mathbf{R}_0^0\begin{bmatrix}0\\0\\1\end{bmatrix} \times (\mathbf{P}_3^0 - \mathbf{P}_0^0) & \mathbf{R}_1^0\begin{bmatrix}0\\0\\1\end{bmatrix} \times (\mathbf{P}_3^0 - \mathbf{P}_1^0) & \mathbf{R}_2^0\begin{bmatrix}0\\0\\1\end{bmatrix} \\ \mathbf{R}_0^0\begin{bmatrix}0\\0\\1\end{bmatrix} & \mathbf{R}_1^0\begin{bmatrix}0\\0\\1\end{bmatrix} & \begin{bmatrix}0\\0\\0\end{bmatrix} \end{bmatrix}. \tag{2.31}$$

Matrice transformacija $\mathbf{T}_1^0$, $\mathbf{T}_2^0$ i $\mathbf{T}_3^0$, prethodno izračunate izrazima (2.8) do (2.12), i jedinična $4x4$ matrica transformacije $\mathbf{T}_0^0$ uvrštavaju se u (2.31) pa izraz (2.30) postaje:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} L_3 s_1 s_2 & -L_3 c_1 c_2 & -c_1 s_2 \\ -L_3 c_1 s_2 & -L_3 s_1 c_2 & -s_1 s_2 \\ 0 & L_3 s_2 & -c_2 \\ 0 & -s_1 & 0 \\ 0 & c_1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{d}_3 \end{bmatrix}, \tag{2.32}$$

iz čega slijede relacije:

$$\begin{aligned} \dot{x} &= L_3 s_1 s_2 \dot{\theta}_1 - L_3 c_1 c_2 \dot{\theta}_2 - c_1 s_2 \dot{d}_3, \\ \dot{y} &= -L_3 c_1 s_2 \dot{\theta}_1 - L_3 s_1 c_2 \dot{\theta}_2 - s_1 s_2 \dot{d}_3, \\ \dot{z} &= L_3 s_2 \dot{\theta}_2 - c_2 \dot{d}_3, \\ \omega_x &= -s_1 \dot{\theta}_2, \\ \omega_y &= c_1 \dot{\theta}_2, \\ \omega_z &= \dot{\theta}_1. \end{aligned} \tag{2.33}$$

Točnost Jacobijeve matrice može se provjeriti analizom jednadžbi (2.33), vidljivo je da:

- brzina upravljane koordinate $\dot{d}_3$ nikako ne utječe na brzinu orijentacije robota $\omega_x, \omega_y, \omega_z$, što je logično jer je d_3 prizmatični zglob,
- jedino brzina prvog zgloba $\dot{\theta}_1$ utječe na brzinu zakreta ω_z oko z osi, što je očito i iz slike robota,
- brzina prvog zgloba $\dot{\theta}_1$ ne utječe na brzinu pomaka $\dot{z}$ po z osi.

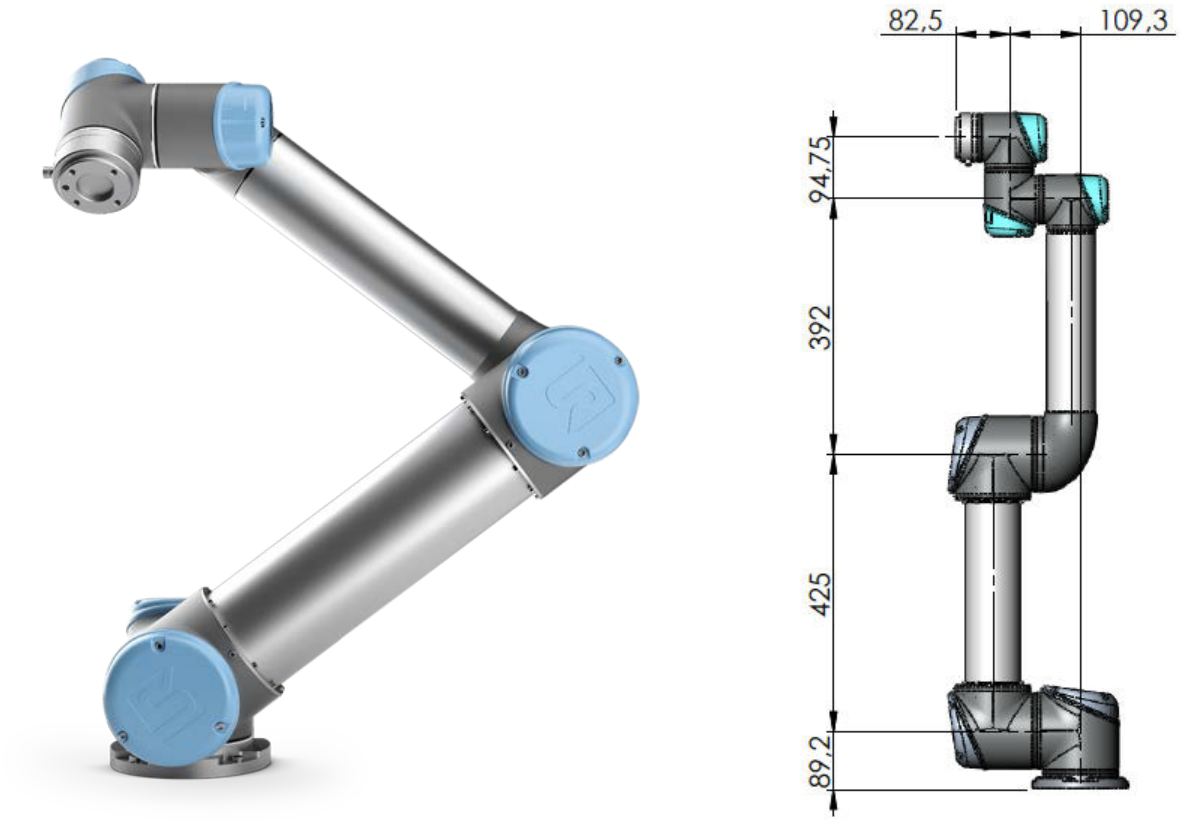
Robot iz primjera posjeduje samo tri upravljive koordinate $\mathbf{q}$ pa njegova Jacobijeva matrica ima dimenzije $\mathbf{J}_{6,3}$. Potrebno je odbaciti dio izračunate matrice (2.32) da bi matrica postala kvadratna te da joj se može izračunati inverz $\mathbf{J}^{-1}$, brzina translacije robota u tri osi može se izraziti samo kao:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} L_3 s_1 s_2 & -L_3 c_1 c_2 & -c_1 s_2 \\ -L_3 c_1 s_2 & -L_3 s_1 c_2 & -s_1 s_2 \\ 0 & L_3 s_2 & -c_2 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{d}_3 \end{bmatrix}. \tag{2.34}$$

Ako determinanta nove matrice $\mathbf{J}_{3,3}$ nije jednaka nuli za nju se može izračunati inverz i iz (2.29) računati nepoznate brzine upravljanih koordinata $\mathbf{q}$ preko poznatih brzina vanjskih koordinata $\mathbf{r}$.

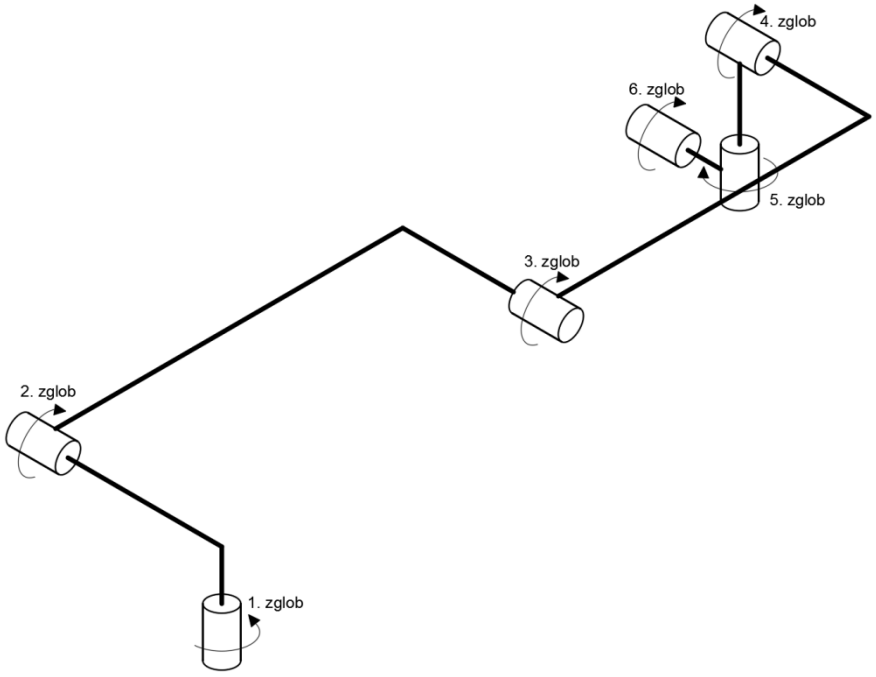
2.4. Inverzna kinematika UR5 robota [7]

Universal Robots UR5 robotska ruka programski je upravljana na daljinu i pomoću očitanja senzora sila prati trajektoriju drugog robota. Za potrebe programskog upravljanja izrađen je kinematički model UR5 robota, a u sljedećem poglavlju detaljnije su opisane njegove značajke.



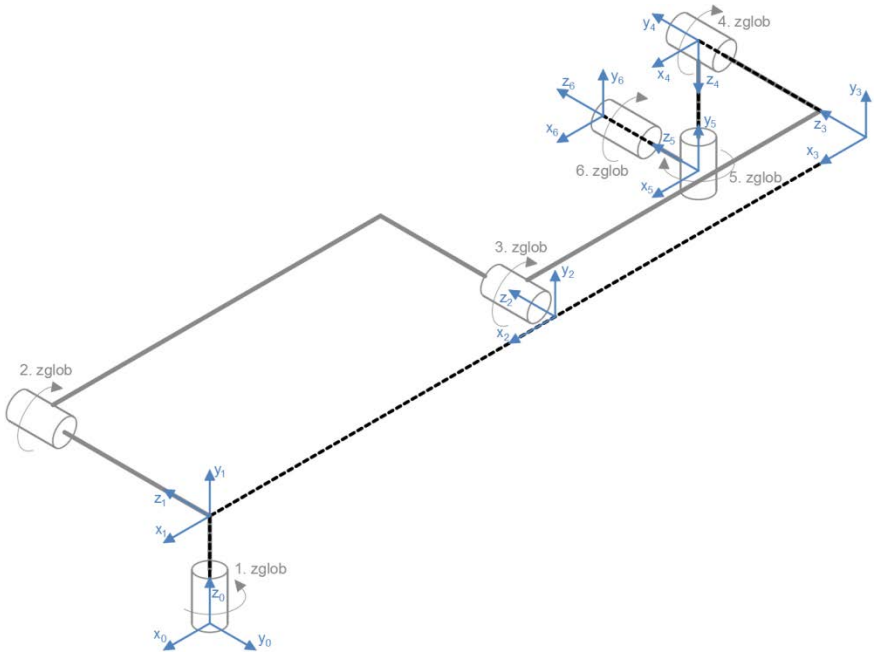
Slika 2.14. Universal Robots UR5 robotska ruka

Slikom 2.14. prikazan je UR5 industrijski robot i njegove glavne dimenzije, a slikom 2.15. prikazana je pojednostavljena shema robota. Radi olakšanja postavljanja koordinatnih sustava svi zglobovi robota postavljeni su u nulti položaj, odnosno zakret svih zglobova jednak je $\theta_i = 0°$.



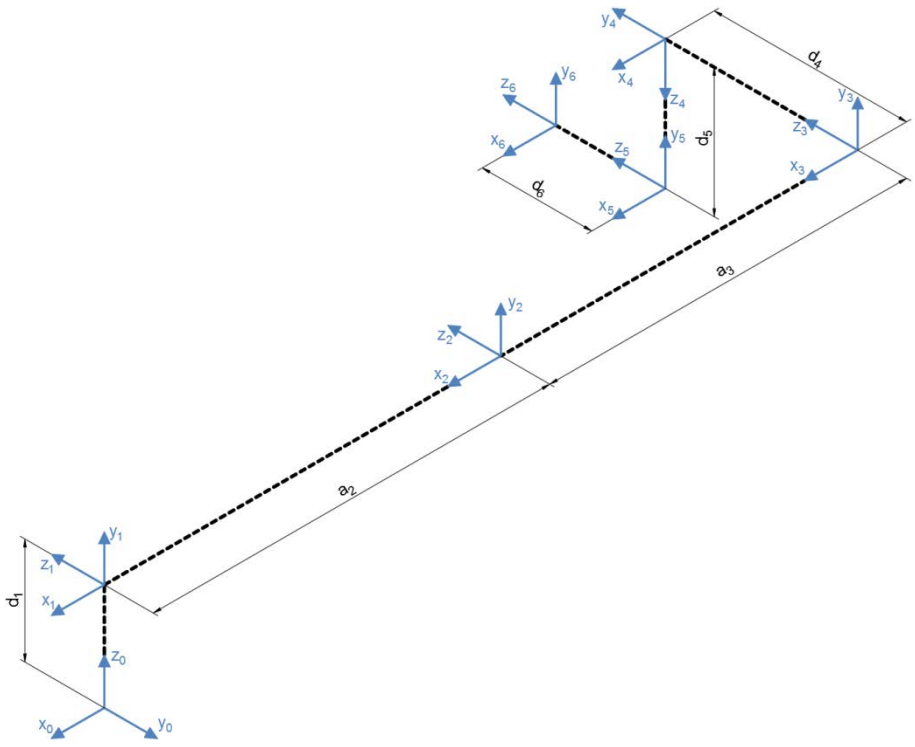
Slika 2.15. Pojednostavljeni prikaz UR5 robota

Prema pravilima D-H konvencije označeni su svi koordinatni sustavi, vodeći računa o pozitivnim smjerovima zakreta zglobova kod odabira smjera z osi (Slika 2.16).



Slika 2.16. Dodjeljivanje koordinatnih osi

Kompleksna geometrija robota može se uz poštivanje D-H konvencije jednostavnije opisati kao na slici 2.17. Tablica 2.3 prikazuje sve D-H parametre.

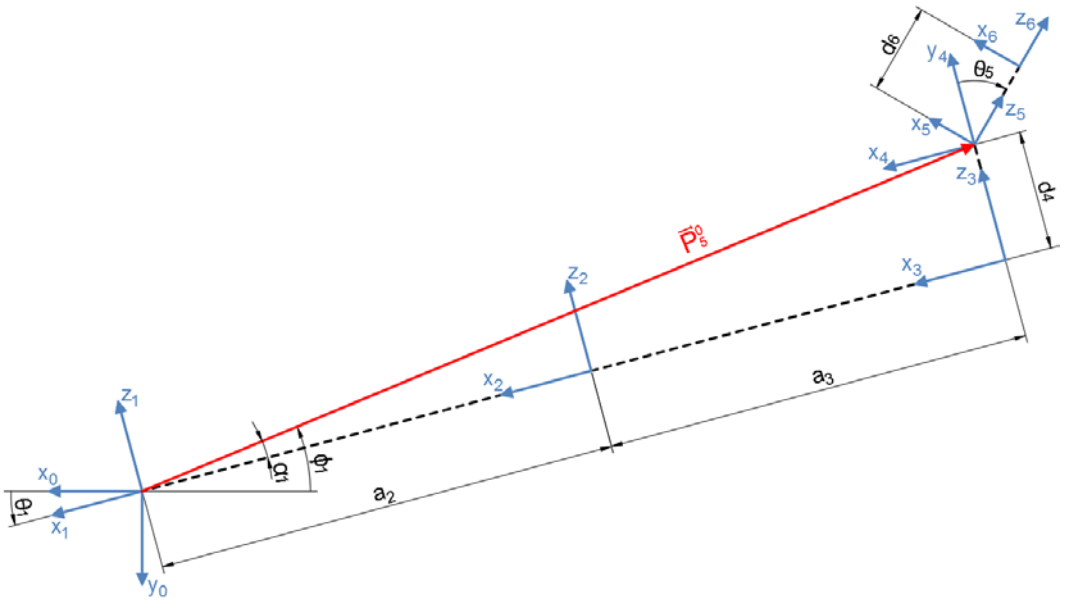


Slika 2.17. UR5 robot opisan D-H konvencijom

Tablica 2.3. D-H parametri UR5 robota

Segment	a_i **[mm]**	α_i [°]	d_i **[mm]**	θ_i
1	0	90	89,2	θ_1
2	−425	0	0	θ_2
3	−392	0	0	θ_3
4	0	90	109,3	θ_4
5	0	−90	94,75	θ_5
6	0	0	82,5	θ_6

Kao što je napomenuto u prijašnjem poglavlju, smjerovi z osi odabrani su koristeći pravilo desne ruke s obzirom na pozitivan smjer vrtnje svakog zgloba, a x_0 os odabrana je da odražava smjer interne x osi samog robota. Ostale x_i osi, $x_{2,3,4,5,6}$, usmjerene su kao početna zbog minimiziranja θ_i parametara.



Slika 2.18. Pronalaženje ishodišta $\{x_5, z_5\}$ koordinatnog sustava

Jedino je poznata matrica transformacije $\mathbf{T}_6^0$ kojom se opisuje prijelaz iz baznog koordinatnog sustava $\{x_0, y_0, z_0\}$ do vrha prihvatnice. Slikom 2.18. prikazana je projekcija robota gledanog odozgo i vidljivo je da su z_5 i z_6 osi jednako usmjerene za bilo koje gibanje robota, iz ovog razloga položaj ishodišta koordinatnog sustava $\{x_5, y_5, z_5\}$ može se izraziti jednostavnim oduzimanjem segmenta duljine d_6 u smjeru osi z_6 tako da:

$$\vec{\mathbf{P}}_5^0 = \vec{\mathbf{P}}_6^0 \begin{bmatrix} 0 \\ 0 \\ -d_6 \end{bmatrix}, \tag{2.35}$$

gdje su $\vec{\mathbf{P}}_5^0$ i $\vec{\mathbf{P}}_6^0$ pozicijski vektori matrica transformacija $\mathbf{T}_5^0$ i $\mathbf{T}_6^0$, odnosno. Također je vidljivo da se kut zakreta θ_1 može izraziti kao:

$$\theta_1 = \phi_1 - \alpha_1, \tag{2.36}$$

gdje su:

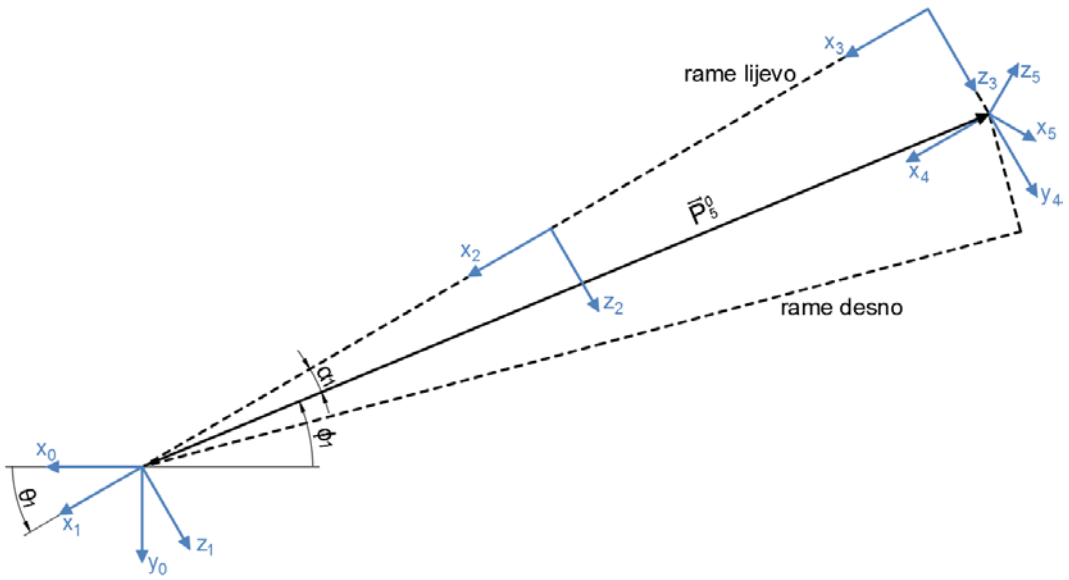
$$\phi_1 = \pi + \text{atan2}\big((P_5^0)_y, (P_5^0)_x\big), \tag{2.37}$$

$$\sin(\alpha_1) = \frac{d_4}{\sqrt{(P_5^0)_x^2 + (P_5^0)_y^2}}, \tag{2.38}$$

$$\cos(\alpha_1) = \pm\sqrt{1 - \sin^2(\alpha_1)}, \tag{2.39}$$

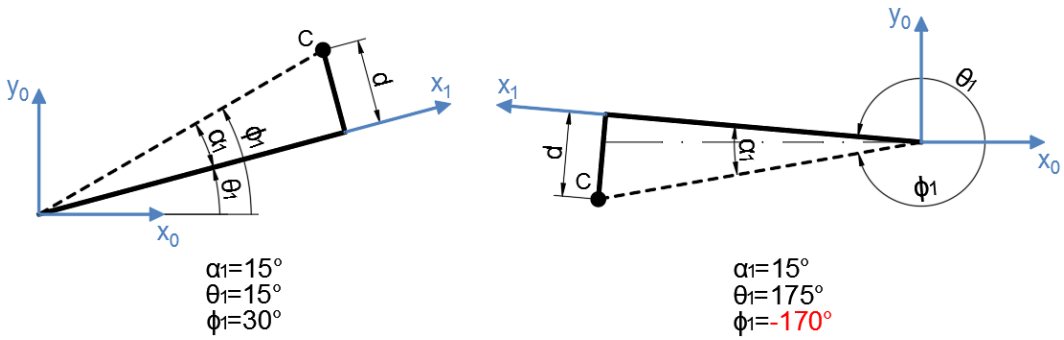
$$\alpha_1 = \text{atan2}(\sin(\alpha_1), \cos(\alpha_1)). \tag{2.40}$$

Treba napomenuti da relacija (2.39) ima dva rješenja pa kut α_1 može biti pozitivan ili negativan, kutovi u (2.36) tada se zbrajaju umjesto da se oduzimaju što odgovara slučaju kada je robot u izokrenutoj orijentaciji kao što je opisano slikom 2.12. Ovisno o predznaku, za rame robota smatra se da je s lijeve ili desne strane (Slika 2.19).



Slika 2.19. Rame lijevo ili rame desno

Važno je napomenuti da relacija (2.36) ne vrijedi za rubni slučaj kada je $\theta_1 < 180°$, a $\phi_1 > 180°$. Zakreće li se nekom robotu prvi zglob kutom θ_1, točka centra alata C će u jednom trenu prijeći put veći od π. U tom trenutku kut ϕ_1, računat funkcijom $\text{atan2}(y_C, x_C)$, neće iznositi preko 180°, već će biti neki manji negativan kut (Slika 2.20). Dobiveni rezultat tada uzrokuje neočekivane pokrete robota.



Slika 2.20. Posebni uvijet

Kako bi se izbjegao ovaj problem i proširio raspon rješenja kutova s $[0,\pi]$ na $[-\pi,\pi]$, umjesto (2.36) koriste se sljedeće relacije [8]:

$$\sin(\theta_1) = \sin(\phi_1 - \alpha_1), \tag{2.41}$$

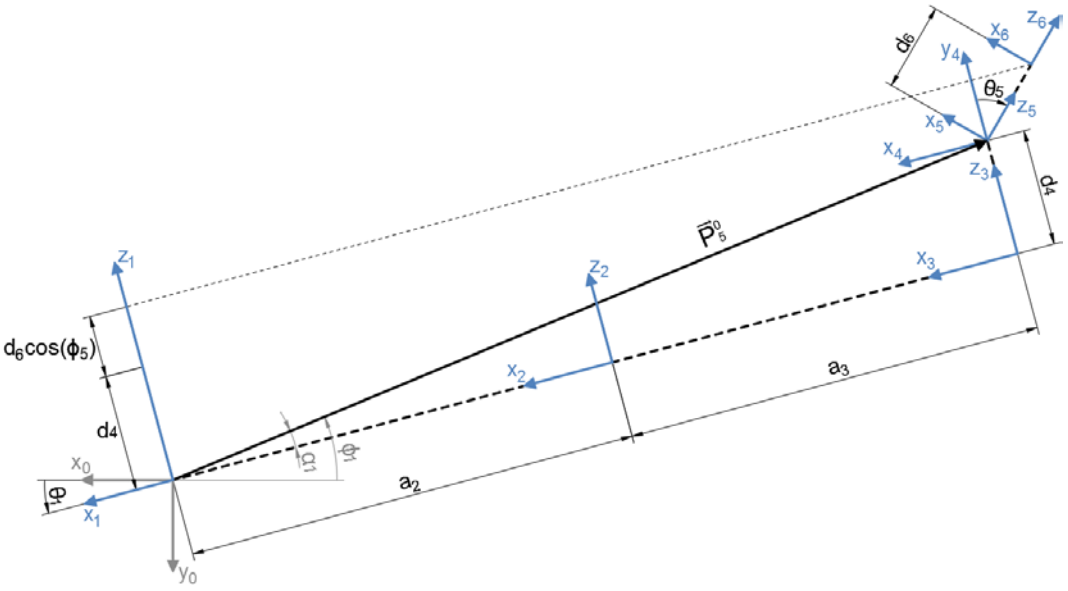
$$\cos(\theta_1) = \cos(\phi_1 - \alpha_1), \tag{2.42}$$

$$\theta_1 = \text{atan2}(\sin(\theta_1), \cos(\theta_1)). \tag{2.43}$$

Za nalaženje vrijednosti kuta θ_5 potrebno je sagledati robota iz perspektive koordinatnog sustava $\{x_1, y_1, z_1\}$ na način da se vrijednost prvog kuta θ_1, zajedno s odgovarajućim D-H parametrima, uvrsti u matricu transformacija (2.7). Pomoću dobivene matrice transformacije $\mathbf{T}_1^0$ i poznate matrice $\mathbf{T}_6^0$ računa se matrica $\mathbf{T}_6^1$:

$$\mathbf{T}_6^1 = (\mathbf{T}_1^0)^{-1}\mathbf{T}_6^0 = \mathbf{T}_0^1\mathbf{T}_6^0. \tag{2.44}$$

Ako se robota ponovo sagleda odozgo, ali iz perspektive $\{x_1, y_1, z_1\}$ koordinatnog sustava (Slika 2.21),



Slika 2.21. Nalaženje kuta θ_5

pozicija prihvatnice može se izraziti pomoću osi z_1 sljedećom relacijom:

$$(P_6^1)_z = d_6 \cos(\theta_5) + d_4, \tag{2.45}$$

gdje je $(P_6^1)_z$ poznata vrijednost iz $\mathbf{T}_6^1$ matrice transformacije:

$$\mathbf{T}_6^1 = \begin{bmatrix} x_x & y_x & z_x & (P_6^1)_x \\ x_y & y_y & z_y & (P_6^1)_y \\ x_z & y_z & z_z & \boldsymbol{(P_6^1)_z} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.46}$$

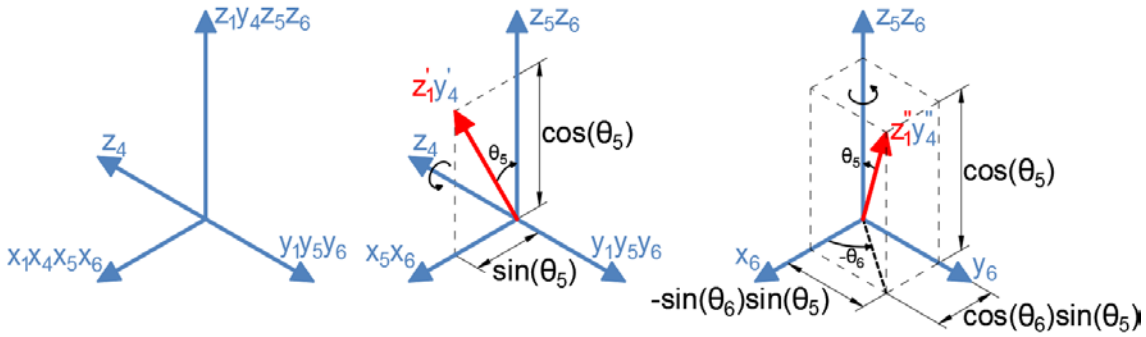
što znači da je:

$$\cos(\theta_5) = \frac{(P_6^1)_z - d_4}{d_6}, \tag{2.47}$$

$$\sin(\theta_5) = \pm\sqrt{1 - \cos^2(\theta_5)}, \tag{2.48}$$

$$\theta_5 = \operatorname{atan2}(\sin(\theta_5), \cos(\theta_5)). \tag{2.49}$$

Peti zglob, zakrenut kutom θ_5, može se nazvati zapešćem robota. Za zapešće, kao i za prvi zglob kojeg se može nazvati ramenom robota, postoje dva rješenja. Relacija (2.48) može biti negativna, a time će i kut θ_5 biti negativan, što znači da se zapešće nalazi u donjem položaju.



Slika 2.22. Nalaženje kuta θ_6

Za računanje kuta θ_6 potrebno je zanemariti translacije koordinatnih sustava i promatrati z_1 os kao jedinični vektor koji se zakreće u odnosu na koordinatni sustav $\{x_6, y_6, z_6\}$. Vektor $\mathbf{z}_1$ tada se može opisati sfernim koordinatama gdje kut θ_5 predstavlja kutnu visinu (eng. *polar angle*), a kut θ_6 predstavlja azimut (Slika 2.22). X i y komponente jediničnog vektora $\mathbf{z}_1$ mogu se očitati direktno ih matrice transformacije $\mathbf{T}_1^6$, koja je jednaka invertiranoj matrici $\mathbf{T}_6^1$:

$$\mathbf{T}_1^6 = (\mathbf{T}_6^1)^{-1} = \begin{bmatrix} x_x & y_x & \boldsymbol{z_x} & (P_1^6)_x \\ x_y & y_y & \boldsymbol{z_y} & (P_1^6)_y \\ x_z & y_z & z_z & (P_1^6)_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.50}$$

$$z_x = \cos(\theta_6)\sin(\theta_5), \tag{2.51}$$

$$z_y = -\sin(\theta_6)\sin(\theta_5), \tag{2.52}$$

$$\theta_6 = \operatorname{atan2}\left(\frac{-z_y}{\sin(\theta_5)}, \frac{z_x}{\sin(\theta_5)}\right). \tag{2.53}$$

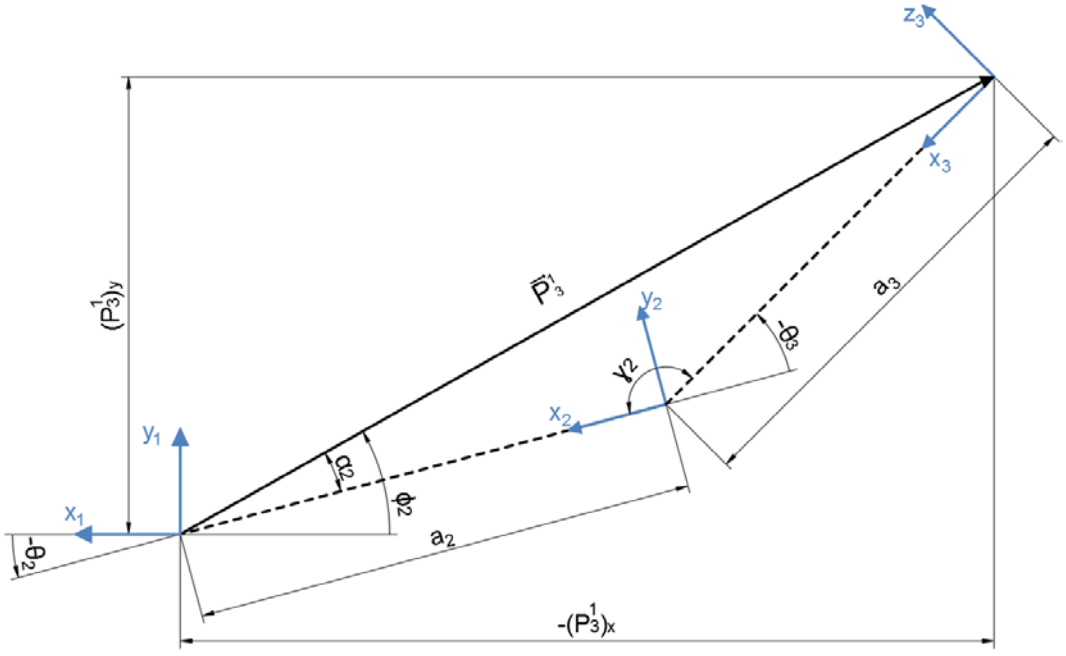
Vidljivo je da kut θ_6 nije definiran kada je $\sin(\theta_5) = 0$, odnosno kada su zglobovi 2, 3, 4 i 6 paralelni. Pošto se tada pozicija i orijentacija prihvatnice u ravnini određuju s četiri zgloba (četiri stupnja slobode gibanja), broj rješenja je beskonačan. Kako bi sustav bio definiran i za taj slučaj odabire se neka proizvoljna vrijednost θ_6.

Kutovi θ_2 i θ_3 mogu se smatrati dijelom planarnog dvosegmentnog robota kao na slici 2.23. Uvrste li se kutovi θ_5 i θ_6 u (2.7) moguće je tvoriti matrice transformacija $\mathbf{T}_5^4$ i $\mathbf{T}_6^5$ pa se matrica $\mathbf{T}_4^1$ može izraziti kao:

$$\mathbf{T}_4^1 = \mathbf{T}_6^1\mathbf{T}_4^6 = \mathbf{T}_6^1(\mathbf{T}_5^4\mathbf{T}_6^5)^{-1}, \tag{2.54}$$

nakon čega se položaj ishodišta koordinatnog sustava $\{x_3, y_3, z_3\}$ može izračunati oduzimanjem segmenta duljine d_4 u smjeru osi y_4:

$$\vec{\mathbf{P}}_3^1 = \vec{\mathbf{P}}_4^1 \begin{bmatrix} 0 \\ -d_4 \\ 0 \end{bmatrix}. \tag{2.55}$$



Slika 2.23. Nalaženje θ_2, θ_3

Norma pozicijskog vektora $\left|\vec{\mathbf{P}}_3^1\right|$ može se jednostavno izraziti kao:

$$\left|\vec{\mathbf{P}}_3^1\right| = \sqrt{(P_3^1)_x^2 + (P_3^1)_y^2 + (P_3^1)_z^2}, \tag{2.56}$$

gdje je $(P_3^1)_z^2$ uvijek jednak nuli. Iz slike 2.23. vidljivo je da za unutrašnje kutove trokuta $\Delta a_2 a_3 \left|\vec{\mathbf{P}}_3^1\right|$ vrijedi kosinusni poučak:

$$\cos(\gamma) = \frac{a_2^2 + a_3^2 - \left|\vec{\mathbf{P}}_3^1\right|^2}{2a_2 a_3}, \tag{2.57}$$

iz čega slijedi:

$$\cos(\gamma) = -\cos(\pi - \gamma) = -\cos(-\theta_3) = \cos(\theta_3), \tag{2.58}$$

$$\sin(\theta_3) = \pm\sqrt{1 - \cos^2(\theta_3)}, \tag{2.59}$$

$$\theta_3 = \operatorname{atan2}\big(\sin(\theta_3), \cos(\theta_3)\big). \tag{2.60}$$

Nadalje:

$$\cos(\alpha_2) = \frac{\left|\vec{\mathbf{P}}_3^1\right|^2 + a_2^2 - a_3^2}{2\left|\vec{\mathbf{P}}_3^1\right|^2 a_2}, \tag{2.61}$$

$$\sin(\alpha_2) = \pm\sqrt{1 - \cos^2(\alpha_2)}, \tag{2.62}$$

$$\alpha_2 = \operatorname{atan2}\big(\sin(\alpha_2), \cos(\alpha_2)\big), \tag{2.63}$$

$$\phi_2 = \pi - \operatorname{atan2}\big((P_3^1)_y, (P_3^1)_x\big), \tag{2.64}$$

i konačno:

$$\sin(\theta_2) = \sin(\phi_2 - \alpha_2), \tag{2.65}$$

$$\cos(\theta_2) = \cos(\phi_2 - \alpha_2), \tag{2.66}$$

$$\theta_2 = -\operatorname{atan2}(\sin(\theta_2), \cos(\theta_2)). \tag{2.67}$$

Relacije (2.59) i (2.62) imaju dvostruka rješenja pa tako kutovi θ_2 i θ_3 mogu biti pozitivni ili negativni. Zglobovi 2 i 3 zajedno mogu se smatrati laktom robota koji može biti orijentiran gore ili dolje.

Razmotri li se izraz (2.7) vidljivo je da se jedini nepoznati kut θ_4 može lako izračunati direktno iz matrice transformacija $\mathbf{T}_4^3$:

$$\mathbf{T}_i^{i-1} = \begin{bmatrix} \boldsymbol{c_{\theta_i}} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ \boldsymbol{s_{\theta_i}} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.68}$$

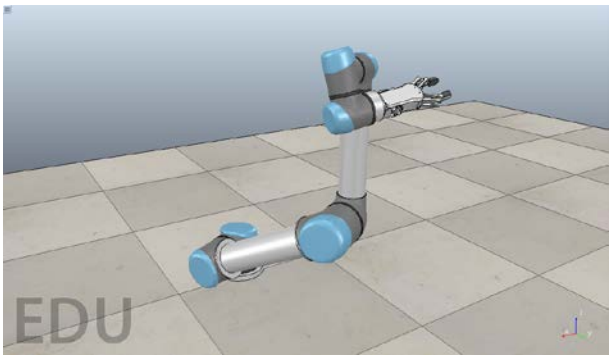
Uvrštavanjem kutova θ_2 i θ_3 u (2.68) dobivaju se matrice transformacija $\mathbf{T}_2^1$ i $\mathbf{T}_3^2$, a uz već izračunatu matricu $\mathbf{T}_4^1$ moguće je sastaviti matricu transformacija $\mathbf{T}_4^3$:

$$\mathbf{T}_4^3 = \mathbf{T}_1^3 \mathbf{T}_4^1 = (\mathbf{T}_2^1 \mathbf{T}_3^2)^{-1} \mathbf{T}_4^1, \tag{2.69}$$

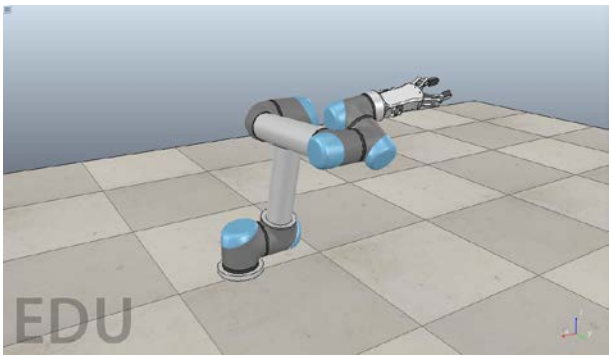
iz jednostavno izračunati θ_4 kao:

$$\theta_4 = \text{atan2}(x_y, x_x). \quad \textbf{(2.70)}$$

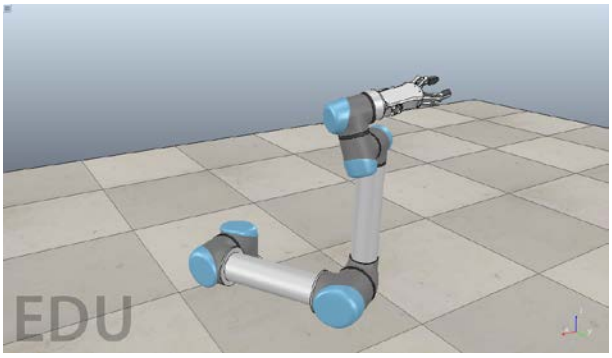
Sa svim izračunatim kutovima moguće je za jednu poziciju i orijentaciju prihvatnice dobiti osam rješenja inverzne kinematike, ovo je prikazano *V-REP PRO EDU* platformom kompanije *Coppelia Robotics GmbH* koristeći originalni model UR5 robota (Slika 2.24).



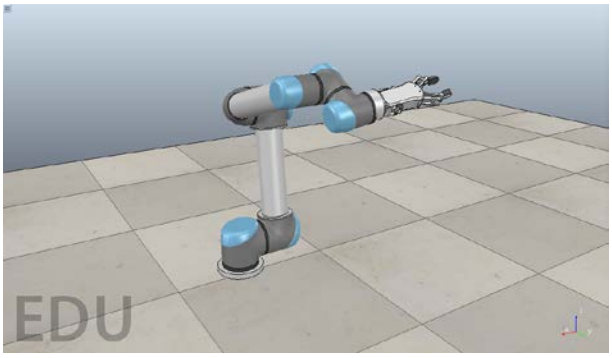
a) rame lijevo, lakat gore, zapešće dolje



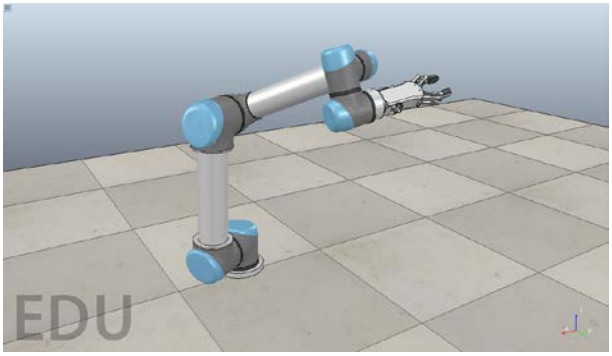
b) rame desno, lakat dolje, zapešće dolje



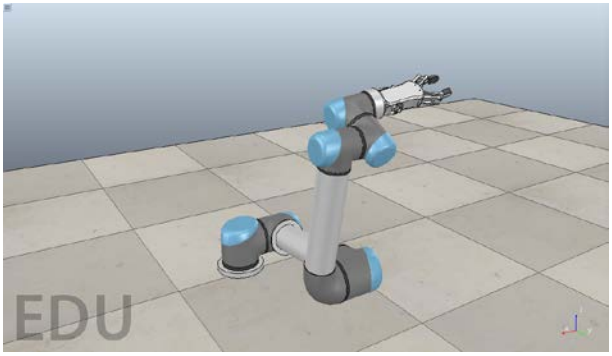
c) rame lijevo, lakat gore, zapešće gore



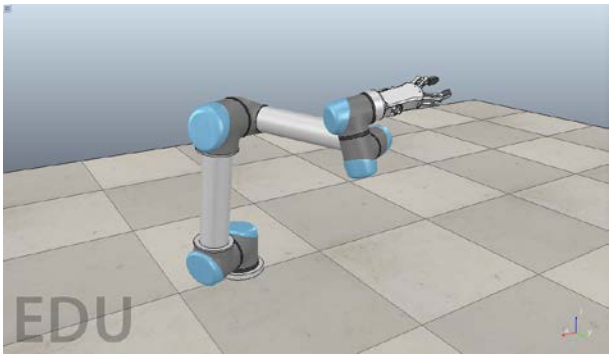
d) rame desno, lakat dolje, zapešće gore



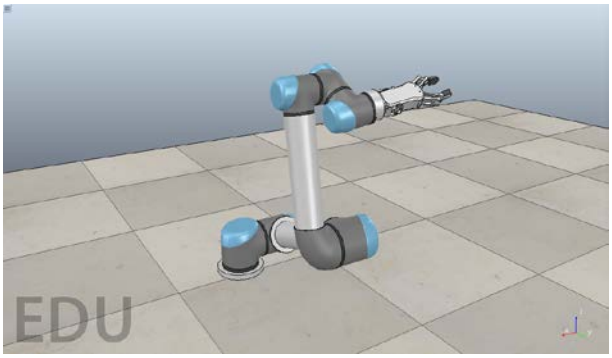
e) rame lijevo, lakat dolje, zapešće dolje



f) rame desno, lakat gore, zapešće dolje



g) rame lijevo, lakat dolje, zapešće gore



h) rame desno, lakat gore, zapešće gore

Slika 2.24. Osam mogućih rješenja inverzne kinematike robota

2.5. Jacobijeva matrica UR5 robota

Jacobijeva matrica računa se kao i u jednostavnom primjeru, kutove izračunate inverznom kinematikom pod (2.43) (2.49) (2.53) (2.60) (2.67) i (2.70) uvrsti se u (2.7) i izračunaju se matrice transformacija $\mathbf{T}_1^0$, $\mathbf{T}_2^0$, $\mathbf{T}_3^0$, $\mathbf{T}_4^0$, $\mathbf{T}_5^0$ i $\mathbf{T}_6^0$:

$$\mathbf{T}_1^0 = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & 89{,}2 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{T}_2^0 = \begin{bmatrix} c_1c_2 & -c_1s_2 & s_1 & -425c_1c_2 \\ c_2s_1 & -s_1s_2 & -c_1 & -425c_2s_1 \\ s_2 & c_2 & 0 & 89{,}2 - 425s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.71}$$

$$\mathbf{T}_3^0 = \begin{bmatrix} c_1(c_2c_3 - s_2s_3) & -c_1(c_2s_3 + c_3s_2) & s_1 & c_1(392(s_2s_3 - c_2c_3) - 425c_2) \\ s_1(c_2c_3 - s_2s_3) & -s_1(c_2s_3 - c_3s_2) & -c_1 & s_1(392(s_2s_3 - c_2c_3) - 425c_2) \\ c_2s_3 + c_3s_2 & c_2c_3 - s_2s_3 & 0 & 89{,}2 - 392(c_2s_3 + c_3s_2) - 425s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{T}_4^0 = [\dots], \mathbf{T}_5^0 = [\dots], \mathbf{T}_6^0 = [\dots].$$

Jacobijeva matrica **J** dimenzija je 6×6 jer UR5 ima šest upravljivih koordinata (zglobova):

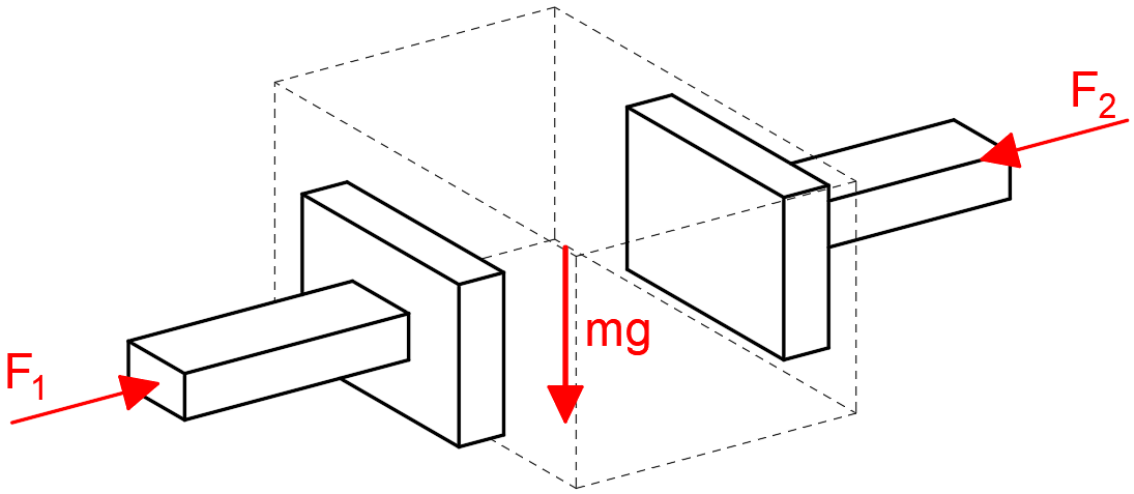
$$\mathbf{J} = \begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} & J_{15} & J_{16} \\ J_{12} & J_{22} & J_{23} & J_{24} & J_{25} & J_{26} \\ J_{13} & J_{23} & J_{33} & J_{34} & J_{35} & J_{36} \\ J_{14} & J_{24} & J_{43} & J_{44} & J_{45} & J_{46} \\ J_{15} & J_{25} & J_{53} & J_{45} & J_{55} & J_{56} \\ J_{16} & J_{26} & J_{63} & J_{46} & J_{56} & J_{66} \end{bmatrix}. \tag{2.72}$$

Koristeći stupac za revolutne zglobove u tablici (2.31) i izračunate matrice transformacija (2.71) popunjava se Jacobijeva matrica (2.72) i dobiva se izraz za računanje brzina vanjskih koordinata u odnosu na poznate brzine upravljanih koordinate (2.73). Invertiranjem matrice **J** moguće je računanje brzina upravljanih koordinata u odnosu na poznate brzine vanjskih koordinata (2.74):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \mathbf{J} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \\ \dot{\theta}_5 \\ \dot{\theta}_6 \end{bmatrix}, \tag{2.73}$$

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \\ \dot{\theta}_5 \\ \dot{\theta}_6 \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \tag{2.74}$$

3. Postava tehničkog sustava



Slika 3.1. Dvoručno rukovanje predmetom

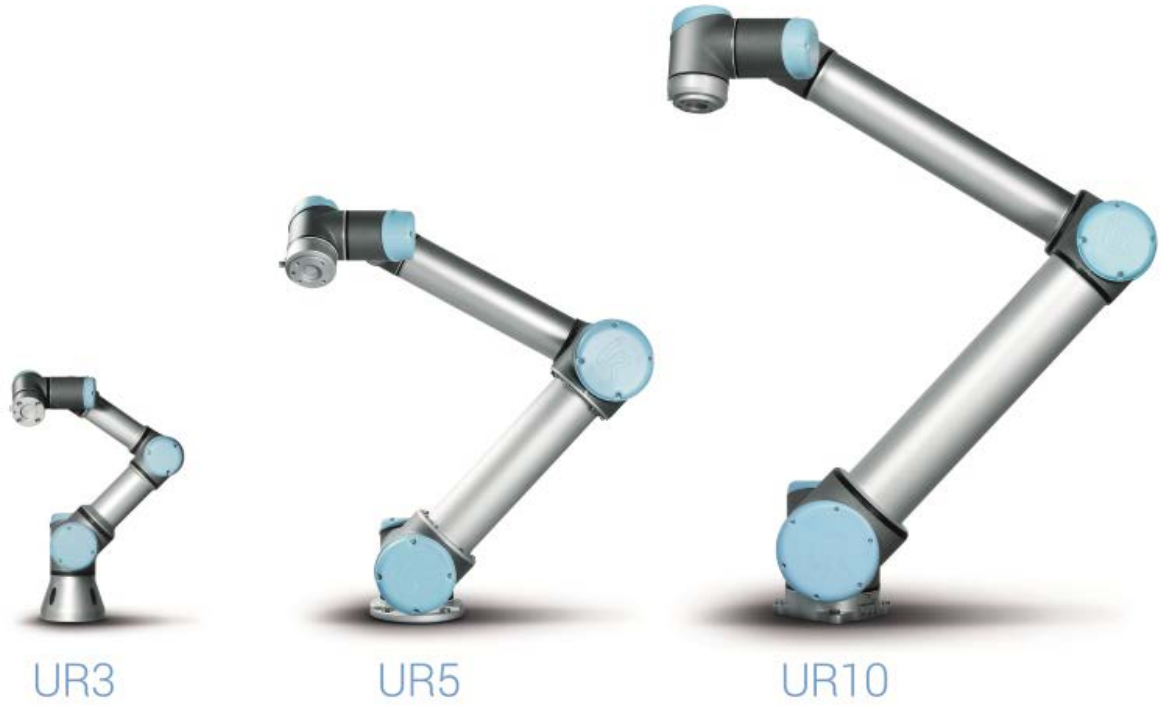
Cilj ovog rada je ostvariti rukovanje predmetom u prostoru, prikazano slikom 3.1., na način da dvije robotske ruke s ravnim prihvatnicama pridržavaju predmet primjenom sile. Predmet će ostati u prihvatu sve dok sila trenja između prihvatnica i predmeta može nadjačati silu težine predmeta, a pomak u smjeru djelovanja sila F_1 i F_2 ostvaruje se njihovim omjerom. Ukoliko je sila F_1 manja od sile F_2, predmet će se gibati u smjeru sile F_2 i obratno. Poželjno je da sila prihvata nije premala kako bi se izbjeglo klizanje predmeta uslijed početka gibanja ili promjene smjera, također je poželjno da sila nije prevelika kako ne bi došlo do deformacije predmeta. Sila prihvata idealno je konstantna i pozitivna u smjeru normale na površinu predmeta.

Pridržavanje predmeta i praćenje trajektorije zadaća je samo jedne ruke u ovom radu, stoga se sila prihvata regulira jednom silom, odnosno programskim upravljanjem jednim robotom. Jedina zadaća druge ruke je zadavanje trajektorije koju prva ruka treba pratiti. Osim pomaka u smjeru sile prihvata, predmet je potrebno moći translatirati i rotirati po svim trima osima.

Industrijski robot UR3 proizvođača *Universal Robots* korišten je u ulozi robota koji zadaje trajektoriju jednostavnim predefiniranim programom kretanja napisanim na privjesku za učenje. UR5 robot upravljan je programski od strane eksternog računala i, uz pomoć FT 150 senzora sila i momenata proizvođača *Robotiq*, prati trajektoriju predmeta.

3.1. *Universal Robots* industrijski roboti

Danski proizvođač Universal Robots u svojoj ponudi ima tri veličine industrijskih robota, na slici 3.2. poredani po veličini od najmanjeg prema najvećem to su UR3, UR5 i UR10. Iz imena se lako može iščitati maksimalna korisna nosivost (eng. *payload*) svakog robota, odnosno masa kojom robot može rukovati (npr. maksimalna korisna nosivost za robota UR5 iznosi 5 kilograma).



Slika 3.2. Postava robota proizvođača *Universal Robots*

Svi roboti sastoje se od šest revolutnih zglobova i ubrajaju se u robote sa šest stupnjeva slobode gibanja. Razmjerno s veličinom, svaki robot savršeno je namijenjen za određeni tip zadatka, UR3 robot, sa svojim manjim dimenzijama i beskonačno rotirajućim šestim zglobom idealan je za montažu manjih dijelova i sklapanje vijcima [9]. Usporedba glavnih tehničkih karakteristika dana je tablicom 3.1.

Tablica 3.1. Usporedba karakteristika UR robota

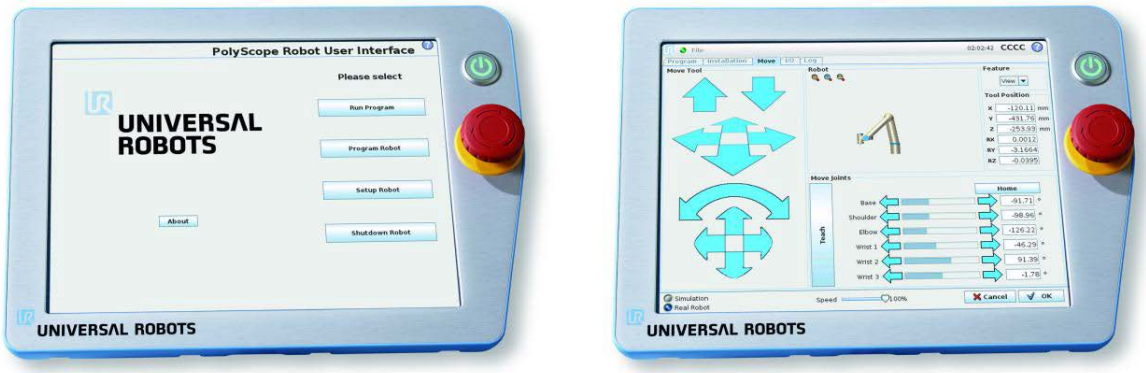
Model robota	UR3	UR5	UR10
Masa	11 kg	18,4 kg	28,9 kg
Korisna nosivost	3 kg	5 kg	10 kg
Doseg	500 mm	850 mm	1300 mm
Raspon kuta zglobova	±360°, zadnji zglob: ±∞	±360°	±360°
Brzina	prva tri zgloba: ±180°/s, ostali zglobovi: ±360°/s	±180°/s	prva dva zgloba: ±120°/s, ostali zglobovi: ±180°/s
IP procjena	IP64	IP54	IP54
Ponovljivost		±0,1 mm	
Tipična potrošnja	125 W	150 W	250 W

Naspram klasičnih industrijskih robota (Slika 3.3), koji moraju biti okruženi sigurnosnim kavezom ukoliko su postavljeni u blizini radnika, UR roboti smatraju se sigurnim robotima i mogu raditi u neposrednoj blizini radnika bez sigurnosnog kaveza. Upravljačka jedinica UR robota konstantno radi očitanja napona i struje u zglobovima robota i estimira moment primijenjen na svaki zglob

(moment električkog motora linearno je proporcionalan struji motora [10]). Robot se automatski zaustavlja u trenutku kada estimirana sila bilo kojeg zgloba prekorači određenu vrijednost (200 N u slučaju UR5 robota) što ga čini idealnim za znanstveni rad i blisku suradnju s ljudima. UR roboti i dalje nisu bezopasni i svakako mogu ozlijediti čovjeka, ali daleko su sigurniji od klasičnih robota bez ovih sigurnosnih sustava.

Slika 3.3. Klasični industrijski roboti

Operativni sustav robota zasniva se na *Linux* platformi sa posebnim programskim sučeljem *PolyScope* dostupnim na privjesku za učenje (eng. *teach pendant*) (Slika 3.4). Jednostavno programsko sučelje robota nalik je na *C++* i omogućuje pisanje programa na samom privjesku i dostatno je za većinu zadataka (npr. *pick and place, force control*, paletiziranje). Ukoliko je potrebna dodatna obrada podataka za složenije upravljanje robotom, moguće je i preuzimanje podataka o stanju robota i slanje naredbi od eksternog računala na daljinu putem TCP/IP protokola.



Slika 3.4. Privjesak za učenje

3.2. Robotiq FT 150 senzor sila i momenata [11]

Kao što je napomenuto, UR robot može bez dodatnog senzora estimirati sile i momente u zglobovima i estimirane vrijednosti mogu se koristiti unutar *PolyScope* sučelja. Postavlja se pitanje je li moguće jednostavno osposobiti upravljanje silom na privjesku za učenje i rukovati predmetom u prostoru bez ikakve dodatne opreme i programske obrade. Nažalost, estimirane vrijednosti sila i momenata poprilično su netočne i nekonzistentne su s obzirom na položaj robota pa je upotreba dodatnog senzora sila i momenata neophodna za ovaj zadatak. Senzor sa slike 3.5., FT 150 proizvođača *Robotiq*, idealan je dodatak UR robotima jer je proizveden u suradnji s proizvođačem *Universal Robots* s ciljem što bolje integracije.

Slika 3.5. Šesteroosni senzor sila i momenata FT 150

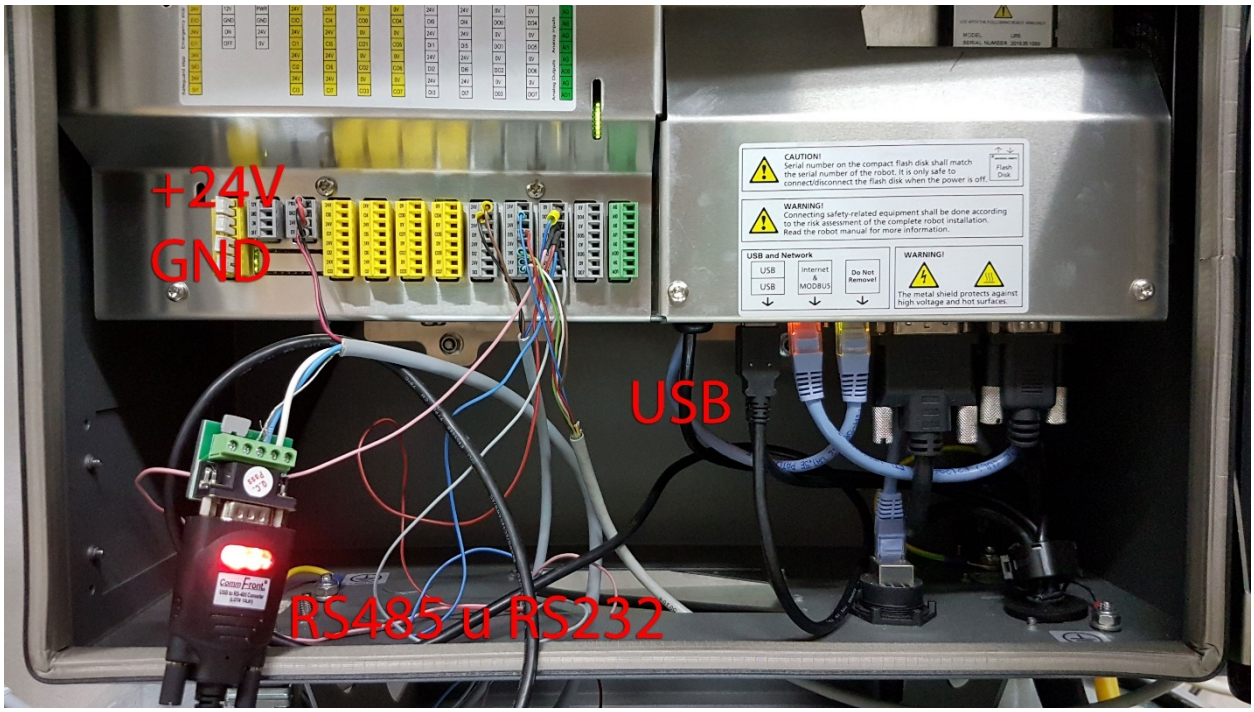
Glavne karakteristike senzora dane su tablicom 3.2.

Tablica 3.2. Karakteristike FT 150 senzora

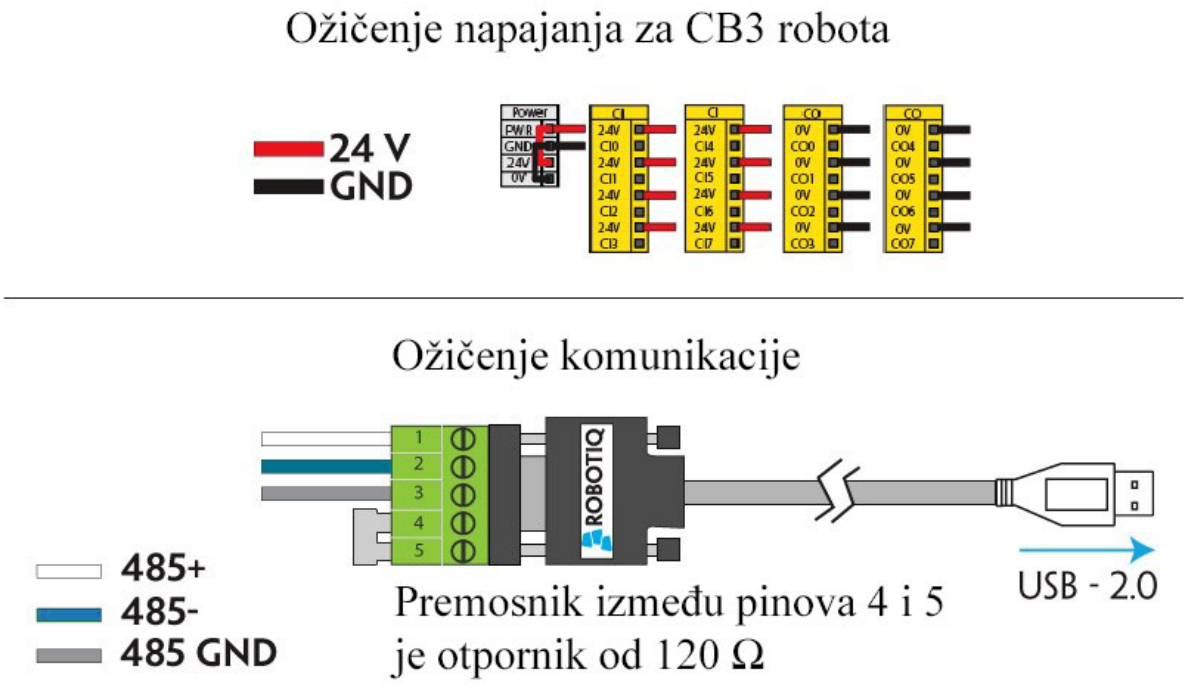
Model senzora	**FT 150**
Masa	650 g
Mjerni raspon	$\pm$150 N $\pm$15 Nm
Zanošenje (eng. *drift*)	F_x, F_y, F_z $\pm$3 N (u danima) M_x, M_y, M_z neznatno
Preopterećenje	750 N (500%)
Šum	F_x, F_y 0,5 N F_z 0,25 n M_x, M_y 0,015 Nm M_z 0,02 Nm
Brzina odziva	100 Hz
Vanjski promjer	120 mm
Unutarnji promjer	45 mm
Debljina	37,5 mm
IP procjena	54

3.2.1. Instalacija senzora

Spajanje senzora prikazano je slikom 3.7., senzor se spaja direktno u računalo UR robota. Napajanje od 24 V i uzemljenje GND klemom za napajanje spajaju se odvojeno od podatkovnog sučelja koje se preko RS485 u RS232 adaptera spaja USB konektorom u računalo robota. Dijagram ožičenja senzora prikazan je slikom 3.7.



Slika 3.6. Spajanje FT 150 senzora u računalo robota



Slika 3.7. Dijagram ožičenja senzora

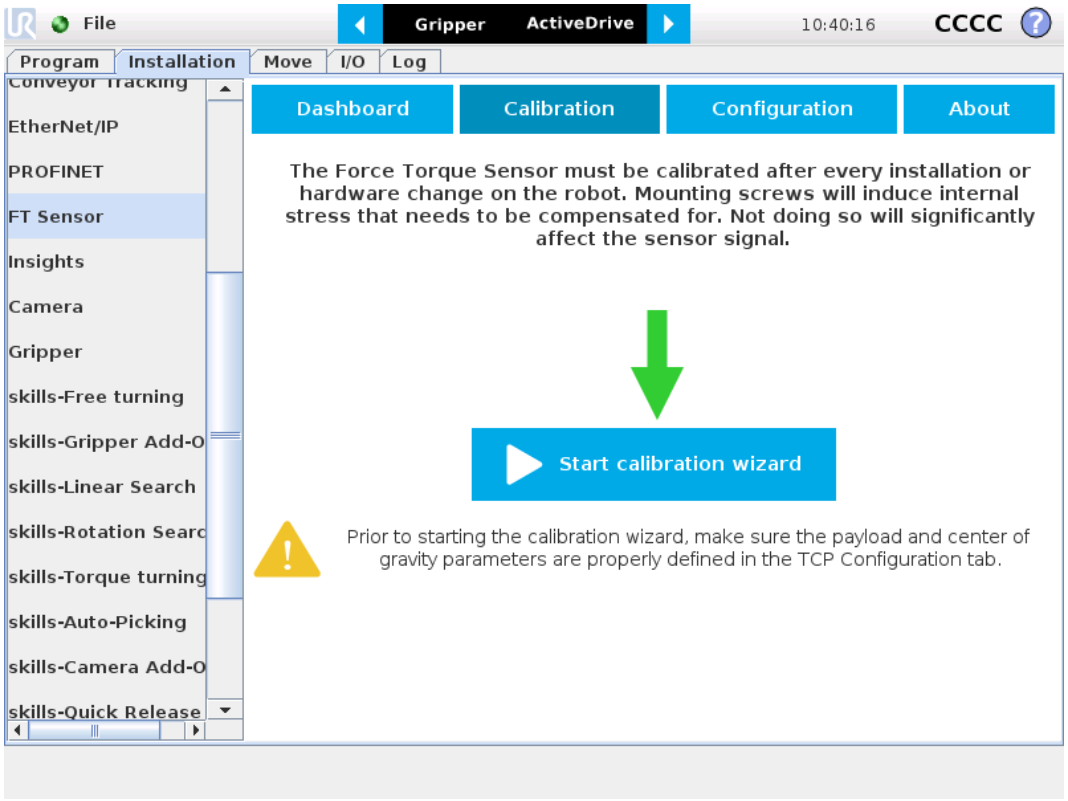
Kako bi računalo robota moglo očitavati vrijednosti sa FT 150 senzora potrebno je na računalu robota instalirati *URCaps* paket. *URCaps* (*caps* dolazi od eng. riječi *capabilities*) je platforma u razvoju za implementiranje hardverskih i softverskih dodataka na UR robote, zamišljena je kao besplatna platforma gdje distributeri, razvijači usluga i dobavljači predlažu gotova rješenja svojih *URCaps* sustava i *Universal Robots* odlučuje o njihovoj implementaciji.

Kompatibilnost *URCaps* platforme i UR robota ovisi je o hardverskoj verziji UR računala (Tablica 3.3) i softverskoj verziji *PolyScope* programskog sučelja (*PolyScope* verzija 3.3 ili novije).

Tablica 3.3. Kompatibilnost *URCaps* platforme s verzijama UR računala

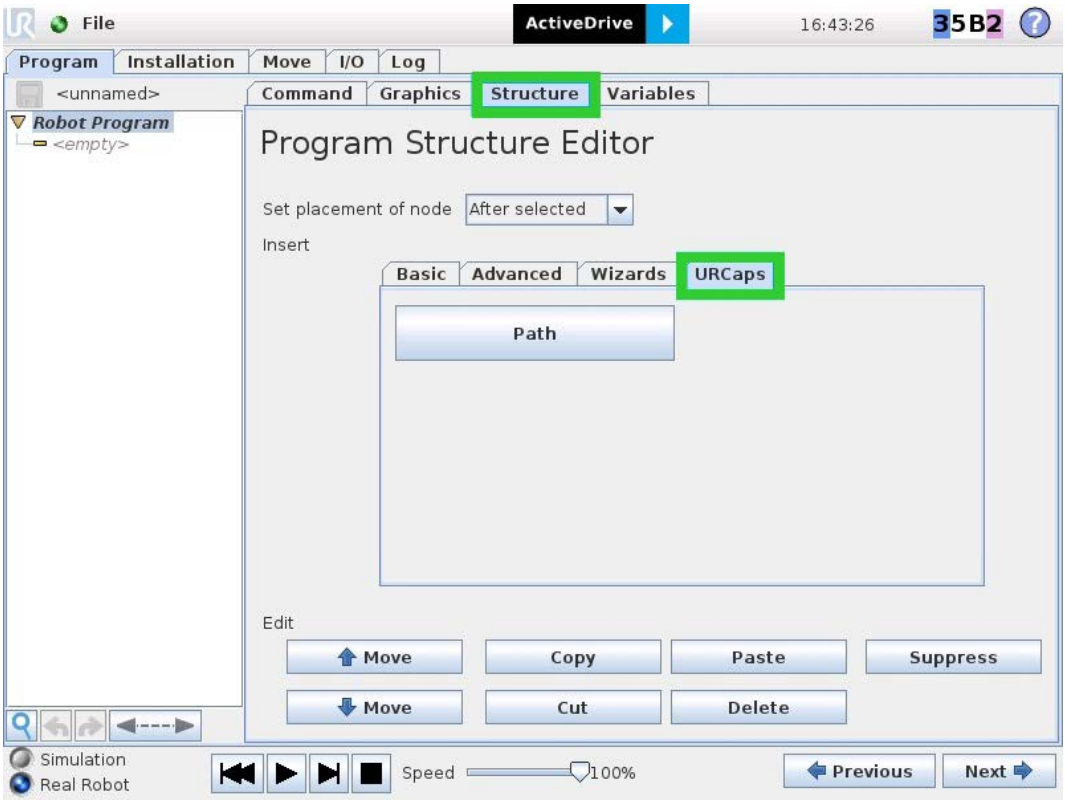
Model UR računala	CB1	CB2	CB3	CB3.1
Legacy Driver paket	nekompatibilno	kompatibilno	kompatibilno	kompatibilno
ActiveDrive alatna traka	nekompatibilno	kompatibilno	kompatibilno	kompatibilno
FT senzor *URCaps* paket (do verzije 1.1.1)	nekompatibilno	nekompatibilno	kompatibilno	kompatibilno
FT senzor *URCaps* paket (verzija 1.2)	nekompatibilno	nekompatibilno	nekompatibilno	kompatibilno

Model računala robota korištenog u radu verzije je CB3 pa je instaliran *URCaps* paket verzije 1.1.1 prateći upute *Robotiq* priručnika za senzor te je unaprijeđena verzija *PolyScope* sučelja na verziju 3.5 prateći upute *Universal Robots* priručnika za robota. Nakon instalacije *URCaps* paketa u *PolyScope* sučelju postaje dostupna *ActiveDrive* alatna traka (vidljiva na vrhu sučelja na slici 3.8.) pomoću koje se mogu koristi funkcije paketa (npr. praćenje trajektorije primjenom sile na senzoru).



Slika 3.8. Instalirani *URCaps* paket unutar *PolyScope* sučelja

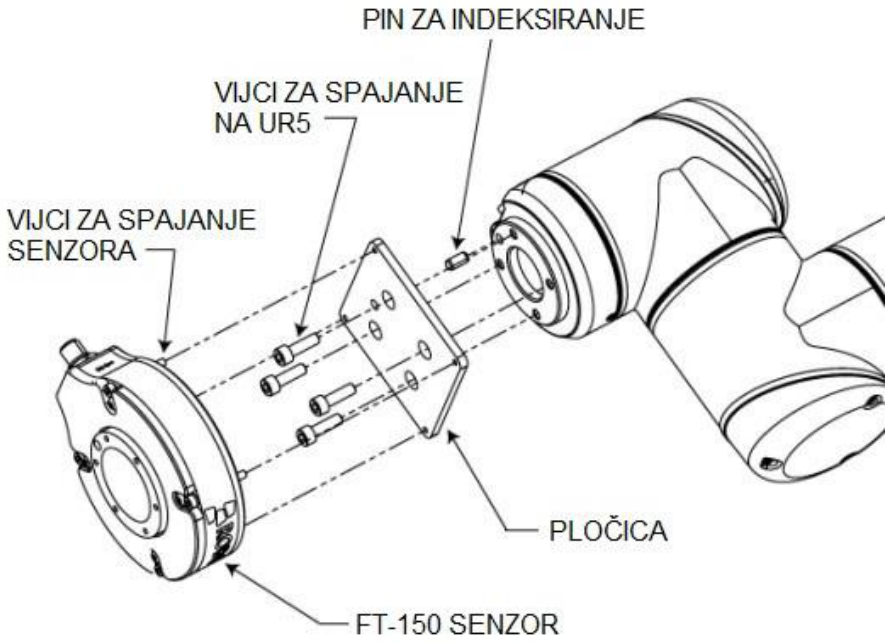
Također postaje dostupna funkcija snimanja trajektorije (ili puta) koju se može uključiti direktno u program na privjesku za učenje (Slika 3.9).



Slika 3.9. Funkcija snimanja puta

3.2.2. Kalibracija senzora

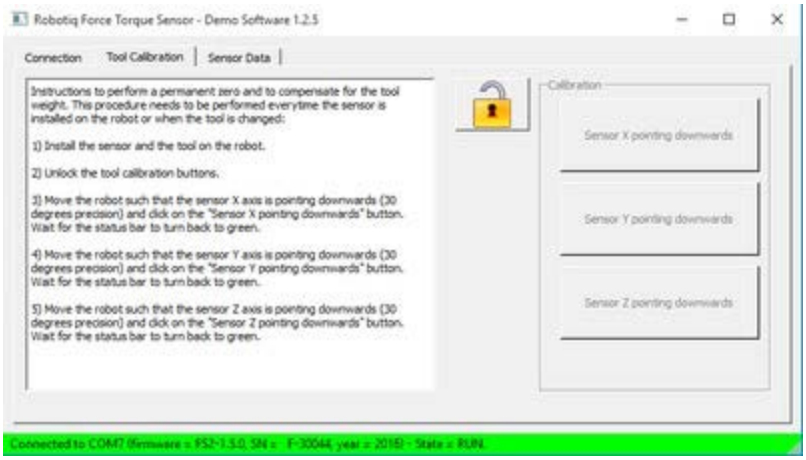
Senzor je potrebno kalibrirati svaki put nakon postavljanja senzora na robot ili mijenjanja alata na senzoru jer se svakim pritezanjem vijaka stvaraju neravnomjerna unutarnja naprezanja u senzoru što uzrokuje kriva očitanja, shema postavljanja senzora na robota prikazana je slikom 3.10.



Slika 3.10. Hardverska instalacija senzora

Postoje dvije opcije za kalibriranje senzora, takozvane privremena i permanentna kalibracija. Privremena kalibracija jednostavno anulira sva očitanja za trenutnu orijentaciju senzora, dok permanentna kalibracija nastoji negirati utjecaj mase alata uslijed djelovanja sile gravitacije za sve orijentacije senzora. Permanentnu kalibraciju potrebno je napraviti jednom nakon svakog mijenjanja alata, dok privremena kalibracija služi za anuliranje zanošenja signala (eng. *drift*) i preporučeno ju je napraviti prije svakog pokretanja programa robota.

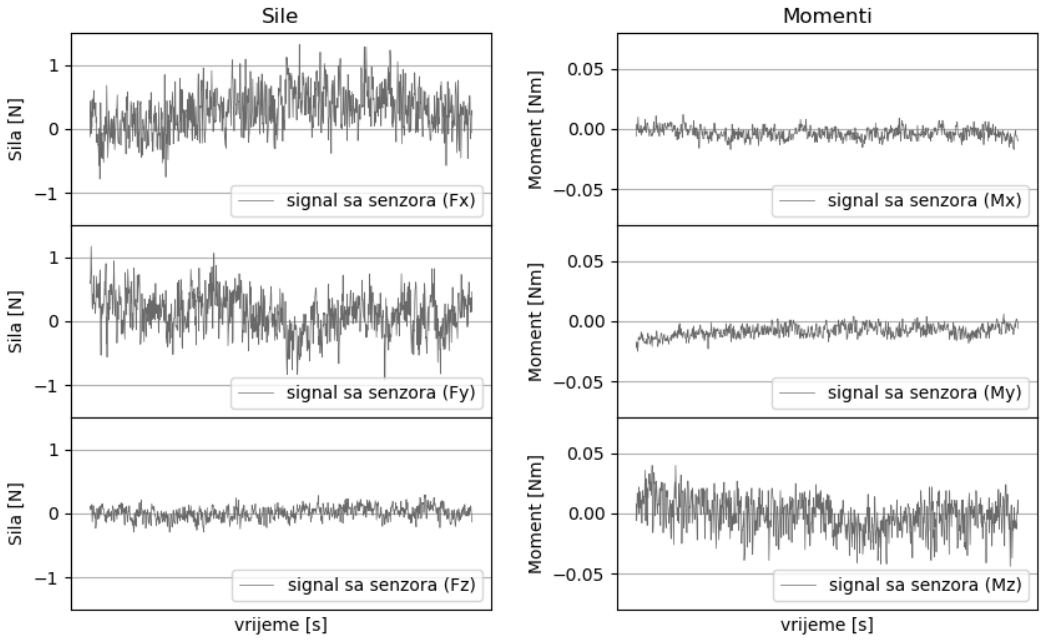
Verzija 1.1.1 *URCaps* paketa nema opciju za permanentnu kalibraciju senzora unutar *PolyScope* sučelja pa je potrebno na eksterno računalo (npr. laptop) skinuti i pokrenuti demonstracijski program sa stranice proizvođača *Robotiq* te senzor spojiti USB konektorom. Robot, odnosno senzor, postavlja se u tri orijentacije prateći upute na sučelju programa prikazanim slikom 3.11.



Slika 3.11. Kalibracija senzora putem eksternog računala (laptopa)

3.2.3. Niskopropusni filter (eng. lowpass)

Signal senzora nije podložan elektromagnetskom zračenju okoline jer se obrađuje direktno na senzoru i u digitalnom obliku putuje kablom do računala robota. Unatoč tome, kao i svaki elektromehanički mjerni uređaj, senzor je podložan šumu mjerenja i njegov signal potrebno je obraditi. Količina šuma i zanošenja signala je iz iskustva nešto veća nego što to tvrdi proizvođač u karakteristikama senzora (Tablica 3.2). Zanošenje signala sile od jednog do drugog dana može iznositi i do 6 N, a količina šuma prikazana je slikom 3.12. gdje je provedeno uzorkovanja signala kalibriranog i neopterećenog senzora u stacionarnom položaju u trajanju od 10 sekundi.



Slika 3.12. Uzorkovanje senzora pri 100 Hz u trajanju od 10 sekundi

Za kvalitetan prihvat i rukovanje predmetom u prostoru potreban je signal sa što manje šuma, odnosno sa što manje visokofrekvencijskih oscilacija. U računalno orijentiranim regulacijskim

sustavima česta je upotreba diskretiziranog niskopropusnog filtera prvog reda [12]. Prijenosna funkcija niskopropusnog filtra u kompleksnoj domeni dana je izrazom:

$$\mathrm{H}(s)=\frac{\mathrm{Y}(s)}{\mathrm{U}(s)}=\frac{1}{T_f s+1}=\frac{1}{\frac{1}{\omega_b}s+1}=\frac{1}{\frac{1}{2\pi f_b}s+1}, \tag{3.1}$$

gdje su:

$\mathrm{H}(s)$ – prijenosna funkcija filtera
$\mathrm{Y}(s)$ – izlaz filtera
$\mathrm{U}(s)$ – ulaz filtera
T_f – vremenska konstanta [s],
ω_b – širina frekvencijskog pojasa (eng. *bandwidth*) [rad/s],
f_b – širina frekvencijskog pojasa [Hz].

Radi računalne implementacije filter dan izrazom (3.1) potrebno je diskretizirati (prebaciti u diskretno područje), metoda diskretizacije koja se često koristi je metoda unazadne diferencijacije (eng. *backward differentiation*). Kao parametar filtera korištena je vremenska konstanta T_f i iz (3.1) dobije se:

$$\left(T_f s+1\right)\mathrm{Y}(s)=\mathrm{U}(s), \tag{3.2}$$

$$T_f s\mathrm{Y}(s)+\mathrm{Y}(s)=\mathrm{U}(s). \tag{3.3}$$

Prebaci li se izraz (3.3) iz kompleksnog u vremensko područje inverznom Laplaceovom transformacijom obaju strana dobije se:

$$T_f\dot{\mathrm{y}}(t)+\mathrm{y}(t)=\mathrm{u}(t). \tag{3.4}$$

Sadašnji trenutak (ili korak) vremena može se predstaviti s t_k tako da:

$$T_f\dot{\mathrm{y}}(t_k)+\mathrm{y}(t_k)=\mathrm{u}(t_k). \tag{3.5}$$

Vremenska derivacija $\dot{\mathrm{y}}(t_k)$ se metodom unazadne diferencijacije mijenja u:

$$\dot{\mathrm{y}}(t_k)=\frac{\mathrm{y}(t_k)-\mathrm{y}(t_{k-1})}{T_s}, \tag{3.6}$$

gdje su:

t_{k-1} – bivši trenutak vremena,
T_s – vrijeme uzorkovanja [s].

Uvrsti li se (3.6) u (3.5), dobije se:

$$T_f\frac{\mathrm{y}(t_k)-\mathrm{y}(t_{k-1})}{T_s}+\mathrm{y}(t_k)=\mathrm{u}(t_k), \tag{3.7}$$

$$\mathrm{y}(t_k)=\frac{T_f}{T_f+T_s}\mathrm{y}(t_{k-1})+\frac{T_s}{T_f+T_s}\mathrm{u}(t_k), \tag{3.8}$$

a predstavi li se novi parametar filtra a kao:

$$a=\frac{T_s}{T_f+T_s}, \tag{3.9}$$

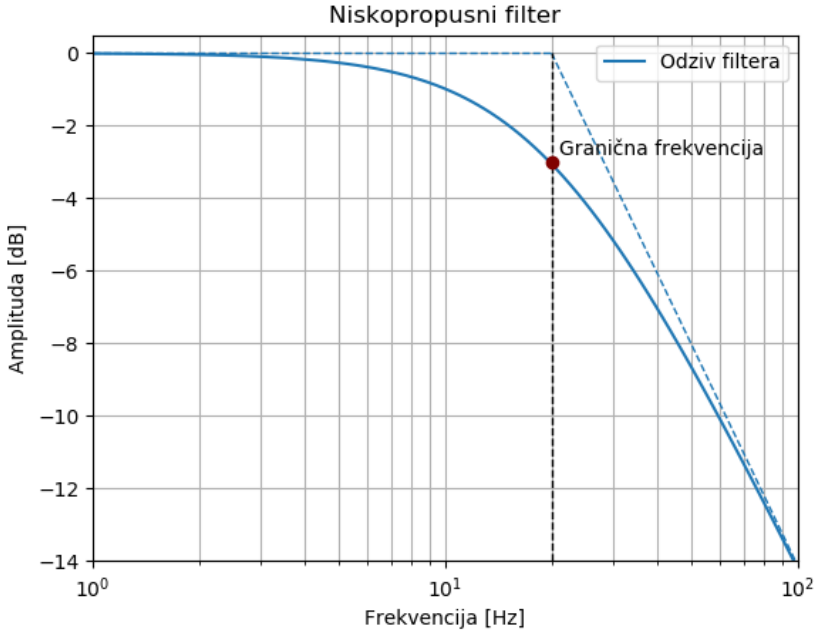
izraz (3.8) može se jednostavnije pisati kao:

$$\mathrm{y}(t_k) = (1 - a)\mathrm{y}(t_{k-1}) + a\mathrm{u}(t_k). \tag{3.10}$$

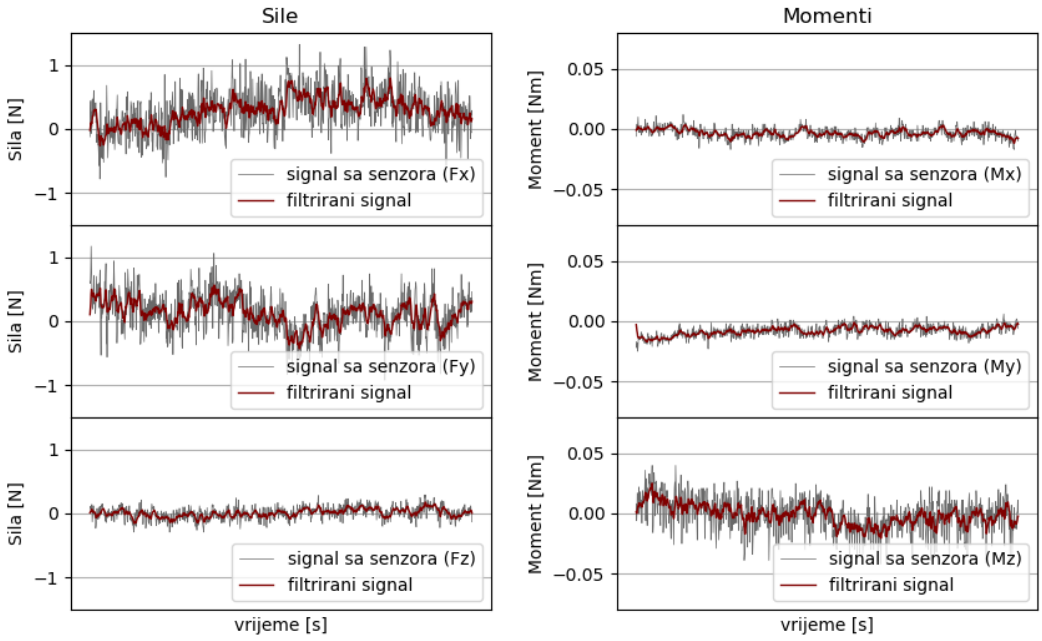
Vrijeme uzorkovanja T_s mora biti znatno manje od vremenske konstante filtera T_f kako bi se postiglo zadovoljavajuće filtriranje visokih frekvencija, u pravilu to je:

$$T_s \leq \frac{T_f}{5}. \tag{3.11}$$

Izraz za diskretizirani niskopropusni filter (3.10) lako se implementira u kodu programa, a slikama 3.13. i 3.14. prikazana je njegova karakteristika i djelovanje na signal senzora.



Slika 3.13. Karakteristika niskopropusnog filtera prvog reda



Slika 3.14. Primjena niskopropusnog filtera

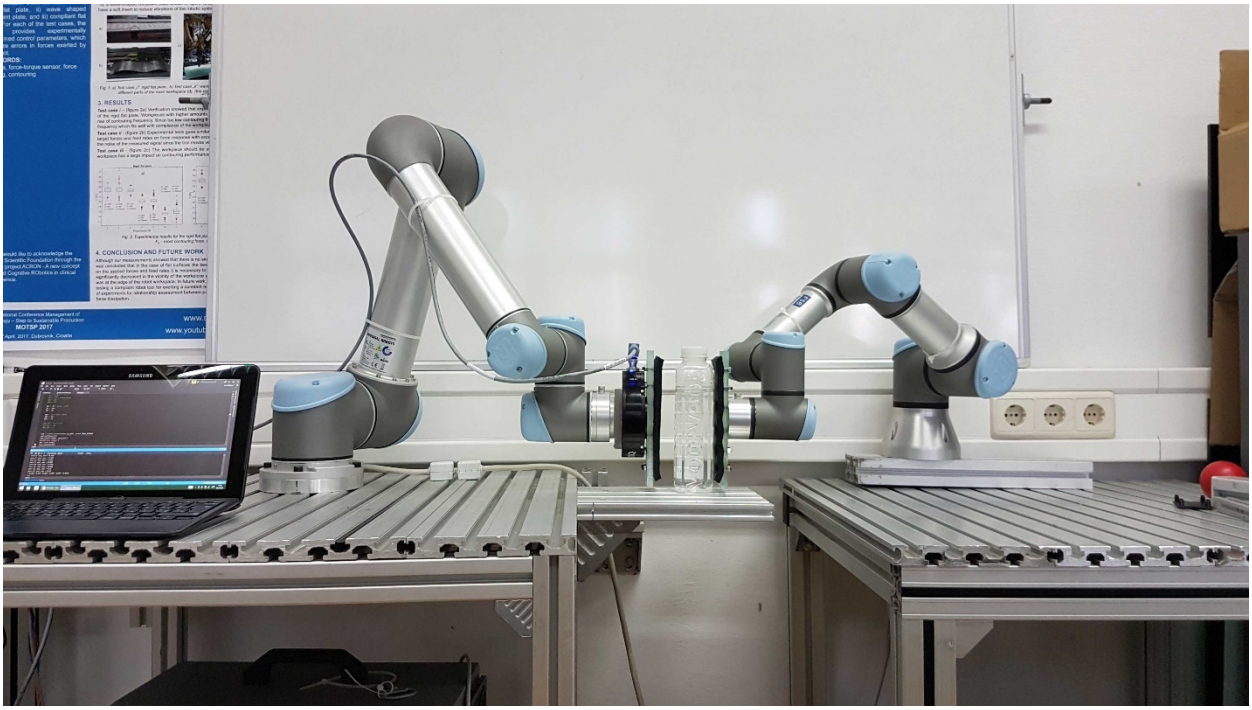
3.3. Radna okolina

Kao predmet rukovanja, zbog oblika i tvrdoće, odabrana je četvrtasta PET boca mineralne vode od 0,5 litara proizvođača VODAVODA d.o.o. (Slika 3.15).

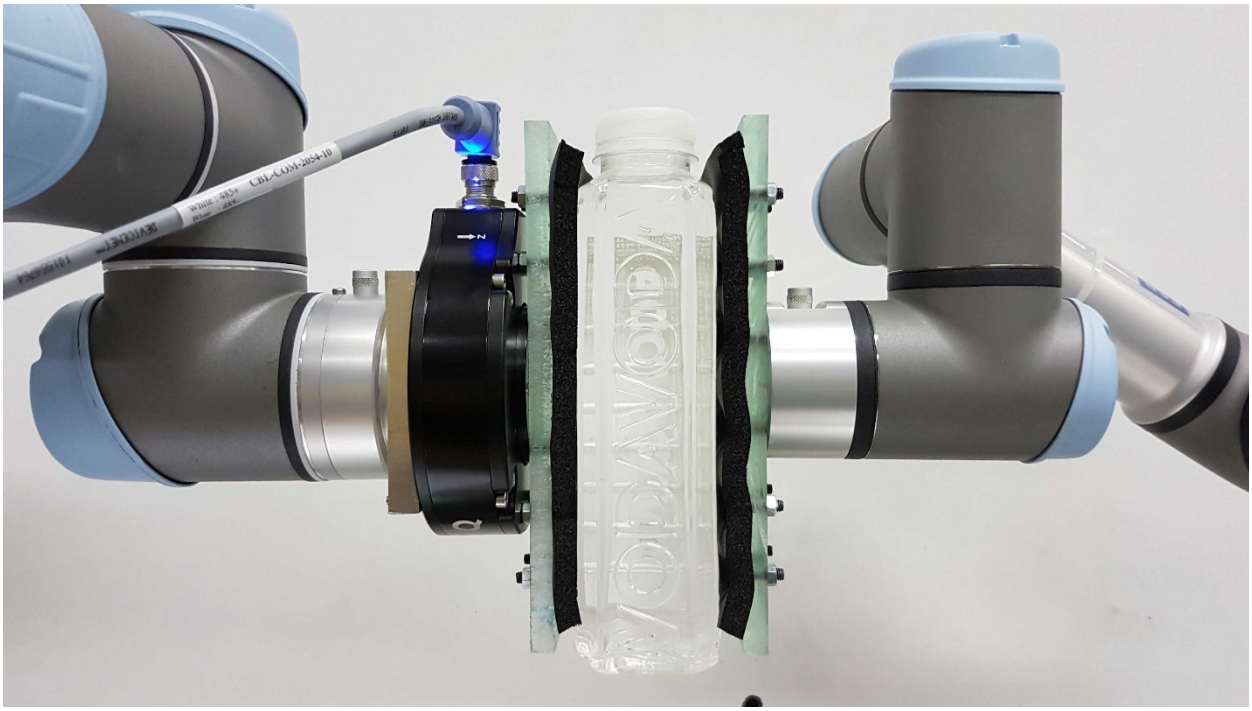
Slika 3.15. VODAVODA [13]

Slikom 3.16. prikazana je radna okolina sustava, veći UR5 robot, sa senzorom sile na prihvatnici, i manji robot UR3 postavljeni su da im se područja utjecaja preklapaju, a između njih postavljen

je predmet rukovanja. Program upravljanja izvršava se na prijenosnom računalu na stolu kraj UR5 robota. Radi prigušenja oscilacija sile prihvata predmeta i povećanja površine trenja, na senzor sile UR5 robota i prihvatnicu UR3 robota postavljena je akrilna pločica sa pričvršćenom spužvom za pakiranje (Slika 3.17.).



Slika 3.16. Radna okolina



Slika 3.17. Prihvat predmeta

4. Programsko upravljanje UR5 robotom

Iako je robotom UR5 moguće upravljati kroz *PolyScope* programsko sučelje dostupno na privjesku za učenje, ono je prilično limitirano za kompleksnije zadatke. Program zadan privjeskom za učenje izvršava se liniju po liniju s predviđenim vremenom za zalet i usporavanje robota od početne do konačne zadane točke (eng. *waypoint*). Ovo je smislena odluka jer izbjegavanjem trzaja uslijed naglih pokretanja i zaustavljanja robota motori i zupčanici robota manje se oštećuju, ali zato je fina kontrola gibanja ograničena. Naredbe za pokretanje robota također se ne mogu dinamički mijenjati dok je robot u pokretu, što znači da se robot obavezno zaustavlja na kraju svake naredbe.

4.1. Komunikacija s robotom

Komunikaciju s robotom moguće je uspostaviti preko TCP protokola koristeći jedan od pokrenutih poslužitelja (eng. *server*) koji su dostupni na portovima (eng. *socket*):

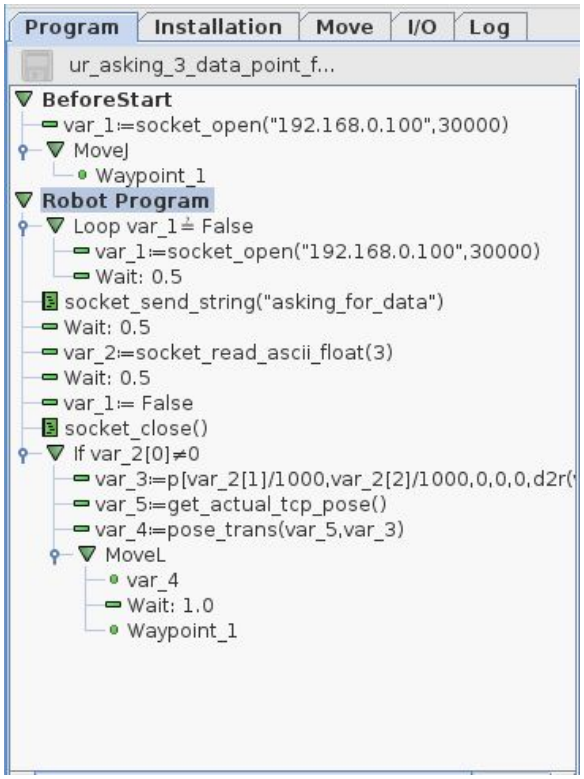
- 29999 – server nadzorne ploče (eng. *dashboard*),
- 30001 – primarni server,
- 30002 – sekundarni server,
- 30003 – *real time* server.

Slanje naredbi direktno putem servera nadzorne ploče nije moguće, namjena servera je postavljanje i kontrola izvršavanja gotovih programa na jednom ili više umreženih robota, neke naredbe su:

- load<program.urp> – učitavanje programa
- play – pokretanje programa
- stop – zaustavljanje programa
- pause – pauziranje programa
- shutdown – gašenje robota

Primarni, sekundarni i *real time* serveri koriste se za direktno upravljanje robotom slanjem naredbi i primanje informacija o stanju robota.

Osim postojećih servera, podržan je i klijent – poslužitelj (eng. *client – server*) način rada koristeći neki proizvoljni port koji nije u uporabi (npr. port 12345 ili 30000). U primjeru na slici 4.1. robot u ulozi klijenta u petlji osluškuje stanje veze sa eksternim računalom na zadanom portu. Nakon ostvarivanja veze, naredbom *socket_send_string(“”)* šalje se string nekog sadržaja i povratni podatak pohranjuje se u varijablu za daljnju uporabu (u ovom slučaju to su informacije o koordinatama predmeta) [14].



Slika 4.1. Program na privjesku za učenje (robot kao klijent) [14]

Eksterno računalo u ulozi poslužitelja uspostavlja vezu i zaprima poruke od robota, ovisno o sadržaju poruke odlučuje što će poslati nazad robotu. Primjer programa na eksternom računalu, pisan u *Python* programskom jeziku, prikazan je ispod [14].

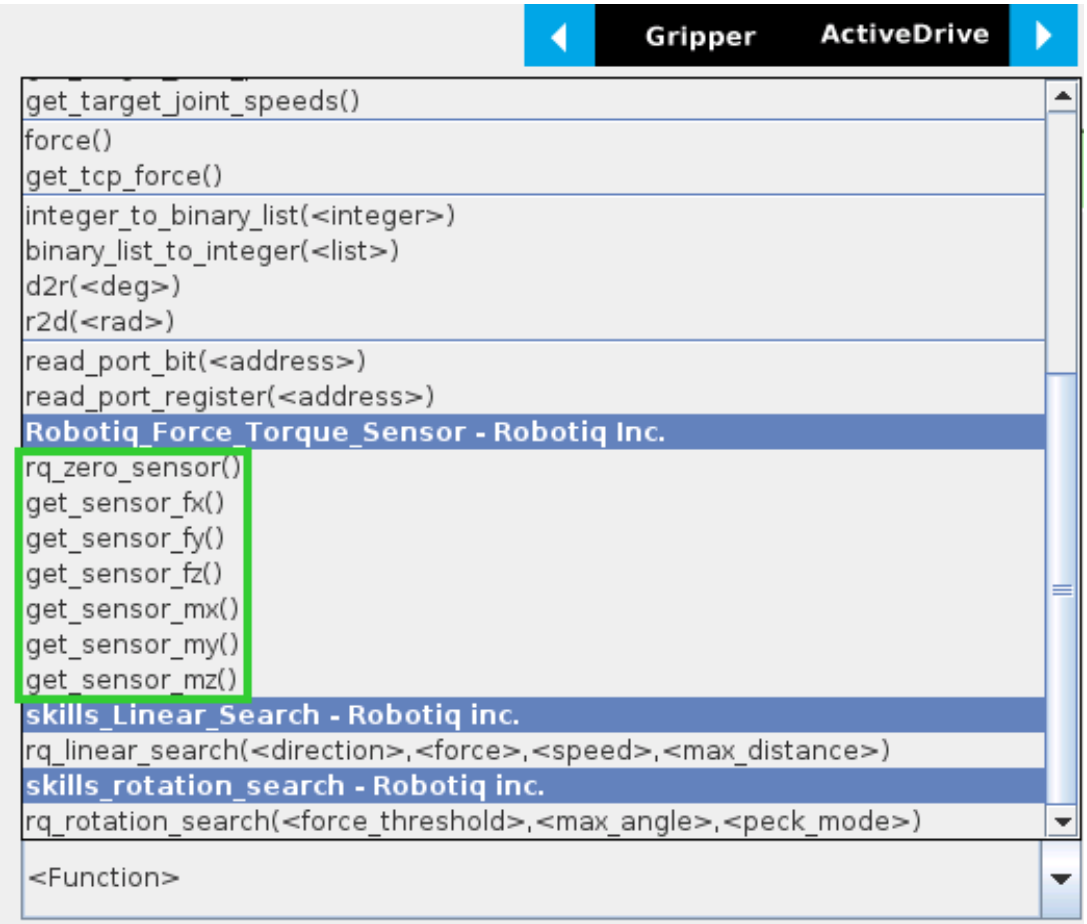
```
# Echo client program
import socket
import time
HOST = "192.168.0.100" # The remote host
PORT = 30000 # The same port as used by the server
print "Starting Program"
count = 0

while (count < 1000):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    s.bind((HOST, PORT)) # Bind to the port
    s.listen(5) # Now wait for client connection.
    c, addr = s.accept() # Establish connection with client.
    try:
        msg = c.recv(1024)
        print msg
        time.sleep(1)
        if msg == "asking_for_data":
            count = count + 1
            print "The count is:", count
            time.sleep(0.5)
            print ""
            time.sleep(0.5)
            c.send("(200,50,45)");
            print "Send 200, 50, 45"
    except socket.error as socketerror:
        print count
c.close()
s.close()
print "Program finish"
```

Klijent – poslužitelj način rada moćan je jer omogućuje razmjenu varijabli između programa robota i programa eksternog računala. Podatke prikupljene sa senzora spojenih na računalo robota

moguće je slati na eksterno računalo, a moguće je i upravljati robotom koristeći podatke dobivene eksternim vizijskim sustavom ili drugim senzorima koji nisu podržani od strane *URCaps* programskog paketa robota.

Ovaj način rada nije korišten jer funkcije za čitanje stanja senzora (prikazane slikom 4.2.) nisu bile dostupne na privjesku za učenje unatoč unaprjeđenju softverske verzije sučelja robota na najnoviju verziju za vrijeme izrade rada (*PolyScope* v3.5), problem je vjerojatno u kompatibilnosti *URCaps* paketa sa starijim modelom robota (Tablica 3.3).



Slika 4.2. Funkcije za čitanje stanja senzora [11]

Osim očitanjem preko privjeska za učenje, podatke sa senzora moguće je prikupljati i putem toka podataka (eng. *stream*) kojeg serijskom vezom senzor šalje računalu robota, a dostupan je na portu 63351 frekvencijom od 100 Hz. Dostupan je i, zbog jednostavnosti implementacije odabran, port 63350 kojim se očitanja senzora dobavljaju na zahtjev s naredbom *socket_send_string("READ DATA")*. Port za tok podataka 63351 nespretan je za korištenje ako je program sporiji od toka podataka ili ako program ima pauze u zaprimanju očitanja jer robot sprema nedostavljene podatke u međuspremnik (eng. *buffer*) i, umjesto svježih, ispostavlja ih kada program ponovo zatraži očitanje.

Komunikacija s primarnim i sekundarnim serverima odvija se pri 10 Hz, dok se komunikacija s *real time* serverom odvija pri 125 Hz. Sva tri servera mogu primati pojedinačne *URScript* naredbe kao i cijele programe od eksternog računala. Poslani program izvršava se čim je zaprimljen zadnji red programa (odnosno naredba *end*), a pojedinačne naredbe izvršavaju se trenutno bez obzira je li robot u toku izvršavanja prethodne naredbe ili miruje. Svaka naredba poslana putem eksternog računala mora završavati sa oznakom novog reda „\n" (eng. *enter*). Mogućnost prekida izvršavanja naredbi neophodna je za realizaciju zadatka, a poželjna je i što veća brzina komunikacije, stoga je odabrana komunikacija preko *real time* servera (port 30003).

Od trenutka ostvarivanja veze s *Real time* serverom, putem porta 30003, računalo robota šalje konstantan tok podataka s informacijama o stanju robota s veličinom svake poruke od 1108 bajtova. Poruka sadrži razne sistemske informacije, podatke o koordinatama, brzinama, akceleracijama, silama, momentima, temperaturama i druge informacije o zglobovima robota kao i razne proračune internog računala.

4.2. Upravljanje pozicijom ili upravljanje brzinom

Glatko kretanje robota kroz više zadanih točki može se postići uporabom *blend* parametra r pri pomicanju robota *movej* ili *movel* naredbama [15].

```
movej(q, a=1.4, v=1.05, t=0, r=0)
movel(pose, a=1.2, v=0.25, t=0, r=0)
```

Naredba *movej* predstavlja linearno kretanje s obzirom na unutarnje koordinate (zglobove) robota iz trenutnog položaja u položaj zadan parametrom q (parametar q predstavlja vektor upravljanih koordinata $\mathbf{q}$ i zadan je kao $[\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]$). *movel* naredba predstavlja kretanje prihvatnice robota linearno u odnosu na vanjske koordinate zadane parametrom *pose* (parametar *pose* zadan je kao $\mathrm{p}[\mathrm{x}, \mathrm{y}, \mathrm{z}, \mathrm{R_x}, \mathrm{R_y}, \mathrm{R_z}]$). Obje naredbe primaju parametre željene brzine v i željene akceleracije a, kao i *blend* radijus r koji omogućuje prijelaz u sljedeću liniju koda i glatko pomicanje zadanim lukom u sljedeću zadanu točku.

Naredbe *movej* i *movel* prikladne su za *pick and place* tip zadatka jer za njihovu uporabu sve točke kretanja moraju biti unaprijed poznate, naredba se također ne može prekinuti ili joj se izmijeniti parametri usred izvršavanja. Za uspješno praćenje trajektorije predmeta uz primjenu konstantne sile potrebno je moći dinamično mijenjati parametre naredbi kretnje s obzirom na očitanja senzora. Također, pošto su buduće točke trajektorije predmeta nepoznate zbog same prirode zadatka, robotom je praktičnije upravljati u terminima brzine prihvatnice nego pozicijom prihvatnice.

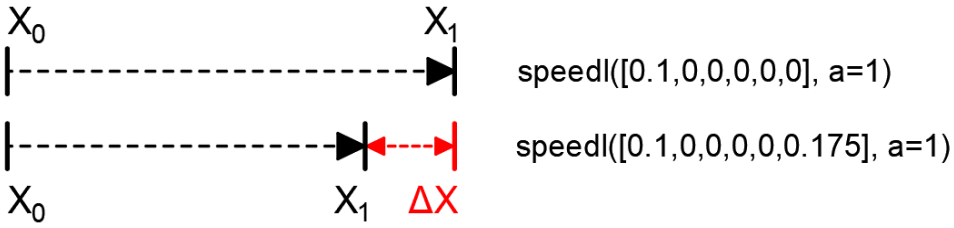
Računalo robota uz pomoć inkrementalnih enkodera prati kutove zakreta zglobova robota i, koristeći poznati kinematički model robota, računa inverznu kinematiku. Znači da je iz toka

podataka kojeg robot šalje moguće direktno očitati položaj upravljanih (**q**) i vanjskih (**r**) koordinata robota bez potrebe za ručnim računanjem direktne ili inverzne kinematike. Također, dostupni su i podaci o brzinama i akceleracijama u upravljanim i vanjskim koordinatama bez potrebe ručnog računanja Jacobijeve matrice, tako je moguće direktno upravljanje brzinama robota uporabom naredbi *speedj* i *speedl*.

```
speedj(qd, a, t)
speedl(xd, a, t, aRot='a')
```

Obje naredbe primaju tražene vrijednosti brzina u svojim respektivnim koordinatnim sustavima (qd predstavlja brzine upravljanih koordinata $\dot{\mathbf{q}} = [\dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5, \dot{q}_6]$ dok xd predstavlja brzine vanjskih koordinata $\dot{\mathbf{r}} = [\dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z]$), željenu akceleraciju a (u slučaju *speedl* naredbe tu je i neobavezna kutna akceleracija *aRot*) i maksimalno vrijeme izvršavanja t (neobavezno).

Uz uporabu *speedl* naredbe upravljanje robotom svodi se na očitavanje sila i momenata sa senzora, pretvaranje sila i momenata u vektor željenih linearnih i kutnih brzina i, uz pomoć matrice transformacija, orijentiranje vektora brzina u vanjske koordinate robota. Nažalost, uporaba *speedl* naredbe iziskuje nejednoliki pomak linearnih koordinata uslijed istovremenog linearnog i kutnog pomaka, simbolički prikazano slikom 4.3.



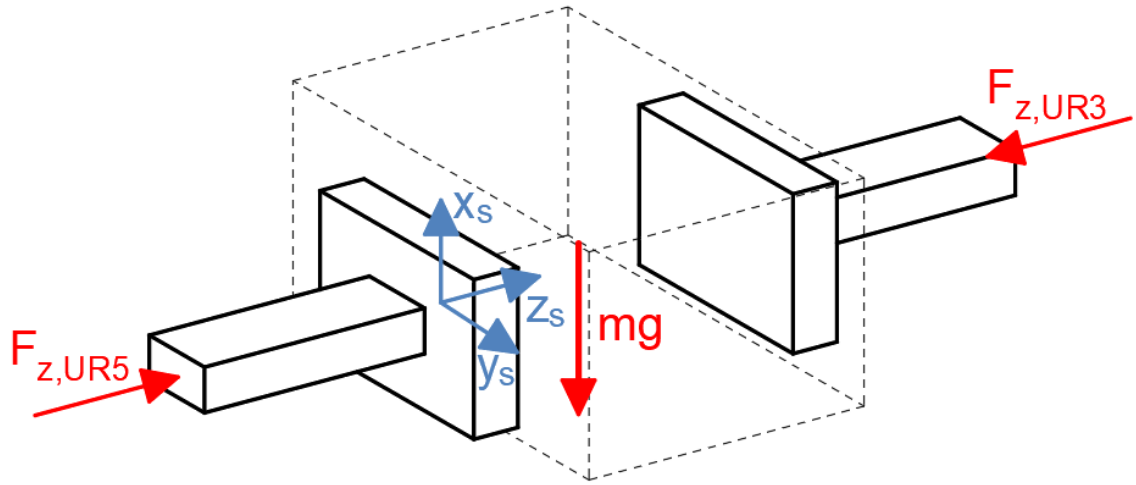
Slika 4.3. Uporaba *speedl* naredbe

U oba slučaja robotu je zadana linearna brzina $\dot{x} = 100\ \mathrm{mm/s}$ i željena akceleracija $a = 1\ \mathrm{mm/s^2}$, jedino je u drugom slučaju robotu zadana i kutna brzina $\omega_z = 10°/\mathrm{s}$ ($aRot = a$ ukoliko *aRot* nije zadano). U slučaju kada se naredba koristi za istovremeno translacijsko i rotacijsko gibanje dolazi do odstupanja ΔX od očekivanog linearnog pomaka po osi x i zbog nastale greške praćenja predmet ispada iz prihvata. Količina odstupanja ΔX varira s podešavanjem a i *aRot* vrijednosti, kao i s omjerom linearnih i kutnih brzina, ali neka greška uvijek je prisutna.

Naredba *speedj* pogodnija je upravljanje robotom jer služi za direktno upravljanje brzinom svakog zgloba i ne radi razliku između linearnih i kutnih brzina. Za korištenje *speedj* naredbe, i upravljanje robotom, potrebno je raditi očitanja trenutnih položaja zglobova **q** i pomoću kinematičkog modela robota računati inverznu Jacobijevu matricu te vektor dobivenih brzina vanjskih koordinata $\dot{\mathbf{r}}$ postaviti u poziciju i orijentaciju prihvatnice robota uz pomoć matrice transformacije.

Eksterno računalo u svakom koraku izvršavanja programa robotu šalje naredbu *speedj* sa svim pripadajućim izračunatim brzinama i ostalim parametrima, primitkom naredbe robot prekida izvršavanje prethodne naredbe i započinje izvršavanje s novim parametrima bez zaustavljanja robota.

5. Distribuirano upravljanje dvoručnim robotskim rukovanjem



Slika 5.1. Prihvat predmeta

Slikom 5.1. prikazana je shema prihvata predmeta, senzor S sa ishodištem u $\{x_S, y_S, z_S\}$ postavljen je na vrh prihvatnice UR5 robota i očitava sile i momente u svim smjerovima. Kako bi predmet ostao u prihvatu tokom kretanja očitana sila pritiska $F_{z,UR5} = F_{z,UR3}$ treba biti jednaka referentnoj sili $F_{z,ref} = 10$ N. UR5 robot silu pritiska $F_{z,UR5}$ treba održavati konstantnom, što znači da treba prilaziti i odmicati se od predmeta s obzirom na očitanu silu. Kada senzor u smjeru osi z_S očita silu malo veću od referentne ($F_{z,UR5} > F_{z,ref}$), robot UR5 treba se polagano odmicati od predmeta sve dok sila pritiska ponovo nije jednaka referentnoj. Ako senzor očita puno veću silu od referentne ($F_{z,UR5} \gg F_{z,ref}$), tada se UR5 treba brže odmicati. Odnos sile i brzine robota može se dakle smatrati linearnim, brzina je tada jednaka umnošku sile (dio sile manji ili veći od referentne sile u osi z_S ili sila različita od nule u smjerovima osi x i y) i nekog faktora pretvorbe sila k_S (eksperimentalno odabrano $k_S = 0{,}01$). Odnos momenta i kutne brzine također se može smatrati linearnim pa je kutna brzina jednaka umnošku momenta i faktora pretvorbe momenata k_M (eksperimentalno odabrano $k_M = 0{,}3$).

Da bi se predmet moglo podignuti i održavati u zraku potrebno je negirati utjecaj sile izazvane masom predmeta, $mg = 5$ N. Pošto smjer djelovanja sile mg ovisi o orijentaciji predmeta, njen utjecaj negira se koristeći matricu rotacije $\mathbf{R}_6^0$:

$$\mathbf{F}_S = \mathbf{F}_S + \mathbf{R}_6^0 \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}, \tag{5.1}$$

gdje je $\mathbf{F}_S$ vektor senzorom očitanih sila.

Kako bi bilo moguće čisto translacijsko gibanje u svim smjerovima potrebno je negirati i momente koji nastaju uslijed smičnog pomaka predmeta u smjeru osi x_S i y_S oduzimanjem vektorskog

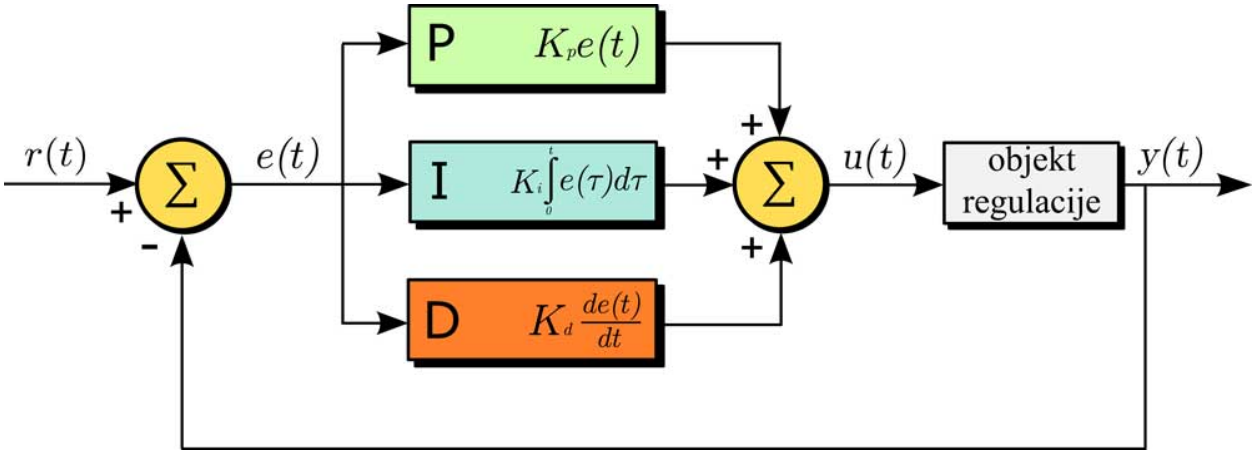
produkta vektora udaljenosti i vektora očitanih sila $\mathbf{F}_S$ (0,015 m visina je akrilne pločice i spužve postavljene na senzor, prikazano slikom 3.17.)

$$\mathbf{M}_S = \mathbf{M}_S - \begin{bmatrix} 0 \\ 0 \\ 0{,}015 \end{bmatrix} \times \mathbf{F}_S \tag{5.2}$$

gdje je $\mathbf{M}_S$ vektor senzorom očitanih momenata.

5.1. PID regulacija

UR5 robot inicijalno nije u kontaktu s predmetom pa je potrebno robotu poslati naredbu da mu polako prilazi nekom brzinom. Zbog kašnjenja odziva, duljine trajanja proračuna, numeričkih grešaka u proračunu, inercije robota, nesavršenosti fizikalnih komponenti robota i drugih faktora, u trenutku dodira s predmetom robot nastavlja prilaziti predmetu i sila prihvata postaje veća od željene. U sljedećem koraku programa očitana sila prihvata veća je od željene i program šalje naredbu robotu da se odmakne od predmeta. Robot se odmiče od predmeta sve dok sila prihvata ne padne ispod željene i ponovo je potrebno prilaziti predmetu. Rezultat je nezadovoljavajuće oscilatorno praćenje predmeta što znači da za uspješnu regulaciju sustava nije dovoljna sama pretvorba očitanih sila i momenata u linearne i kutne brzine.



Slika 5.2. Shema PID regulacije [16]

Proporcionalno-integralno-derivacijski (PID) regulatori u širokoj su upotrebi u industriji zbog učinkovitosti i jednostavnosti implementacije za većinu potreba regulacije [16], slikom 5.2. shematski je prikazana struktura PID regulatora, gdje su:

$r(t)$ – referentna (željena) veličina,
$e(t)$ – greška između referentne $r(t)$ i izlazne veličine $y(t)$,
$u(t)$ – ulazna veličina objekta regulacije,
$y(t)$ – izlazna veličina objekta regulacije,
K_p – proporcionalno pojačanje,
K_i – integracijsko pojačanje,
K_d – derivacijsko pojačanje,

a opći oblik PID regulatora opisan u vremenskoj domeni glasi:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}, \quad \textbf{(5.3)}$$

PID regulator kontinuirano računa vrijednost greške $e(t)$ kao razliku referentne $r(t)$ i izlazne veličine $y(t)$ i uz pomoć proporcionalnog, integralnog i derivacijskog djelovanja regulira veličinu ulazne varijable $u(t)$ sve dok se greška ne svede na minimum. Proporcionalni dio regulatora P predstavlja trenutnu grešku sustava (npr. ako je greška velika i pozitivna, izlaz iz regulatora bit će velik i pozitivan). Integralni dio regulatora I predstavlja prijašnje greške (npr. ako izlazna varijabla nije dovoljna da smanji grešku, greška će se nakupiti tijekom vremena i natjerat će sustav da poveća izlaznu varijablu). Derivacijski dio regulatora D predviđa buduće greške temeljene na brzini promjene. Ispod je napisan pseudokod za implementaciju PID regulatora [17].

```
bivša_greška = 0
integral = 0
K_p = ⋯
K_i = ⋯
K_d = ⋯
while (True):
        greška = referentna_veličina − izmjerena_veličina
        integral = integral + (greška · vrijeme_iteracije)
        derivacija = (greška − bivša greška)/vrijeme_iteracije
        izlaz = K_p · greška + K_i · integral + K_d · derivacija
        bivša_greška = greška
        sleep(vrijeme_iteracija)
```

Pri regulaciji nije nužno korištenje svih djelovanja, u praksi se P član najčešće koristi u kombinaciji s jednim ili oba druga člana (PI, PD, PID) ovisno o financijskim mogućnostima, zahtjevima regulacije i kompleksnosti implementacije. Željeni odziv postiže se podešavanjem K_p, K_i i K_d parametara PID regulatora, tablicom 5.1. prikazano je djelovanje svakog člana na regulaciju sustava.

Tablica 5.1. Utjecaj parametara PID regulatora na sustav [17]

Parametar	Vrijeme odziva	Prebačaj odziva	Vrijeme smirivanja	Greška u stacionarnom stanju	Stabilnost sustava
$\boldsymbol{K_p}$	smanjuje	povećava	malo utječe	smanjuje, ali ne uklanja	pogoršava
$\boldsymbol{K_i}$	smanjuje	povećava	povećava	potpuno uklanja	pogoršava
$\boldsymbol{K_d}$	malo utječe	smanjuje	smanjuje	nema učinka u teoriji	poboljšava ako je K_d mali

Stabilnost sustava osnovni je zahtjev regulacije, dok je važnost drugih zahtjeva ovisna o namjeni regulatora. U ovom radu nije pretjerano važno eliminiranje greške u stacionarnom stanju jer je

predmet u upotrebi dovoljno elastičan da ga veća sila neće deformirati, a izabrana referentna sila dovoljno je velika da predmet neće ispasti ako je sila prihvata nešto manja. Znatno je važnije reduciranje vremena smirivanja i količine prebačaja odziva, vrijeme odziva također je važno u trenu pokretanja ili promjene smjera predmeta.

Podešavanje parametara PID regulatora može se izvršiti na više načina, *Ziegler-Nicholsova* metoda, *Tyreus Luyben* metoda, *Cohen-Coon* metoda itd. Svaka metoda dobra je na svoj način i ima svoje nedostatke, ali zbog velike nelinearnosti sustava odbrana je ručna eksperimentna metoda. Ručnom metodom eliminiraju se K_i i K_d parametri postavljanjem u nulu i povećava se K_p parametar dok sustav ne počne reagirati oscilatorno. Zatim se parametar K_d, koji smanjuje prebačaj i vrijeme smirivanja sustava, povećava sve dok ne prestane imati pozitivan utjecaj na odziv. Parametri K_p i K_d naizmjenično se postepeno povećavaju dok se ne postigne najbolji mogući odziv. Regulator s previše proporcionalnog djelovanja može izazvati velike prebačaje, dok previše derivacijskog djelovanja čini sustav podložnijim na šum. Parametar K_i dodaje se na kraju da eliminira grešku u stacionarnom staju. Eksperimentalno se je pokazalo da je pomak u smjeru z osi senzora najkritičniji, predmet lako ispada s vrlo malom količinom prebačaja, a oscilacije tokom kretanja imaju tendenciju previše oslabiti silu prihvata. Pomaci u ostalim i zakreti u svim osima manje su podložni ovim problemima, a to je odraženo i izborom parametara regulatora (tablica 5.2.).

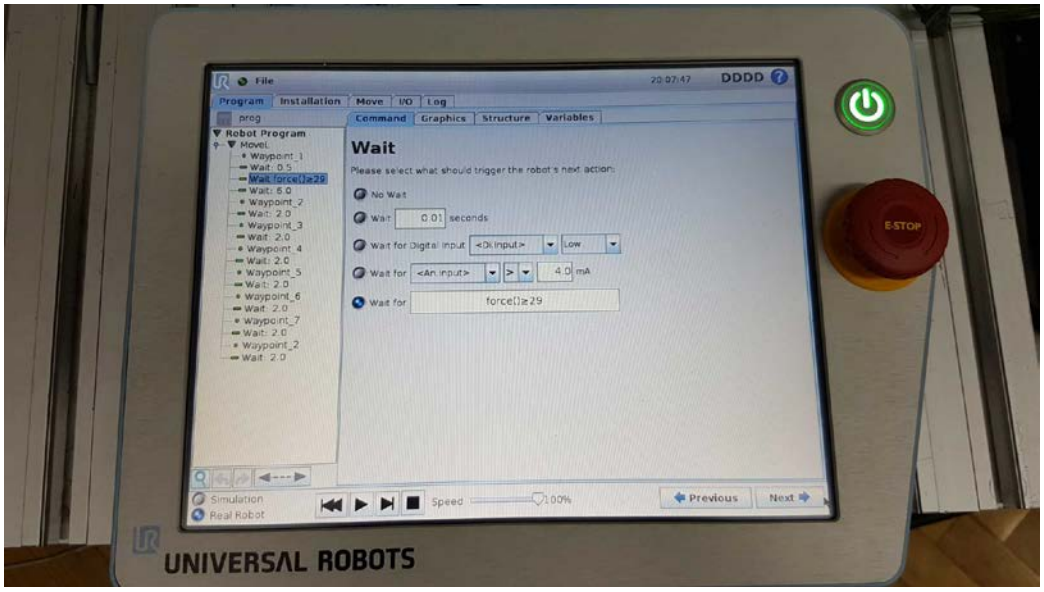
Tablica 5.2. Parametri PID regulatora

Parametar	Sile u z osi	Ostale sile i momenti
K_p	0,2	0,45
K_i	0,01	0,005
K_d	0,12	0,15

5.2. Primjena dvoručnog rukovanja

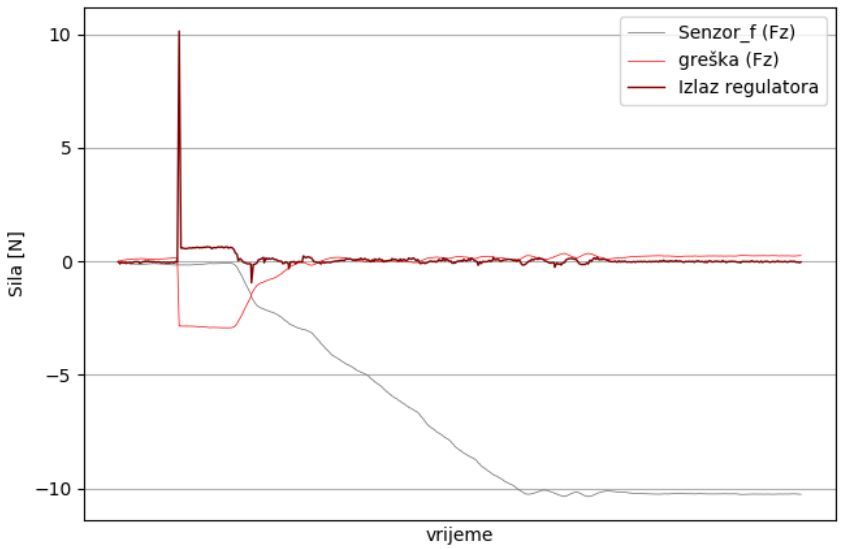
Roboti UR3 i UR5 dovedeni su u početnu točku kao što je prikazano slikom 3.16., i pokrenuti su zasebni programi svakog robota. Program UR3 robota napisan je na privjesku za učenje i prikazan slikom 5.3. Njegova zadaća je da čeka detekciju sile pritiska koristeći internu funkciju za očitanje sile u zglobovima $Wait\ force() \geq 29$, a zatim prolazi kroz predefinirane točke u prostoru te se vraća u početni položaj. Prag sile od 29 N za UR3 robota dobiven je eksperimentalno i nekada ga je potrebno podesiti jer robot nekonzistentno estimira sile. Uporaba *Wait* naredbe u trajanju od

dvije sekunde nakon dolaska u svaku točku nije nužna, ali brzina kretanja UR3 robota usporena je radi izbjegavanja naglih promjena stanja koje PID regulator nije u stanju pratiti.



Slika 5.3. Program UR3 robota

Program upravljanja UR5 robotom uz pomoć PID regulacije pisan je u *Python* programskom jeziku i prikazan je u prilogu. Slikom 5.4. prikazan je graf filtrirane očitane sile i odziva PID regulatora u smjeru z osi za početak rukovanja predmetom.

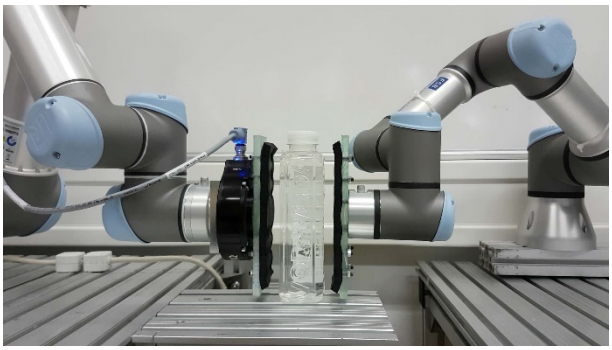


Slika 5.4. Prikaz očitane sile i odziva PID regulatora za z os

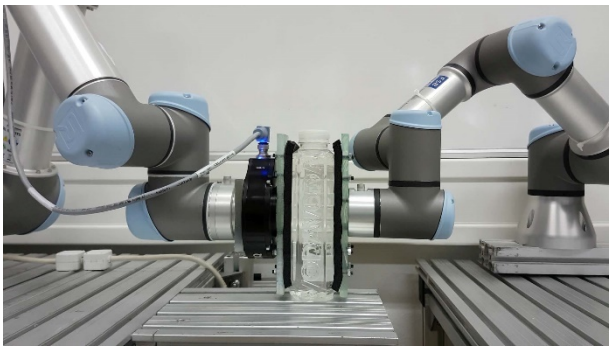
Pri pokretanju programa robot je nepomičan jer je zadana referentna sila jednaka nuli pa je i greška između referentne i očitane sile jednaka nuli. Pritiskom na tipku na tipkovnici postavlja se mala referentna vrijednost sile od -3 N koja uzrokuje jednaku grešku od -3 N, PID regulator tada, u nastojanju da spusti grešku na nulu, počne davati pozitivan signal i robot počinje primicati

predmetu. Na početku primicanja vidljiv je veliki skokoviti odziv regulatora, uzrokovan zamahom integracijskog člana (eng. *integrator windup*), koji je lako programski ignorirati ili ograničiti. Kada robot dođe u prihvat s predmetom i greška padne na nulu, referentna sila postepeno se povećava do odabrane sile prihvata od 10 N. Kretanje robota ograničeno je na pozitivno i negativno pomicanje u smjeru osi z sve dok nije postignuta sila prihvata, nakon čega počinje kompenzacija utjecaja gravitacije i momenata izazvanih poprečnim silama kao što je opisano s (5.1) i (5.2). UR3 robot tada treba detektirati povećanu silu pritiska i početi prolaziti kroz svoje predefinirane točke, slikama 5.5. od a) do p) prikazano je rukovanje predmetom.

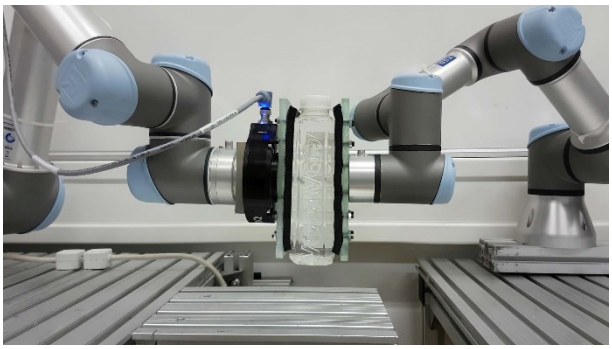
Vidljivo je da je rukovanje predmetom uspješno realizirano, UR5 robot (lijevo) uspješno prati trajektoriju zadanu UR3 robotom (desno) s predmetom u prihvatu. Koordinatni sustav senzora u donjim slikama usmjeren je kao na slici 5.1. Zakretanje predmeta oko x osi senzora kao i linearno kretanje u smjeru y osi senzora nešto je teže pratiti jer je površina prihvata (površina trenja) s predmetom u y osi najmanja. Iz slike 5.5.d) vidljivo je da UR5 zaostaje za zakretom predmeta oko x osi senzora, greška se ispravlja tek sljedećom rotacijom oko y osi kao što slika 5.5.e) pokazuje. Problem praćenja također je prikazan slikama 5.5.i) – n) gdje predmet skoro ispada iz prihvata uslijed istovremene translacije po y osi i rotacije oko x osi. Rukovanje u ostalim smjerovima ne predstavlja problem te je praćenje trajektorije predmeta zadovoljavajuće.



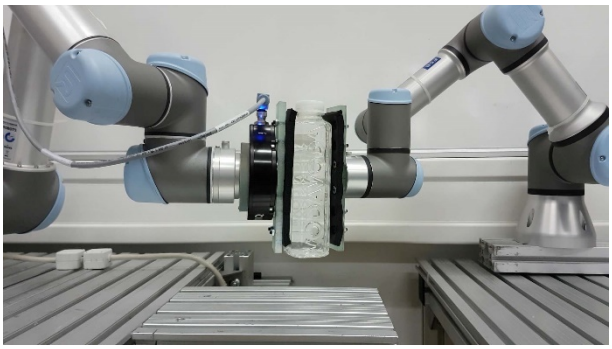
a) prije prihvata



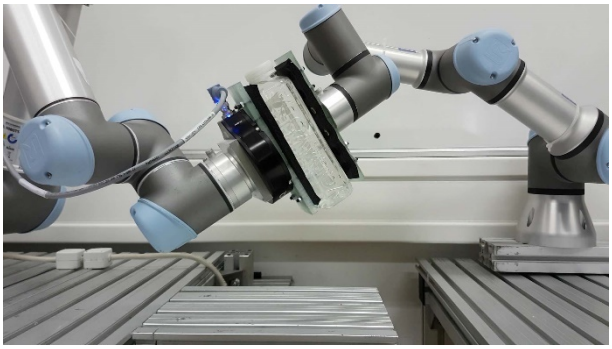
b) predmet u prihvatu



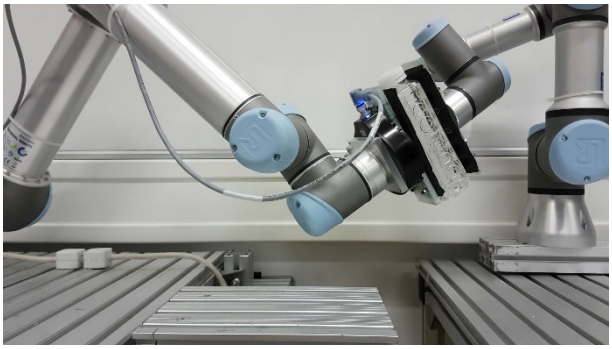
c) podizanje predmeta



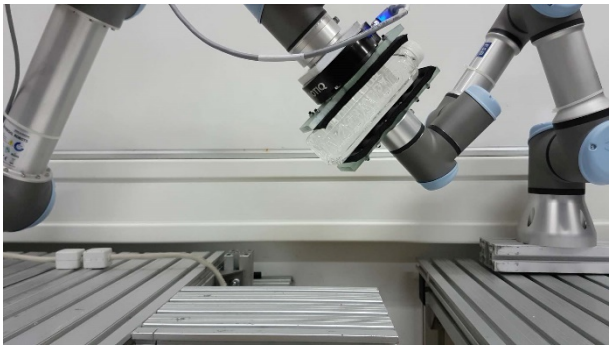
d) zakretanje u dubinu (oko x osi senzora)



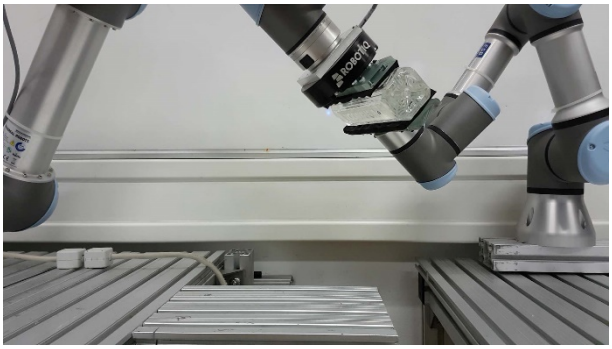
e) zakretanje oko y osi senzora



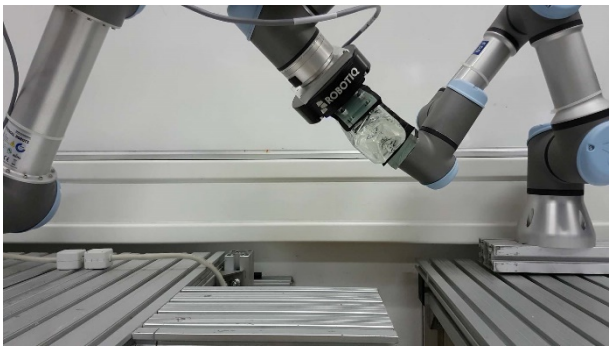
f) translacija po dvije osi



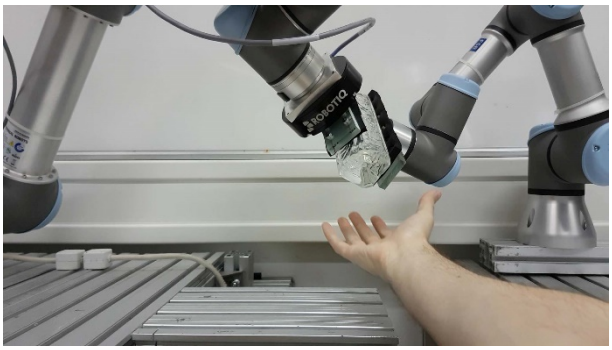
g) zakretanje oko y osi



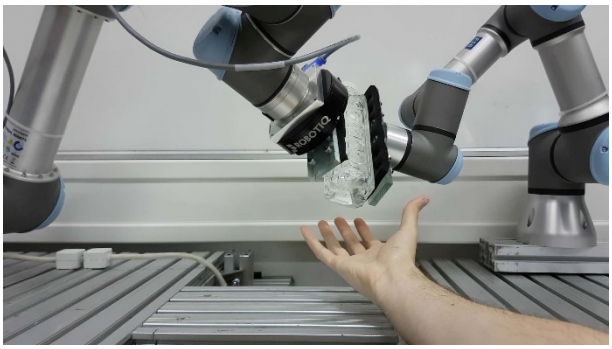
h) zakretanje oko z osi



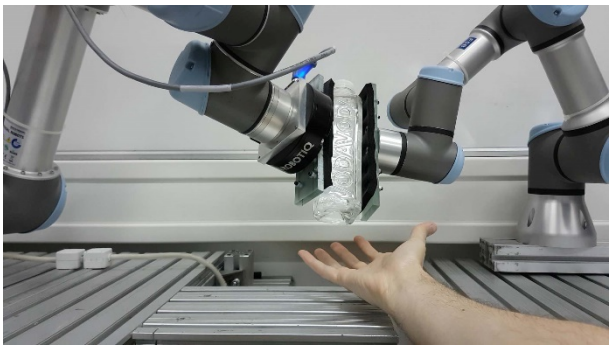
i) zakretanje i translacija



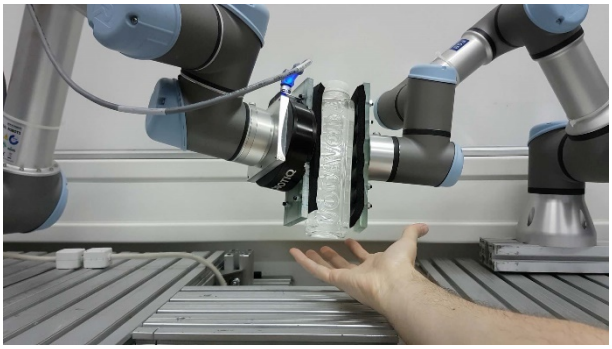
j) zakretanje i translacija



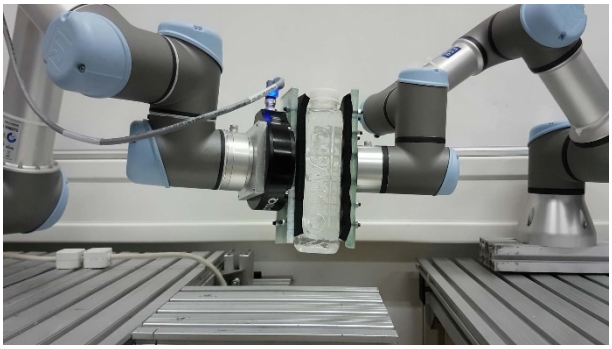
k) zakretanje i translacija



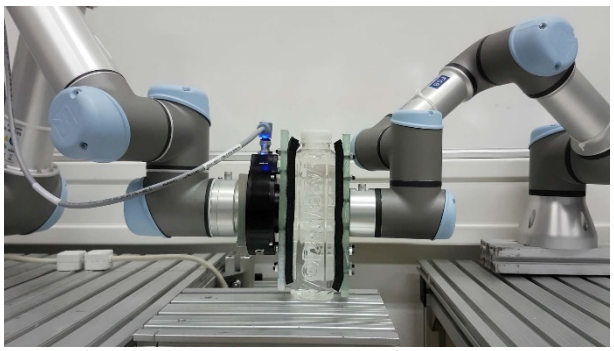
l) zakretanje i translacija



m) zakretanje i translacija



n) zakretanje i translacija



p) povratak u početni položaj

Slika 5.5. Primjer rukovanja uz PID regulaciju

6. Zaključak

Distribuirano dvoručno rukovanje predmetom u prostoru uz pomoć šesteroosnog senzora sila i momenata i bez međusobne komunikacije između robota uspješno je realizirano. Upravljanje UR5 robotom, koji prati trajektoriju UR3 robota, ostvareno je putem naredbi poslanih s eksternog računala koristeći TCP protokol. Problemi u praćenju nastaju prilikom pomaka uzduž ili zakretom oko uske dimenzije predmeta zbog manje veličine kontaktne površine u tom smjeru, boljim izborom prihvatnice za uske predmete moglo bi se pribjeći ovom problemu.

Brzina kretanja robota ograničena je jer PID regulacija najbolje radi s linearnim sustavima, a pri većim brzinama inercija robotske ruke može izazvati nelinearno ponašanje. Robotska ruka sama po sebi nije linearni sustav, ali može se smatrati linearnim sustavom budući da interno računalo robota kompenzira za gravitaciju i trenje u zglobovima. Bolje praćenje moglo bi se postići daljnjim podešavanjem PID parametara, ali PID regulacija nije optimalna za ovakav sustav jer parametri regulatora nisu dinamički promjenjivi i vrijede za samo jedan određeni slučaj za koji je regulator podešen.

Kada bi matematički model robota bio poznat, bolji rezultati, poput veće brzine, mogli bi se postići uporabom nekog drugog linearnog (*linear quadratic regulator*, *model predictive control*) ili nelinearnog adaptivnog (*sliding mode control*) regulatora. Robot bi također mogao brže reagirati na nagle promjene očitanih sila kada bi zadana akceleracija sustava a bila varijabilna (npr. izražena kao razlika trenutne i tražene brzine robota) umjesto da je konstantna (u radu $a = 1\ \mathrm{m/s^2}$).

Dinamiku robota, uz poznavanje masa segmenata te kinetičkih i potencijalnih energija sustava, moguće je opisati jednom od metoda modeliranja sustava (npr. *Euler-Lagrange* metodom), ali dinamika internog upravljačkog sklopa robota je i dalje nepoznata i teško ju je modelirati. Robotom također nije moguće upravljati direktno preko momenata u zglobovima ukoliko se ne kompajlira upravljački program na nižoj C-API razini koji se pokreće direktno na računalu robota. Druga mogućnost modeliranja sustava je korištenje jedne od metoda identifikacije nelinearnih sustava (NARMAX, *subspace identification*), gdje se sustav pobuđuje poznatim ulaznim signalom (uglavnom sinusoidnim signalima varirajućih frekvencija) i pomoću mjerenog odziva računa se matematički model sustava.

LITERATURA

[1] http://cyberneticzoo.com/robots/1937-the-robot-gargantua-bill-griffith-p-taylor-australiancanadian/, *Meccana Gargantua*, pristupljeno lipanj 2018.

[2] https://en.wikipedia.org/wiki/Unimation, *Unimation*, pristupljeno lipanj 2018.

[3] https://ifr.org/img/uploads/Presentation_market_overviewWorld_Robotics_29_9_2016.pdf, *IFR Press Conference*, pristupljeno lipanj 2018.

[4] **Šurina, Tugomir; Crneković, Mladen**: *Industrijski roboti*, Školska knjiga, Zagreb, 1990.

[5] **Craig, J. John**: *Indtroduction to Robotics: Mechanics and Control (3rd Edition)*, Pearson Prentice Hall, Upper Saddle River, New Jersey, SAD, 2004.

[6] https://youtu.be/VjsuBT4Npvk, *Forward and Inverse Kinematics*, pristupljeno lipanj 2018.

[7] **Keating, Ryan**: *UR5 Inverse Kinematics*, Johns Hopkins University, Baltimore, SAD, 2016.

[8] https://youtu.be/Ibl709gEX3g, *PUMA inverse kinematics*, pristupljeno lipanj 2018.

[9] https://www.universal-robots.com/products/ur3-robot/, *Universal robots UR3*, pristupljeno lipanj 2018.

[10] **Deur, Joško**: *Elektromotorni servopogoni*, podloge s predavanja.

[11] https://robotiq.com/support/ft-300-force-torque-sensor/downloads-instruction-manual, *FT 150 Instruction Manual*, pristupljeno lipanj 2018.

[12] **Haugen, Finn**: *Discretization of simulator, filter, and PID*, TechTeach, 10. svibanj 2010.

[13] http://www.photoorange.rs/food-beverages/, *Food & Drinks*, slika boce, pristupljeno lipanj 2018.

[14] http://www.zacobria.com/universal-robots-knowledge-base-tech-support-forum-hints-tips/knowledge-base/script-client-server/, *Universal-Robots Script Client-Server example*, pristupljeno lipanj 2018.

[15] **Universal Robots**: *The URScript Programming Language v3.5.1*, Odense, Danska, 2017.

[16] https://en.wikipedia.org/wiki/PID_controller, *PID controller*, pristupljeno lipanj 2018.

[17] http://robotsforroboticists.com/pid-control/, *PID Control (with code), Verification, and Scheduling*, pristupljeno lipanj 2018.

[18] https://www.universal-robots.com/media/1514597/101081_199901_ur5_technical_details_web_a4_art03_rls_eng.pdf, tehničke specifikacije UR5 robota, pristupljeno lipanj 2018.

[19] https://youtu.be/SefTCXrpL8U, *Finding the Jacobian*, pristupljeno lipanj 2018.

PRILOZI

I. CD-R disk

II. Kôd zadatka pisan u *Python* programskom jeziku

```
import socket
import time
import binascii
import matplotlib.pyplot as plt
import numpy as np
import copy
import keyboard

np.set_printoptions(suppress=True)
sin = np.sin
cos = np.cos
pi = np.pi

def e(temp): # služi za slanje vrijednosti na robot
    temp = temp.encode("utf-8")
    return temp

def parse(byte): # obrada byte podataka s robota, pretvorba u string
    byte = str(byte)
    byte = byte.replace(')', '')
    temp = [0 for i in range(0, 6)]
    temp[0] = float(byte[byte.find('(') + 1:byte.find(',')])
    zarez = 0
    for i in range(1,6):
        zarez = byte.find(',', zarez + 1)
        temp[i] = round(float(byte[zarez + 1:byte.find(',', zarez + 1)]), 4)
    return temp

def kalibracija(): # kalibracija senzora
    N = 200 # average od N vrijednosti
    iter = 0 # koliko puta je pokušano nuliranje
    senzor = np.array([5 for i in range(0, 6)])
    while max(abs(senzor[0:3])) > 0.15: # nuliranje na srednju vrijednost manju od 0.15
        print("zero {}. put, max = {}".format(iter, max(abs(senzor[0:3]))))
        sen.sendall(b'socket_send_string("SET ZRO")')
        time.sleep(0.2)
        senzor = np.array([0.0 for i in range(0, 6)])
        for j in range(0, N):
            sen.sendall(b'socket_send_string("READ DATA")')
            temp = np.asarray(parse(sen.recv(100)))
            senzor += temp
            time.sleep(0.01)
        senzor = np.round(senzor/N, 3)
        iter += 1
    print(senzor)
    return

def pozicija(): # dobavljanje kutova zglobova i TCP koordinata iz modbusa
    regq, regqq, regtcp, regspd = "", "", "", ""
    mod.sendall(b"\x00\x04\x00\x00\x00\x06\x00\x03\x01\x0e\x00\x06") #registri 270-275 (kutovi q)
    regq = mod.recv(1024)
    regq = (binascii.hexlify(regq[9:]))
    mod.sendall(b"\x00\x04\x00\x00\x00\x06\x00\x03\x01\x40\x00\x06") #registri 320-325 (q predznaci)
    regqq = mod.recv(1024)
    regqq = (binascii.hexlify(regqq[9:]))
    mod.sendall(b"\x00\x04\x00\x00\x00\x06\x00\x03\x01\x90\x00\x06") #registri 400-405 (TCP)
    regtcp = mod.recv(1024)
    regtcp = (binascii.hexlify(regtcp[9:]))
    mod.sendall(b"\x00\x04\x00\x00\x00\x06\x00\x03\x01\x9a\x00\x06") #registri 410-415 (TCP brzina)
    regspd = mod.recv(1024)
    regspd = (binascii.hexlify(regspd[9:]))
    q = [i for i in range(0, 6)]
    tcp = [i for i in range(0, 6)]
    spd = [i for i in range(0, 6)]
    for i in range(0, 6):
        if regqq[4*i] == 102: # 102 ako je f, 48 ako je 0 (+ ili -)
            q[i] = round(int(regq[4*i:4*i + 4], 16)/1000 - 2*pi, 4)
        else:
            q[i] = round(int(regq[4*i:4*i + 4], 16)/1000, 4)
        tcp[i] = int(regtcp[4*i:4*i + 4], 16)
        spd[i] = int(regspd[4*i:4*i + 4], 16)
        if tcp[i] < 32768:
            tcp[i] = tcp[i]/10
        else:
            tcp[i] = (tcp[i] - 65535)/10
        if spd[i] < 32768:
            spd[i] = spd[i]/1000
```

```
        else:
            spd[i] = (spd[i] - 65535)/1000
    tcp[0:3] = [round(tcp[x]/1000, 4) for x in range(0, 3)]
    tcp[3:6] = [round(tcp[x]/100, 4) for x in range(3, 6)]
    return q, tcp, spd

def matrica(tcp): # računanje matrice rotacije iz TCP koordinata
    T06 = np.eye(4)
    kut = np.linalg.norm(tcp[3:6])
    vektor = np.array([tcp[3:6]/kut])
    e1, e2, e3 = vektor[0][0], vektor[0][1], vektor[0][2]
    ee = vektor.T.dot(vektor)
    ex = np.array([[0, -e3, e2], [e3, 0, -e1], [-e2, e1, 0]])
    T06[0:3, 0:3] = np.eye(3)*cos(kut) + (1 - cos(kut))*ee + ex*sin(kut)
    T06[:-1, 3] = tcp[0:3]
    T6s = np.array([[0, 1, 0, 0], [-1, 0, 0, 0], [0, 0, 1, 0.05], [0, 0, 0, 1]]) # senzor je
zakrenut za -90° po Z osi i visine 50 mm
    T0s = copy.deepcopy(T06)
    T0s = T06.dot(T6s)
    return T06, T0s

def jacobian(q): # računanje Jacobijeve matrice
    T = np.zeros([6, 4, 4])
    T[:, [0, 1, 2, 3], [0, 1, 2, 3]] = 1
    T0i = copy.copy(T)
    J = np.zeros([6, 6])
    alpha = np.array([pi/2, 0, 0, pi/2, -pi/2, 0])
    a = np.array([0, -0.425, -0.392, 0, 0, 0])
    d = np.array([0.0892, 0, 0, 0.1093, 0.09475, 0.0825])
    d[5] += 0.05 # treba dodati visinu senzora d[5] += 0.05 i zakret senzora q[5] += -pi/2
    q[5] += -pi/2
    for i in range(0, 6):
        T[i] = np.array([[cos(q[i]), -sin(q[i])*cos(alpha[i]),  sin(q[i])*sin(alpha[i]),
a[i]*cos(q[i])],
                         [sin(q[i]),  cos(q[i])*cos(alpha[i]), -cos(q[i])*sin(alpha[i]),
a[i]*sin(q[i])],
                         [0,          sin(alpha[i]),           cos(alpha[i]),           d[i]],
                         [0,          0,                       0,                       1]])
        T0i[i] = T0i[i - 1, :, :].dot(T[i, :, :])
    J[:, 0] = np.append(np.cross([0, 0, 1], T0i[5, 0:3, 3]), [0, 0, 1])
    for i in range(0, 5):
        J[:, i + 1] = np.append(np.cross(T0i[i, 0:3, 0:3].dot([0, 0, 1]), T0i[5, 0:3, 3] - T0i[i,
0:3, 3]), T0i[i, 0:3, 0:3].dot([0, 0, 1]))
    np.round(J,3)
    J_inv = np.linalg.inv(J)
    return J_inv

def move(senzor, T0s, J_inv, spd, oldspd, stay, kontakt): # pokretanje robota
    global ref
    v = copy.copy(senzor)
    if abs(v[0]) < 0.4:
            v[0] = 0
    if abs(v[1]) < 0.4:
            v[1] = 0
    if abs(v[2]) < 0.1:
            v[2] = 0
    if abs(v[3]) < 0.1:
            v[3] = 0
    if abs(v[4]) < 0.13:
            v[4] = 0
    if abs(v[5]) < 0.1:
            v[5] = 0
    for i in range(0, 3):
        v[i] *= 0.01
        v[i + 3] *= 0.3
    if ref[2] > -10:
        v[0], v[1] = 0, 0
        v[3:6] = [0, 0, 0]
    print("target spd: {}; actual spd: {}\n".format(np.round(v, 3), np.round(spd, 3)))
    v = J_inv.dot(np.append(T0s[0:3, 0:3].dot(v[0:3]), T0s[0:3, 0:3].dot(v[3:6])))
    a = 1
    if stay == False:
        s.sendall(b"speedj(["
            + e(str(v[0])) + b"," + e(str(v[1])) + b"," + e(str(v[2])) + b","
            + e(str(v[3])) + b"," + e(str(v[4])) + b"," + e(str(v[5]))
            + b"], a=" + e(str(a)) + b", t=0.2)" + b"\n")
    return spd
```

```
def read(): # očitava vrijednosti sa senzora
    sample = 1 # broj uzoraka za srednju vrijednost
    senzor = np.array([0.0 for i in range(0, 6)])
    for i in range(0, sample):
        sen.sendall(b'socket_send_string("READ DATA")')
        temp = np.asarray(parse(sen.recv(1024)))
        senzor += temp
        time.sleep(0.01)
    senzor /= sample
    senzor = [round(senzor[x], 3) for x in range(0, 6)]
    print("senzor raw:", senzor)
    return senzor

def lowpass_A(senzor, oldsenzor_f, t0, t1):
    Ts = t1 - t0 #0.04
    k = (Ts)/(Ts + 10*Ts)
    senzor_f = (1 - k)*oldsenzor_f + k*senzor
    return senzor_f

def lowpass_B(senzor, oldsenzor_f, t0, t1):
    Ts = t1 - t0 #0.04
    k = (Ts)/(Ts + 25*Ts)
    senzor_f = (1 - k)*oldsenzor_f + k*senzor
    return senzor_f

def lowpass_C(senzor, oldsenzor_f, t0, t1):
    Ts = t1 - t0 #0.04
    k = (Ts)/(Ts + 20*Ts)
    senzor_f = (1 - k)*oldsenzor_f + k*senzor
    return senzor_f

def PID(Kp, Ki, Kd, er, er_1, I_1, t0, t1):
    Ts = t1 - t0 #0.04
    I = I_1 + er*Ts
    D = (er - er_1)/Ts
    u = -(Kp*er + Ki*I + Kd*D)
    if abs(u) > 25: # limitiranje regulatora
        u = np.sign(u)*25
    return u, I

def PID_all(Kp, Ki, Kd, er, er_1, I_1, t0, t1):
    Ts = t1 - t0 #0.04
    I = I_1 + er*Ts
    D = (er - er_1)/Ts
    u = -(Kp*er + Ki*I + Kd*D)
    ind = np.where(np.abs(u) > 10) # limitiranje regulatora
    u[ind] = np.sign(u[ind])*10
    return u, I

class flags:
    def flag_stay(key): # postavljanje referentne sile i pokretanje robota
        global ref, stay, kontakt, I
        ref[2] = -3
        kontakt = False
        I = np.array([0. for i in range(6)])
        print("pokretanje")
        stay = not stay

#keyboard.unhook_all()
keyboard.on_press_key('đ', flags.flag_stay, suppress=True)

HOST = "192.168.0.10"  # IP robota
PORT, mod_port, sen_port = 30003, 502, 63350  # robot stream = 30003, modbus = 502, senzor port =
63350 (na zahtjev) ili 63351 (stream)
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.settimeout(1)
s.connect((HOST, PORT))
print("start robot control")
sen = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sen.settimeout(1)
sen.connect((HOST, sen_port))
print("start senzor")
# setup
yo = 0
oldspd = np.array([0.0 for i in range(0, 6)])
Kp = .45 # PID x,y
```

```
Ki = .005
Kd = .15
Kp2 = .2 # PID z
Ki2 = .01
Kd2 = .12
senzor_f = np.array([0. for i in range(6)])
u = np.array([0. for i in range(6)])
u_f = np.array([0. for i in range(6)])
ref = np.array([0. for i in range(6)])
I = np.array([0. for i in range(6)])
error = np.array([0. for i in range(6)])
error_1 = np.array([0. for i in range(6)])
error_2 = np.array([0. for i in range(6)])
contact_count = 0
no_contact_count = 0
# flags
koef = 0.0
stay = True
kontakt = False

kalibracija()

mod = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mod.settimeout(1)
mod.connect((HOST, mod_port))
print("open modbus")
t0 = time.time()
time.sleep(0.001)
with open("D:\\Dropbox\\work work\\diplomski\\python\\PID.txt", "w+") as fajl:
    while True:
        senzor = np.array(read())
        q, tcp, spd = pozicija()
        T06, T0s = matrica(tcp)
        J_inv = jacobian(q)
        ref2 = copy.copy(ref)
        if kontakt:
            if koef < 1:
                koef += .005
            ref2[0:3] = ref[0:3] + T0s[0:3, 0:3].T.dot(np.array([0, 0, -3*koef]).T)
        senzor[3:6] -= np.cross([0, 0, 0.015], senzor[0:3])
        t1 = time.time()
        senzor_f[0:2] = lowpass_B(senzor[0:2], senzor_f[0:2], t0, t1)
        senzor_f[2] = lowpass_C(senzor[2], senzor_f[2], t0, t1)
        senzor_f[3:6] = lowpass_A(senzor[3:6], senzor_f[3:6], t0, t1)
        print("filter:", np.round(senzor_f, 3))
        error_1 = copy.copy(error)
        error = ref2 - senzor_f
        u = copy.copy(senzor_f)
        u[2], I[2] = PID(Kp2, Ki2, Kd2, error[2], error_1[2], I[2], t0, t1)
        u[0:2], I[0:2] = PID_all(Kp, Ki, Kd, error[0:2], error_1[0:2], I[0:2], t0, t1)
        t0 = t1
        if not stay:
            if not kontakt:
                if abs(senzor_f[2]) < 0.5:
                    no_contact_count += 1
                if no_contact_count > 100:
                    no_contact_count = 0
                    ref[2] = 0
                    stay = True
                if spd[2] == 0:
                    contact_count += 1
                else:
                    contact_count = 0
                if contact_count > 15:
                    kontakt = True
                    contact_count = 0
                    no_contact_count = 0
                    print("kontakt ON")
            elif ref[2] > -10:
                ref[2] -= 0.05
            elif abs(senzor[2]) < 0.4:
                no_contact_count += 1
                if no_contact_count > 10:
                    no_contact_count = 0
                    koef = 0
                    stay = True
                    kontakt = False
```

```
                    print("kontakt OFF")
            elif abs(senzor[2]) > 0.4:
                no_contact_count = 0
            oldspd = move(u, T0s, J_inv, spd, oldspd, stay, kontakt)
        fajl.write("\n{} ".format(t1))
        for j in range(6):
            fajl.write("{} {} {} ".format(senzor_f[j], error[j], u[j]))
# plotanje odziva
with open("D:\\Dropbox\\work work\\diplomski\\python\\PID.txt", "r") as fajl:
    data = np.loadtxt(fajl)
    ax = ["" for i in range(6)]
    raspored = {321:321, 322:323, 323:325, 324:322, 325:324, 326:326} # za subplotove
    oznake = {0:"Fx", 1:"Fy", 2:"Fz", 3:"Mx", 4:"My", 5:"Mz", }
    plt.figure("PID odziv", figsize=(15,10))
    for i in range(6):
        ax[i] = plt.subplot(raspored[i + 321])
        if i == 0:
            plt.title("Sile")
        if i == 3:
            plt.title("Momenti")
        plt.plot(data[:, 0], data[:, 1 + i*3], linewidth = 0.5, color = "dimgray", label = "Senzor_f
({})".format(oznake[i]))
        plt.plot(data[:, 0], data[:, 2 + i*3], linewidth = 0.5, color = "r", label = "greška
({})".format(oznake[i]))
        plt.plot(data[:, 0], data[:, 3 + i*3], linewidth = 1, color = "maroon", label = "Izlaz
regulatora")
        plt.grid(True)
        plt.gca().xaxis.set_major_locator(plt.NullLocator()) # brisanje x osi
        plt.legend(loc = "upper right")
        if i < 3:
            plt.ylabel("Sila [N]")
        else:
            plt.ylabel("Moment [Nm]")
    plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=0)
plt.show()
```