Čovjekom potpomognut robotizirani sustav zavarivanja

Jurman, Andrej

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 103

ČOVJEKOM POTPOMOGNUT ROBOTIZIRANI SUSTAV ZAVARIVANJA

Andrej Jurman

Zagreb, veljača 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 103

ČOVJEKOM POTPOMOGNUT ROBOTIZIRANI SUSTAV ZAVARIVANJA

Andrej Jurman

Zagreb, veljača 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 30. rujna 2024.

DIPLOMSKI ZADATAK br. 103

Pristupnik:	**Andrej Jurman (0036513840)**
Studij:	Informacijska i komunikacijska tehnologija
Profil:	Automatika i robotika
Mentor:	izv. prof. dr. sc. Matko Orsag

Zadatak: **Čovjekom potpomognut robotizirani sustav zavarivanja**

Opis zadatka:

Robotizirano zavarivanje u velikim postrojenjima serijske proizvodnje poznat je inženjerski problem. S druge strane, zahtjevno programiranje takvih sustava velika je prepreka primjeni u malim strojarskim radionicama koje gotovo svakodnevno mijenjaju proizvode. Cilj ovog zadatka je razviti sučelje čovjeka i robota kojim će se omogućiti programiranje zavarivanja potpomognuto demonstracijom čovjeka. Potrebno je razviti algoritme planiranja trajektorije koji će optimirati snimljene trajektorije kako bi osigurali gibanje konstantnom brzinom prilikom zavarivanja. U sklopu zadatka potrebno je razviti simulacijsko okruženje za provjeru sustava, te osmisliti sustav za kalibraciju i puštanje u pogon stvarnog sustava. Algoritme je potrebno ispitati u laboratorijskom okruženju.

Rok za predaju rada: 14. veljače 2025.

Sadržaj

Uvod

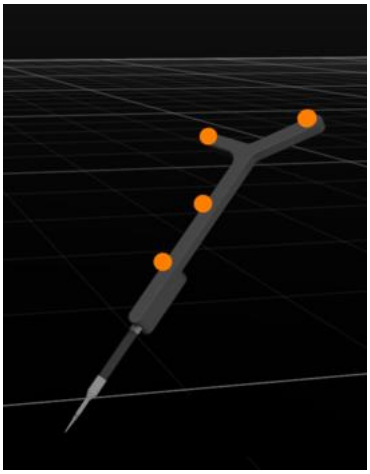
U suvremenoj industriji zavarivanja, robotizirani sustavi postaju ključni za postizanje visoke kvalitete i produktivnosti. Veliki industrijski pogoni već koriste robote za zavarivanje u serijskoj proizvodnji, gdje se sustavi programiraju prema unaprijed definiranim parametrima. S druge strane, u manjim strojarskim radionicama, koje se gotovo svakodnevno suočavaju s promjenama proizvoda, zahtjevno programiranje ovih sustava predstavlja značajnu prepreku. Tradicionalne metode programiranja ne pružaju dovoljno fleksibilnosti te često zahtijevaju specijalizirane stručnjaka za programiranje robota, što dodatno otežava primjenu robotiziranog zavarivanja u ovim okruženjima.

Cilj ovog diplomskog rada je razviti čovjekom potpomognut sustav robotiziranog zavarivanja koji omogućava intuitivno programiranje zavarivanja putem demonstracije. Kroz ovaj pristup, korisnik snima krajnje točke putanje zavarivanja, a razvijeni algoritam planira trajektoriju zavarivanja kako bi se osiguralo gibanje robota konstantnom brzinom tijekom zavarivanja. Konstantna brzina je ključna za postizanje kvalitetnog zavarenog spoja.

Rad se sastoji od dva dijela, prvi je sustav za čovjekom potpomognuto označavanje prostora a drugi je razvoj algoritma za generiranje trajektorije za proces zavarivanja. Ovim radom nastoji se doprinijeti smanjenju tehničkih prepreka pri implementaciji robotiziranih sustava zavarivanja u dinamičnim proizvodnim okruženjima, omogućujući manjem broju stručnjaka da uspješno programiraju i koriste ove napredne tehnologije.

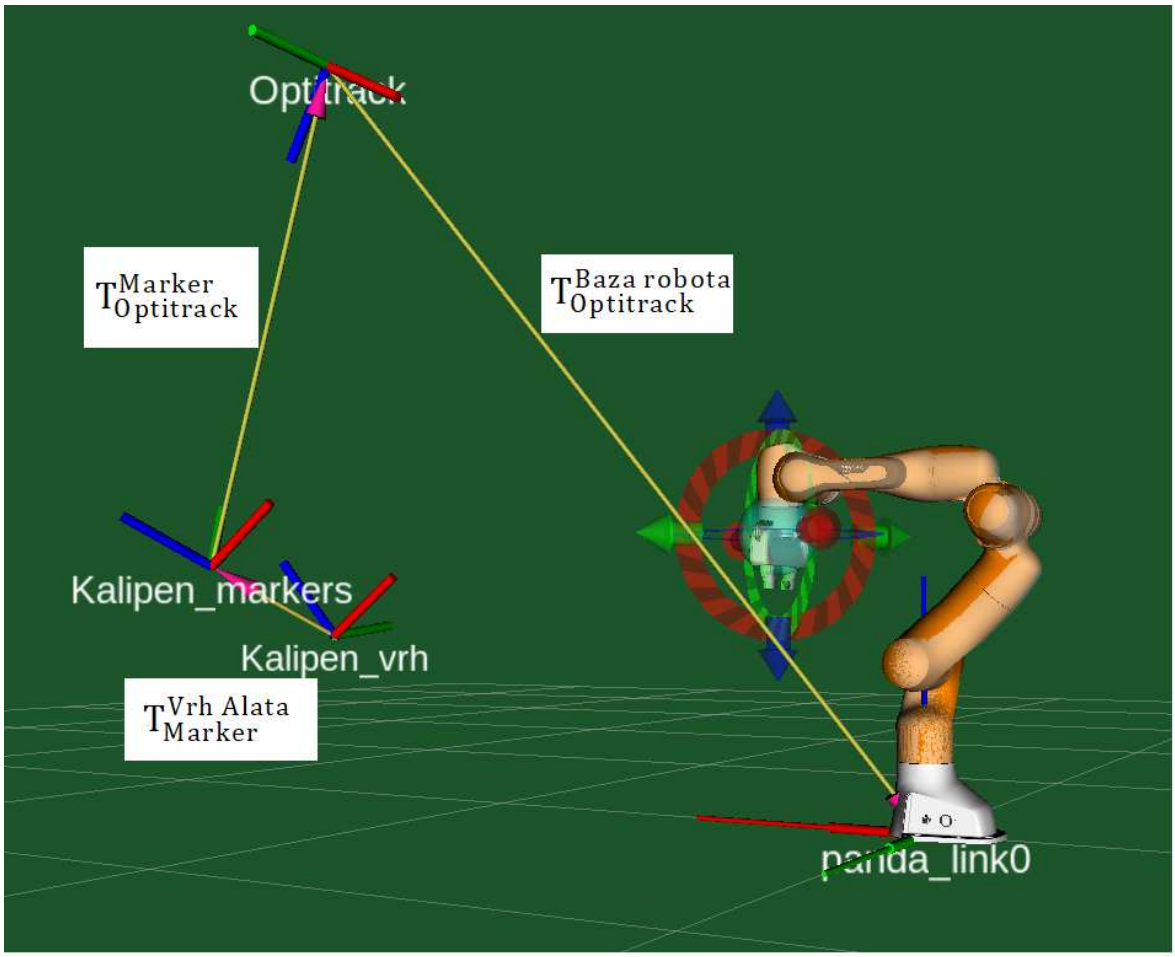
1. Čovjekom potpomognuto definiranje radnog prostora

U diplomskom radu se koriste i nadograđuju ideje iz dvaju znanstvenih članaka: „Human-friendly robot systems deployment using Virtual Pen" [1] i „Unsupervised optimization approach to in situ calibration of collaborative human-robot interaction tools" [2].
Osnova oba rada je korištenje Optitrack [3] sustava koji služi praćenju objekata u prostoru. Optički sustav Optitrack-a koristi infracrvene kamere visoke rezolucije kako bi detektirali reflektirajuće markere, čime se u stvarnom vremenu dobivaju 3D koordinate točaka u globalnom referentnom okviru. Za povezivanje Optitrack sustava koristi se program Motive u kojemu se označuju objekti s reflektirajućim markerima te omogućava slanje informacija o pozicijama objekata u Robot Operating System (ROS).
Virtualna olovka (Kalipen) predstavlja interaktivni alat koji operateru omogućava da intuitivno označi ključne točke u radnom prostoru – primjerice, međutočke, rubove prepreka, plohe ili objekte oblika kvadra. Kada operater usmjeri vrh olovke na određenu točku, optički sustav bilježi poziciju virtualne olovke.
Da bi se Virtualna olovka koristila zajedno s robotom, potrebno je napraviti niz kalibracija za određivanje međusobnih odnosa koordinatnih sustava svih sudionika.



Slika 1. Primjer Virtualne olovke

Na slici 1 je skica jedne Virtualne olovke gdje su narančastom bojom prikazani reflektivni markeri koje Optitrack sustav prepoznaje. Informacija koju primamo iz sustava je poza markera u globalnom koordinatnom sustavu Optitrack-a.



Slika 2 Koordinatni sustavi postava

Na slici 2 su prikazani svi koordinatni sustavi koji se koriste te koje je potrebno odrediti da bi se robot, Kalipen i Optitrack međusobno povezali.

Poze i homogene transformacijske matrice sadrže jednake informacije o odnosu translacija i orijentacija bilo koja dva koordinatna sustava te se oba pojma primjenjiva ovisno o kontekstu. Transformacijske matrice se jednostavno uzastopno množe kako bi se dobio odnos bilo koja dva objekta ako su sve veze poznate. Transformacijska matrica T se sastoji od rotacijske matrice R i vektora pozicije p (x, y, z): T = [R, p].

Kada u programu Motive označimo 4 reflektivna markera i kreiramo virtualni objekt „Kalipen“ te objavimo tu informaciju u ROS sustav, očitavamo $T^{Marker}_{Optitrack}$ koja definira 3D odnos markera u koordinatnom sustavu Optitracka.

Transformaciju koju treba kalibrirati je $T^{Vrh\ Kalipena}_{Marker}$ koja ovisi o fizičkim dimenzijama alata prikazanog na slici 1.

Druga transformacija koja se kalibrira je $T_{Optitrack}^{Baza\ Robota}$ kako bi se točke definirane u koordinatnom sustavu Optitracka preračunavale u koordinatni sustav robota kojim upravljamo slanjem pozicija.

1.1. Kalibracija Kalipen olovke – određivanje $T_{Marker}^{Vrh\ Alata}$

Da bi se Kalipenom mogao označavati prostor, potrebno je odrediti transformaciju između Markera i vrha alata $T_{Marker}^{Vrh\ Kalipena}$. Metoda kalibracije je definirana u radu [2] te se u sklopu ovog rada metoda kalibracije implementirala u programskom jeziku Python i ROS sustavu za jednostavnu integraciju između dijelova sustava.

Ideja kalibracije je da se vrh kalipena postavlja u rupičastu točku kako bi vrh bio stabilan i u konstantnoj poziciji. Kalipen olovka se kružno pomiče te se niz transformacija $T_{Optitrack}^{Marker}$ sprema u listu $[T_1, T_2, \ldots, T_N]$.

Ovom kalibracijom se određuje samo translacijska komponenta transformacijske matrice, odnosno vektor p. $T_{Marker}^{Vrh\ Kalipena} = [R_{Marker}^{Vrh\ Kalipena}, p_{Marker}^{Vrh\ Kalipena}]$

Kalibracija se provodi optimizacijom tako da se traži vektor $p_{Marker}^{Vrh\ Kalipena}$, dok rotacijski dio se postavlja na jediničnu matricu. Za takvu transformacijsku matricu će vrijediti da uvijek pokazuje u istu točku jer vrh kalipena nismo pomicali:

$$T_1 \times T_{Marker}^{Vrh\ Kalipena} = p_1 \quad (1)$$

$$T_2 \times T_{Marker}^{Vrh\ Kalipena} = p_1 \quad (2)$$

...

$$T_N \times T_{Marker}^{Vrh\ Kalipena} = p_1 \quad (3)$$

Sustav je implementiran tako da na pritisak gumba na kontroleru se počinju spremati transformacije i uzima se fiksni broj njih, nakon sakupljanja sustav sam započinje kalibraciju te vraća rezultat u obliku $p_{Marker}^{Vrh\ Kalipena}$.

Kada je poznata translacija, može se odrediti i rotacijski dio. U radu [2] je to napravljeno posebnim alatom, a u ovom diplomskom radu je razvijena jednostavnija ali manje precizna metoda za određivanje rotacije.

Kalipen se postavlja u uspravnu poziciju, tako da je dužina alata poravnata sa $\hat{z}$ osi svijeta, odnosno $\hat{z}$ osi baze robotskog manipulatora. Sprema se trenutna poza objavljena od Optitrack sustava čiji je rotacijski dio $R_{Optitrack_{uspravan}}^{Marker}$.

Pošto je namješteno da je Kalipen poravnat s $\hat{z}$ osi baze robotskog manipulatora, traži se $R_{Marker}^{Vrh\ Alata}$ koji pomnožen s $R_{Optitrack_{uspravan}}^{Marker}$ daje rotacijsku matricu koja opisuje poravnatost Kalipena s osi manipulatora. Kalipen gleda „prema dolje“ a baza ima z os „prema gore“ te njihov odnos redefiniramo rotacijskom matricom:

$$R_{Uspravno} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{4}$$

Onda vrijedi da:

$$R_{Marker}^{Vrh\ Kalipena} \times \mathrm{R}_{\mathrm{Optitrack}_{\mathrm{uspravan}}}^{\mathrm{Marker}} = R_{Uspravno} \tag{5}$$

Matričnim računom dobivamo izraz za traženu rotacijsku matricu:

$$R_{Marker}^{Vrh\ Kalipena} = \left(\mathrm{R}_{\mathrm{Optitrack}_{\mathrm{uspravan}}}^{\mathrm{Marker}}\right)^{-1} \times R_{Uspravno} \tag{6}$$

Spajanjem podataka $p_{Marker}^{Vrh\ Kalipena}$ i $R_{Marker}^{Vrh\ Kalipena}$ sada je poznata $T_{Marker}^{Vrh\ Kalipena}$.

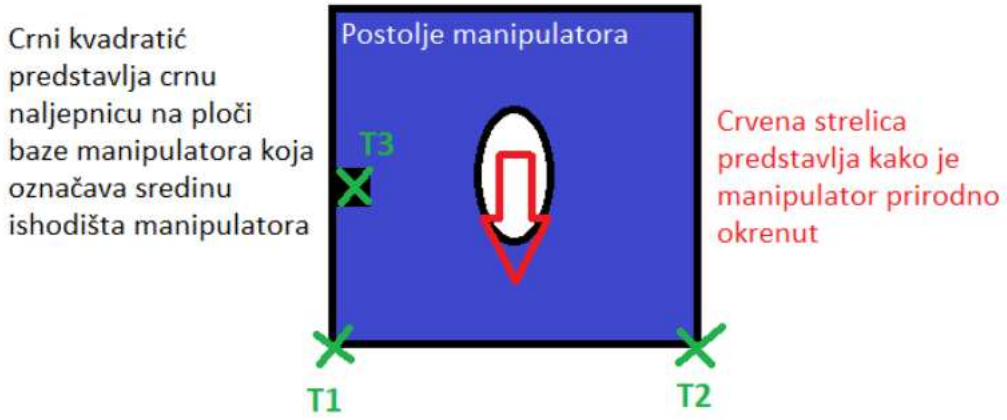
Množenjem transformacijskih matrica se određuje pozicija i orijentacija vrha alata u koordinatnom sustavu Optitrack-a:

$$T_{Optitrack}^{Vrh\ Kalipena} = T_{Optitrack}^{Marker} \times T_{Marker}^{Vrh\ Kalipena} \tag{7}$$

1.2. Kalibracija odnosa robotskog manipulatora u sustavu Optitrack-a - $T_{Optitrack}^{Baza\ Robota}$

Poznavajući sve prijašnje transformacije, sada je moguće označavati prostor u svijetu optitrack-a. Potrebno je označiti gdje se nalazi baza robota kako bi se točke vrha alata mogle preračunati u svijet robota.

Za to je razvijena programska podrška u kojoj se s 3 točke označava baza manipulatora Franka te se računa transformacijska matrica $T_{Optitrack}^{Baza\ Robota}$.



Slika 3 Prikaz baze manipulatora te redoslijed točaka koje se označavaju

Ova metoda je specifična za Franka manipulator koji se nalazi u FER-ovom LARICS laboratoriju jer se manipulator nalazi na postolju koje je izrađeno po narudžbi. Metoda se razlikuje za neki drugi manipulator.

Vrhom kalipena se prikupljaju 3 točke T_1, T_2, T_3 i na temelju njihovih pozicijskih komponenti se računa T.

$$\text{x\_axis} = t_2 - t_1, \quad \text{y\_axis} = t_3 - t_1 \tag{8}$$

$$\text{z\_axis} = \text{cross}(x\_axis, y\_axis) \tag{9}$$

$$R = [\hat{x}, \hat{y}, \hat{z}] \tag{10}$$

A vektor translacije se određuje kao:

$$t = t_1 + x\_axis + \frac{y\_axis}{2} \tag{11}$$

Spajanjem izračunatih R i t, dobivamo $T_{Optitrack}^{Baza\ Robota}$.

1.3. Pretvaranje poze Kalipena u svijet robota

Na kraju se pomoću kombinacije svih kalibriranih i poznatih poza određuje poza vrha kalipena u svijetu robota:

$$T_{Baza\ Robota}^{Vrh\ Kalipena} = \left(T_{Optitrack}^{Baza\ Robota}\right)^{-1} \times T_{Optitrack}^{Vrh\ Kalipena} \tag{12}$$

Informacija o vrhu alata se objavljuje u ROS sustav te ju drugim podsustavom možemo čitati i koristiti za definiranje svijeta u koordinatama robotskog manipulatora.

2. Generiranje trajektorija za zavarivanje

Klasa TrayectorySegment predstavlja segment putanje koji se može definirati kao linearni ili kružni, ovisno o parametru circ_or_lin. U konstruktoru klase, primaju se sljedeći parametri:

- **circ_or_lin**: Oznaka tipa segmenta – "L" za linearni ili "C" za kružni segment.
- **start_time**: Početno vrijeme segmenta.
- **start_pose, end_pose, interrim_pose**: Tri prostorne točke koje definiraju početne i krajnje pozicije segmenta te točku na kružnom luku kod kružnog segmenta.
- **v_max i a_max**: Maksimalna brzina i maksimalna akceleracija izražena za najkraću udaljenost između početne i krajnje točke. Brzine su zadane u kartezijskim koordinatama. Parametri se primjenjuju u trapeznom profilu brzina.
- **d_t**: Vremenski interval (delta t) za generiranje putanje.
- **amplitude i freq**: Amplituda i frekvencija sinusnih oscilacija od savršenog pravca ili kružnog luka.

Sinusne oscilacije su definirane u ravnini koja je okomito na alat kod linijskih putanja i ravnina u kojem je definirana kružnica kod kružnih putanja.

2.1. Generiranje linearne putanje

Funkcija generate_linear_path_from_y_trap() generira putanju koja se kreće ravnom linijom između zadanih točaka (start_point i end_point) koristeći trapezni profil brzine. Na putanju se dodaje sinusni pomak u ravnini okomitoj na početnu orijentaciju (start_orientation), što rezultira oscilatornim odstupanjem s obje strane glavne putanje. Rezultat je lista točaka: vrijeme i globalne koordinate (t, x, y, z) te dodatne informacije o parametrima putanje.

2.1.1. Definicija referentnog okvira

Osnovni vektor pomaka se definira između početne i krajnje točke putanje. Definira ga se kao lokalnu $\vec{y}$ os.

Izračun:

$$\vec{y} = \text{end\_point} - \text{start\_point} \tag{13}$$

Duljina putanje:

$$L = |\vec{y}| \tag{14}$$

Jedinični vektor duž putanje:

$$\hat{y} = \frac{\vec{y}}{L} \tag{15}$$

Lokalni $\vec{x}$ smjer:

Preuzima se prvi stupac rotacijske matrice R dobivene iz kvaterniona (start_orientation) koja je orijentacija prve točke na putanji: $x = R[:,0]$

Kako bi se osiguralo da je $\vec{x}$ okomit na $\hat{y}$, provodi se projekcija

$$x_{\text{proj}} = x - (x \cdot \hat{y})\hat{y} \tag{16}$$

Normalizacija u jedinični vektor:

$$\hat{x} = \frac{x_{\text{proj}}}{|x_{\text{proj}}|} \tag{17}$$

Lokalni $\vec{z}$ smjer:

Računa se vektorskim umnoškom jediničnih vektora $\hat{x}$ i $\hat{y}$ kako bi se dobio desno orijentirani koordinatni sustav

$$\hat{z} = \hat{y} \times \hat{x} \tag{18}$$

2.1.2. Trapezni profil brzine

Putanja se odvija u tri faze: ubrzanje, konstantna brzina i usporavanje.

a) Faza ubrzanja

Vrijeme ubrzanja:

$$t_{\text{acc}} = \frac{v_{\text{max}}}{a_{\text{max}}} \tag{19}$$

Prijeđena udaljenost tijekom ubrzanja:

$$d_{\text{acc}} = 0.5 \cdot a_{\text{max}} \cdot t_{\text{acc}}^2 \tag{20}$$

b) Provjera dosezanja maksimalne brzine

Ako je $2 \cdot d_{\text{acc}} \geq L$, putanja je dovoljno kratka da se maksimalna brzina ne postiže:

$$v_{\text{peak}} = \sqrt{L \cdot a_{\text{max}}} \quad \text{i} \quad t_{\text{acc}} = \frac{v_{\text{peak}}}{a_{\text{max}}}, \quad t_{\text{cruise}} = 0 \tag{21}$$

Inače se koristi:

$$v_{\text{peak}} = v_{\text{max}} \tag{22}$$

Preostala udaljenost za fazu konstantne brzine:

$$d_{\text{cruise}} = L - 2 \cdot d_{\text{acc}} \tag{23}$$

Vrijeme konstantne brzine:

$$t_{\text{cruise}} = \frac{d_{\text{cruise}}}{v_{\text{peak}}} \tag{24}$$

c) Ukupno vrijeme putovanja

Kombinacijom svih faza dobiva se

$$: t_{\text{total}} = 2 \cdot t_{\text{acc}} + t_{\text{cruise}} \tag{25}$$

d) Definicija funkcije puta $s(t)$

Faza ubrzanja:

$$s_{\text{acc}}(t) = 0.5 \cdot a_{\text{max}} \cdot t^2 \tag{26}$$

Faza konstantne brzine:

$$s_{\text{cruise}}(t) = d_{\text{acc}} + v_{\text{peak}} \cdot (t - t_{\text{acc}}) \tag{27}$$

Faza usporavanja:

Neka je $t_{\text{decel}} = t - (t_{\text{acc}} + t_{\text{cruise}})$, tada:

$$s_{\text{dec}}(t) = d_{\text{acc}} + v_{\text{peak}} \cdot t_{\text{cruise}} + \left(v_{\text{peak}} \cdot t_{\text{decel}} - 0.5 \cdot a_{\text{max}} \cdot t_{\text{decel}}^2\right) \tag{28}$$

2.1.3. Sinusni pomak

Kako bi se računala sinusna oscilacija koja je simetrična (vrijednost 0 na početku i kraju putanje) potrebno je

Centriranje sinusnog vala:

Koristi se argument:

$$\text{argument} = s - \frac{L}{2} \tag{29}$$

što osigurava da je $\sin(-\pi n) = 0$ i $\sin(\pi n) = 0$ ako je broj ciklusa n cijeli broj.

Prilagodba frekvencije:

Da bi se osiguralo da se tijekom duljine L završi cijeli broj ciklusa:

$$L \cdot \text{freq}_{\text{adj}} = n \quad \Rightarrow \quad \text{freq}_{\text{adj}} = \frac{n}{L} \tag{30}$$

gdje se n računa kao $n = \text{round}(L \cdot \text{freq})$

U suprotnom se može dogoditi da sinusna funkcija završava u bilo kojoj svojoj točki, npr. amplitudi. U tom slučaju kada se segmenti dodaju jedan na drugi, postoje skokovi između segmenta jer sinusna funkcija nije u ravnotežnom položaju u konačnoj točki segmenta.

Izračun sinusnog odstupanja:

Jednadžbom sinusne funkcije se računa odstupanje u lokalnom $\hat{x}$ smjeru:

$$\text{offset} = \text{amplitude} \cdot \sin\left(2\pi \cdot \text{freq}_{\text{adj}} * \left(s - \frac{L}{2}\right)\right) \tag{31}$$

gdje je s trenutna pozicija u putanji po lokalnoj $\hat{y}$ osi.

2.1.4. Izračun globalne pozicije

Vremena putanje se definiraju od $t = start\_time$ do $t = start\_time + t\_total$. Parametar $start\_time$ se koristi kada se segmenti putanje slažu jedan na drugi te je $start\_time$ segmenta jednak ukupnom vremenu gibanja prošloga segmenta. Za prvi segment je $start\_time$ uvijek jednak nuli. Između početnog i konačnog vremena se uzimaju sva vremena zadana parametrom dt.

Za parametre $start\_time = 0$ i $dt = 0.1$, lista vremenskih koraka je:

$$t = [0,0.1,0.2, \dots, t_total - 0.1, t_total] \tag{32}$$

Za svaki vremenski korak t se:

1. Odredi $s(t)$ ovisno o fazi (ubrzanje, konstanta brzina, usporavanje).
2. Izračuna se sinusni pomak.
3. Konačna globalna pozicija:

$$\text{pos\_global} = \text{start\_point} + s \cdot \hat{y} + \text{offset} \cdot \hat{x} \tag{33}$$

Time se postiže da se kretanje odvija duž glavne linije s dodatnim oscilacijskim odstupanjem u smjeru $\hat{x}$.

2.1.5. Povratne vrijednosti funkcije

path: Lista točaka u obliku (t, x, y, z) koje predstavljaju globalne koordinate u svakom vremenskom koraku.

t_total: Ukupno vrijeme trajanja putanje.

L: Duljina putanje.

start_point: Početna točka

y_dir ($\hat{y}$): Jedinični vektor duž glavne putanje.

x_dir ($\hat{x}$): Lokalni smjer oscilacije.

z_dir ($\hat{z}$): Lokalni smjer koji je okomit na oba prethodna te svi zajedno čine desno orijentirani koordinatni sustav.

Funkcija **generate_linear_path_from_y_trap()** primjenjuje trapezni profil brzine duž glavne osi pomaka između početne i krajnje točke s dodatnim sinusnim pomakom, osiguravajući:

- **Glatko ubrzanje i usporavanje:** Prema postavljenim ograničenjima brzine v_{max} i akceleracije a_{max} .
- **Simetričnu oscilaciju:** Sinusni pomak je centriran tako da je vrijednost odstupanje 0 na početku i kraju, čime se postižu željene oscilacije bez naglih promjena.

2.2. Generiranje kružne putanje

Funkcija generate_path_from_circluar_arc() generira putanju koja se kreće putanjom kružnog luka između zadanih točaka (start_point, interrim_point i end_point) koristeći trapezni profil brzine u smjeru pomaka po kružnom luku. Na putanju se dodaje sinusni pomak u ravnini definiranoj s 3 točke kružnice te se pomak dodaje u radijalnom smjeru (smjer od središta kružnice prema točkama na kružnici). Rezultat je lista točaka: vrijeme i globalne koordinate (t, x, y, z) te dodatne informacije o parametrima putanje.

2.2.1. Definicija ravnine i određivanje parametara kružnice

a) Određivanje ravnine

Izračun normala ravnine:

Tri dane točke p_{start}, $p_{interrim}$ i p_{end} definiraju ravninu čija se normala izračuna:

$$\boldsymbol{n} = (p_{\text{interrim}} - p_{\text{start}}) \times (p_{\text{end}} - p_{\text{start}}) \tag{34}$$

Normalizacija vektora:

$$\hat{n} = \frac{\boldsymbol{n}}{|\boldsymbol{n}|} \tag{35}$$

Ako je $|\boldsymbol{n}|$ vrlo malen, točke su kolinearne te program izbaci pogrešku s odgovarajućom porukom.

b) Određivanje središta kružnice

Koristeći točke u ravnini, definiramo vektore:

$$b = p_{\text{interrim}} - p_{\text{start}}, c = p_{\text{end}} - p_{\text{start}} \tag{36}$$

Primjena formule za središte kružnice:

$$p_{\text{center}} = p_{\text{start}} + \frac{\big(c \times (b \times c)\big)|b|^2 + \big((b \times c) \times b\big)|c|^2}{2\,|b \times c|^2} \tag{37}$$

c) Izračun radijusa kružnice

Po formuli:

$$R = |p_{\text{start}} - p_{\text{center}}| \tag{38}$$

2.2.2. Lokalni koordinatni sustav u ravnini kružnice

Lokalni $\hat{x}$ smjer:

Definira se u smjeru između početne točke i središta kružnice te predstavlja smjer s kutom jednakim nula.

$$\hat{x} = \frac{p_{\text{start}} - p_{\text{center}}}{R} \tag{39}$$

Lokalni $\hat{y}$ smjer:

Računa se tako da zajedno s normalom i lokalnim $\hat{x}$ smjerom čini desno orijentirani koordinatni sustav. Ujedno je taj smjer i tangenta na kružnicu u početnoj točci kružnog luka.

$$\hat{y} = \hat{n} \times \hat{x} \tag{40}$$

Računanje polarnih koordinata

Funkcija angle_from_center(P) računa kut ϕ od točke P u lokalnom koordinatnom sustavu:

$$\phi = atan2(\langle P - p_{\text{center}}, \hat{y}\rangle, \langle P - p_{\text{center}}, \hat{x}\rangle) \tag{41}$$

Ako je kut negativan, dodaje se 2π kako bi se dobio kut u rasponu $[0,2\pi)$

Računaju se kutovi zadanih točaka:

$$\phi_{\text{start}} = 0, \quad \phi_{\text{inter}} = \text{angle\_from\_center}(p_{\text{interrim}}), \quad \phi_{\text{end}} = \text{angle\_from\_center}(p_{\text{end}}) \tag{42}$$

Ako je $\phi_{end} < \phi_{interrim}$, dodaje se 2π kako bi luk sigurno prošao kroz $\phi_{interrim}$.

Ukupni kutni raspon:

$$\Delta\phi = \phi_{\text{end}} - \phi_{\text{start}} \tag{43}$$

Ukupna duljina kružnog luka:

$$S = R \cdot \Delta\phi \tag{44}$$

2.2.3. Kinematika putanje – trapezni profil brzine

Trapezni profil brzine se održava u smjeru kružnog luka.

a) Pretvorba linearnih ograničenja u kutne

Maksimalna kutna brzina:

$$\omega_{\text{max}} = \frac{v_{\text{max}}}{R} \tag{45}$$

Maksimalna kutna akceleracija:

$$\alpha_{\max} = \frac{a_{\max}}{R} \tag{46}$$

b) Parametri trapeznog profila brzina

Vrijeme ubrzanja:

$$t_{\text{acc}} = \frac{\omega_{\max}}{\alpha_{\max}} \tag{47}$$

Kutni pomak tijekom ubrzanja:

$$\phi_{\text{acc}} = 0.5 \cdot \alpha_{\max} \cdot t_{\text{acc}}^2 \tag{48}$$

Ako je $2\ \phi_{\text{acc}} > \Delta\phi$, primjenjuje se trokutasti profil (maksimalna brzina se nikada ne dostiže):

$$\omega_{\text{peak}} = \sqrt{\Delta\phi \cdot \alpha_{\max}}, \quad t_{\text{acc}} = \frac{\omega_{\text{peak}}}{\alpha_{\max}}, \quad t_{\text{cruise}} = 0 \tag{49}$$

Inače:

$$t_{\text{cruise}} = \frac{\Delta\phi - 2 \cdot d_{\text{acc}}}{\omega_{\max}} \tag{50}$$

Ukupno vrijeme putovanja:

$$t_{\text{total}} = 2\, t_{\text{acc}} + t_{\text{cruise}} \tag{51}$$

c) Definicija funkcije kutnog pomaka $\phi(t)$

Definira se ovisno o fazi u kojoj je.

Faza ubrzanja:

$$\phi_{\text{acc}}(t) = 0.5 \cdot \alpha_{\max} \cdot t^2 \tag{52}$$

Faza konstantne brzine:

$$\phi_{\text{cruise}}(t) = d_{\text{acc}} + \omega_{\max} \cdot (t - t_{\text{acc}}) \tag{53}$$

Faza usporavanja za $t_{\text{decel}} = t - t_{\text{acc}} - t_{\text{cruise}}$:

$$\phi_{\text{dec}}(t) = d_{\text{acc}} + \omega_{\max} \cdot t_{\text{cruise}} + (\omega_{\max} \cdot t_{\text{decel}} - 0.5 \cdot \alpha_{\max} \cdot t_{\text{decel}}^2) \tag{54}$$

2.2.4. Sinusni pomak u radijalnom smjeru

Generira se putanja koja sinusno oscilira oko kružnog luka, pomak se primjenjuje u radijalnom smjeru putanje.

Potrebno je centrirati sinusnu funkciju argumentom kako bi funkcija počela i završila s istim pomakom:

$$argument = s - \frac{S}{2} \quad \text{gdje je } s = R\phi\text{, pređeni luk} \tag{55}$$

Da se osigura da funkcija počinje i završava s pomakom jednakim nula, potrebno je prilagoditi frekvenciju:

$$\text{freq}_{\text{adj}} = \frac{n_{\text{cycles}}}{S}, \quad \text{gdje je } n_{\text{cycles}} = \text{round}(\text{freq} \cdot S) \tag{56}$$

Izračun sinusnog pomaka:

$$\text{offset} = \text{amplitude} \cdot \sin\left(2\pi \cdot \text{freq}_{\text{adj}} \cdot \left(s - \frac{S}{2}\right)\right) \tag{57}$$

2.2.5. Izračun globalne pozicije na kružnom putu

Vremenski koraci se generiraju jednako kao kod linearne putanje.

Za svaki vremenski korak t se određuje:

1. **Kutna pozicija** $\phi(t)$:

   Određuje se ovisno o fazi (ubrzanje, konstanta brzina, usporavanje).

2. **Idealna točka na kružnici:**

$$p_{ideal} = p_{center} + R(cos(\phi) \cdot \hat{x}\, sin(\phi) \cdot \hat{y}) \tag{58}$$

3. **Prijeđeni put:**

$$s = R \cdot \phi \tag{59}$$

4. **Radijalni smjer:**

   Jedinstveni smjer u ravnini kružnice okomit na tangentu:

$$\hat{r} = \frac{p_{ideal} - p_{center}}{R} \tag{60}$$

5. **Konačna globalna pozicija:**

   Uključujući sinusni pomak:

$$p = p_{ideal} + \text{offset} \cdot \hat{r} \tag{61}$$

   Ova se pozicija zajedno s vremenom dodaje u izlaznu listu.

2.2.6. Povratne vrijednosti funkcije

path: Lista točaka u obliku (t, x, y, z) koje predstavljaju globalne koordinate u svakom vremenskom koraku.

t_total: Ukupno vrijeme trajanja kretanja (uz start_time pomak).

S: Ukupna duljina luka.

center: Središte kružnice.

local_x: Jedinični vektor u ravnini, smjer od p_{center} prema p_{start}.

local_y: Jedinični vektor u ravnini kružnice, okomit na $\hat{x}$.

normal: Jedinični vektor normala na ravninu kružnice.

Funkcija **generate_path_from_circular_arc** generira putanju duž kružnog luka definiranog s tri nekolinearne točke, kombinirajući:
Geometrijsku analizu:
Određivanje ravnine, središta i radijusa kružnice te definiranje lokalnog koordinatnog sustava.
Kinematiku kretanja:
Implementaciju trapeznog profila brzine (s pretvorbom linearnih ograničenja u kutne) za glatko kretanje duž kružnog luka.
Sinusni oscilatorni pomak:
Primjenu oscilatornog pomaka u radijalnom smjeru s prilagođenom frekvencijom kako bi se osiguralo da je pomak simetričan (nula na početku i kraju luka).

2.3. Generiranje trajektorije robota iz podataka generirane putanje

Ovo poglavlje prolazi kroz sve korake generiranja i izvršavanja trajektorije gibanja na robotu. Od prikupljanja točaka do putanje, računa koordinata u svijetu robota, računanja brzina i akceleracija te provjera ispravnosti generiranog gibanja.

2.3.1. Jednostavno spremanje točaka putanje

Funkcija **click_callback()** omogućuje interakciju korisnika s sustavom putem kontrolera (joystick). Na temelju pritisnutih gumbi, funkcija dinamički prikuplja i kategorizira točke putanje koji definiraju trajektoriju zavarivanja:
Dva su moguća načina prikupljanja točki: ili se prikupljaju točke putem kalipen olovke te transformiraju u koordinatni sustav alata robota ili se direktno mogu prikupljati točke koje se pokažu alatom robota.
Gumb 0: Kada se pritisne, zadana točka se dodaje u listu međutočaka (engl. waypoints). Ako već postoje, nova međutočka se označava kao linearni segment ("L").
Gumb 1: Ovim se pokreće postupak za definiranje kružnog segmenta. Ako je lista međutočaka prazna, prvi položaj se spremi, dok se sljedeći, ovisno o vrijednosti indeksa circle_index, dodaje te se međutočka označava kao kružni segment ("C"). Indeks se koristi za razlikovanje točke na luku kružnice i završne točke kružnog segmenta.

Gumb 3: Pokreće se metoda **run()** koja inicira planiranje i izvršenje generirane trajektorije, a pritom se ispisuju dijagnostičke informacije o prikupljenim međutočkama. Ova funkcionalnost omogućuje korisniku intuitivno definiranje željene putanje, što je ključno za jednostavnost izvedbe gibanja zavarivanja.

2.3.2. Generiranje i Spajanje Segmenta Putanje

metoda createConnectedSegments():

Metoda obrađuje definirane međutočke te, ovisno o tipu segmenta (linearni ili kružni), kreira pojedinačne segmente putanje koristeći zadane parametre: maksimalnu brzinu v_{max}, maksimalnu akceleraciju a_{max}, vremenski korak dt, amplitudu i frekvenciju sinusnog pomaka. Ovi segmenti se generiraju primjenom funkcija **generate_linear_path_from_y_trap** i **generate_path_from_circular_arc**, čime se osigurava glatko i kontinuirano kretanje.

metoda createSinglepath():

Nakon generiranja pojedinačnih segmenata, ova metoda spaja sve segmente u jednu cjelovitu putanju. Rezultat je niz točaka (t, x, y, z) koje predstavljaju kontinuirani put kroz prostor, čime se osigurava konzistentnost cjelokupne trajektorije.

2.3.3. Izračun inverzne kinematike

Inverzna kinematika je proces u kojemu se računaju potrebni zakreti zglobova robota kako bi robot dosegao neku točku i orijentaciju u kartezijskim koordinatama.

Za svaku točku na potpunoj putanji, funkcija iterativno računa rješenje inverzne kinematike. Na temelju željenog položaja i orijentacije (dobivenih spajanjem pozicijskih i orijentacijskih podataka) izračunavaju se zglobne konfiguracije robota.

Ovaj postupak omogućava prevođenje prostornih koordinata u konkretne naredbe za zglobove, osiguravajući izvedivost trajektorije unutar robotskih ograničenja.

U sklopu ovog rade se koristi već gotovi paket for_franka_ros [4] u kojem se uspostavlja ROS servis za račun inverzne kinematike u obliku:

```
response = self.get_ik_client(wanted_pose, last_pose)
```

Ovaj ROS servis se poziva za svaku točku putanje te se dobiva matrica veličine $N \times n\_zglobova$ gdje je N ukupan broj točaka u putanji a n_zglobova je jednak 7 za robot Franku.

U svakom retku matrice je popis veličina zakreta svih zglobova robota da bi se dosegla točka zadana iz putanje. Uz to se definira lista svih vremenskih trenutaka koja je bila

definirana u putanji a sada se ti vremenski trenuci vežu uz zakrete zglobove. Matrica zglobova je definirana u varijabli joint_positions_list a vremena u varijabli time_stamps.

2.3.4. Računanje i filtriranje direktne kinematike

Direktna kinematika je proces računanja kartezijskih koordinata vrha alata pomoću zadanih zakreta zglobova robota. Ovdje se opet koristi već implementirana funkcija iz for_franka_ros [4] paketa koja za listu veličine n_zglobova vraća pozu alata (poziciju i orijentaciju) od zadanih zakreta zglobova.

Izračunom direktne kinematike za sve točke iz joint_position_list dobiva se putanja p_{fk} koja se uspoređuje s planiranom putanjom p_d dobivenom funkcijom createSinglePath().

Filtriranje rješenja inverzne kinematike:

U svakom koraku, izračunava se Euklidska udaljenost između željene pozicije i pozicije izračunate putem direktne kinematike:

$$\Delta = \sqrt{\left(x_d - x_{fk}\right)^2 + \left(y_d - y_{fk}\right)^2 + \left(z_d - z_{fk}\right)^2}.$$

Ako je euklidska udaljenost Δ veća od zadane tolerancije ε, točka se uklanja iz joint_position_list jer je došlo do greške kod računanja inverzne kinematike te točka ne odgovara željenoj.

Izračun brzine i akceleracije:

Brzina v_i i akceleracija a_i zglobova se računaju metodom konačnih razlika. Za dva uzastopna koraka, gdje su zakreti zglobova q_i i q_{i+1} zabilježeni u vremenima t_i i t_{i+1} , brzina se računa kao:

$$v_i = \frac{q_{i+1} - q_i}{t_{i+1} - t_i}.$$

Slično, akceleracija se računa kao:

$$a_i = \frac{v_{i+1} - v_i}{t_{i+1} - t_i}.$$

Brzine i akceleracije se računaju za pojedinačan zglob robotskog manipulatora. Izrazi za brzinu i akceleraciju zglobova su nam bitni kako bi se kreirala trajektorija zglobnim koordinatama koja se šalje robotskom kontroleru na izvršavanje. Također omogućava provjeru fizičkih ograničenja motora na zglobovima manipulatora.

2.3.5. Provjera kinematičkih ograničenja zglobova

Izračunati parametri (brzina i akceleracija) se uspoređuju s definiranim maksimalnim vrijednostima za svaki zglob.

Ako za bilo koji zglob dođe do prekoračenja postavljenih granica, sustav generira upozorenje.

Ova provjera je ključna za sigurnost i mehaničku stabilnost robota jer sprječava izvođenje naredbi koje bi mogle dovesti do oštećenja robota ili opreme.

2.3.6. Generiranje i objavljivanje trajektorije

Nakon filtriranja i provjere svih parametara trajektorije, na temelju preostalih podataka generira se ROS poruka tipa **JointTrajectory** koja sadrži:

- pozicije zglobova (zakrete motora),
- brzine zglobova,
- akceleracije zglobova,
- vremenske oznake

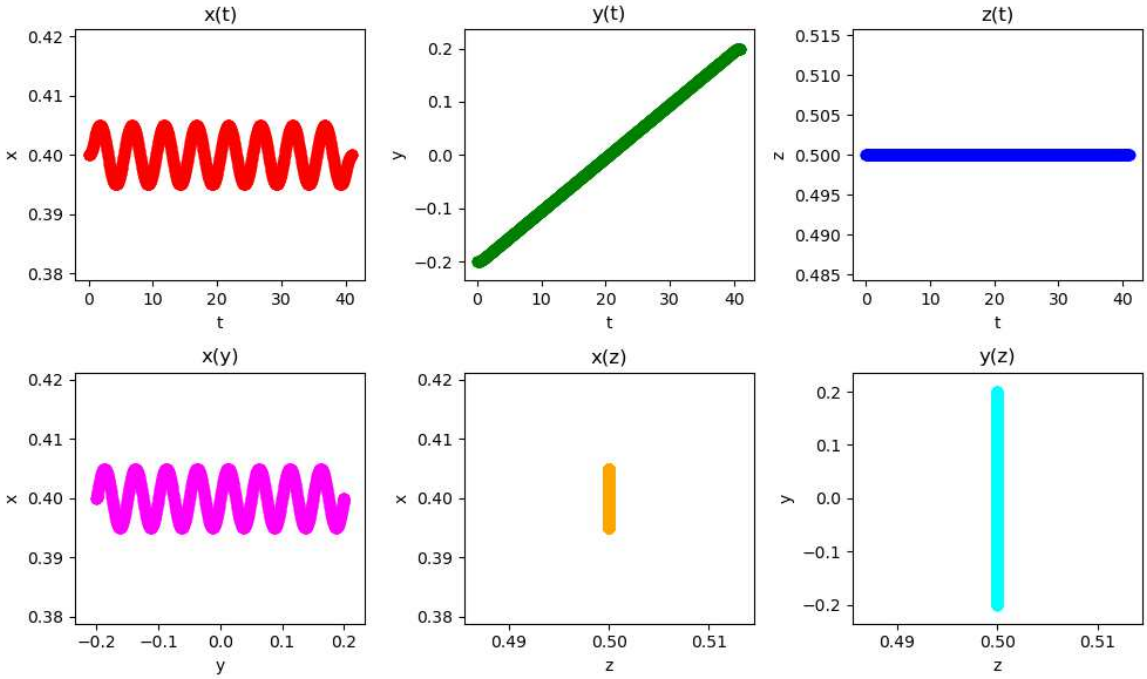
Prvo se robot postavlja u početnu poziciju, a zatim se cjelovita trajektorija objavljuje putem ROS sustava, što omogućava izvedbu zadatka u stvarnom vremenu.

3. Rezultati

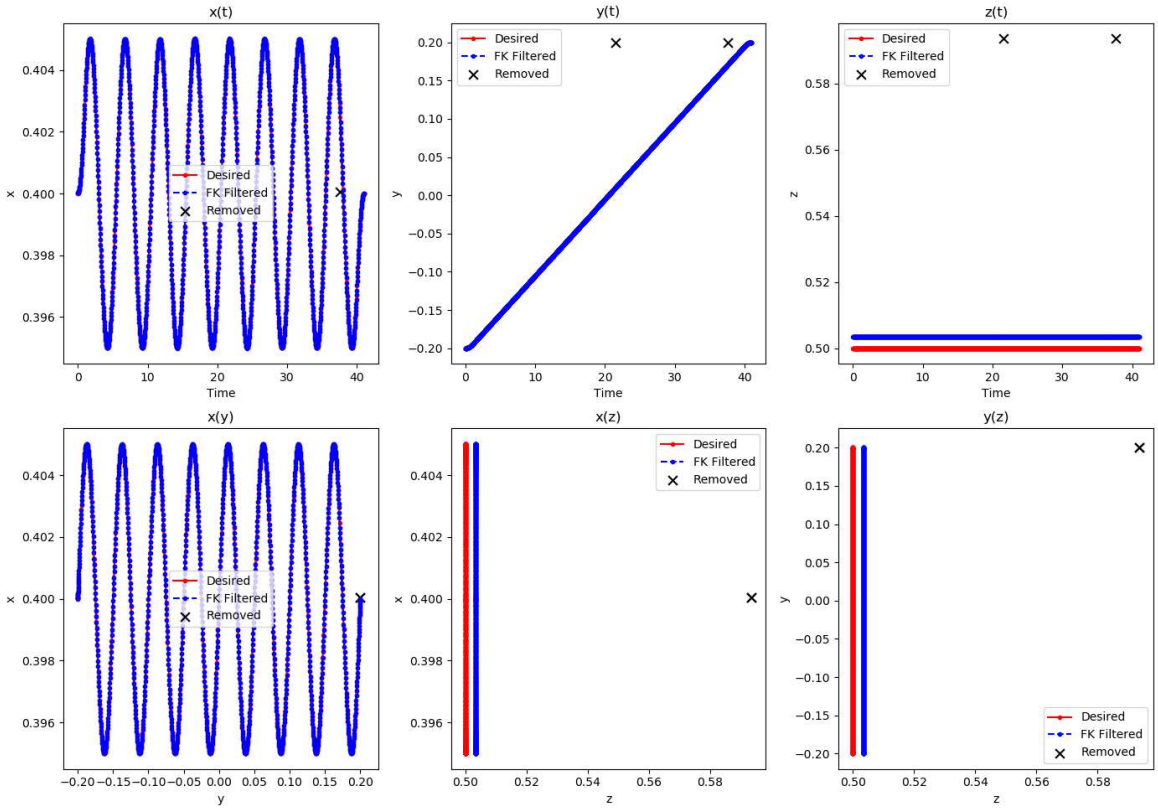
U ovom poglavlju su prikazani rezultati zadavanja i izvršavanja trajektorija zavarivanja. Prikazati će se 3 različite trajektorije s različitim parametrima: jedan segment linarne trajektorije, jedan segment kružne trajektorije te trajektorija koja je kombinirana od više linearnih i kružnih segmenata. Rezultati prve dvije trajektorije su dobiveni iz simulacijskog okruženja, a treća je izvršena na pravom robotskom manipulatoru.

1. Linearna trajektorija zadana točkama (0.4, -0.2, 0.5) i (0.4, 0.2, 0.5) te kvaternionom orijentacije (-1, 0, 0, 0) koja predstavlja orijentaciju alata okomito na x-y ravninu.
Za trapezni profil su i oscilacije su parametri:

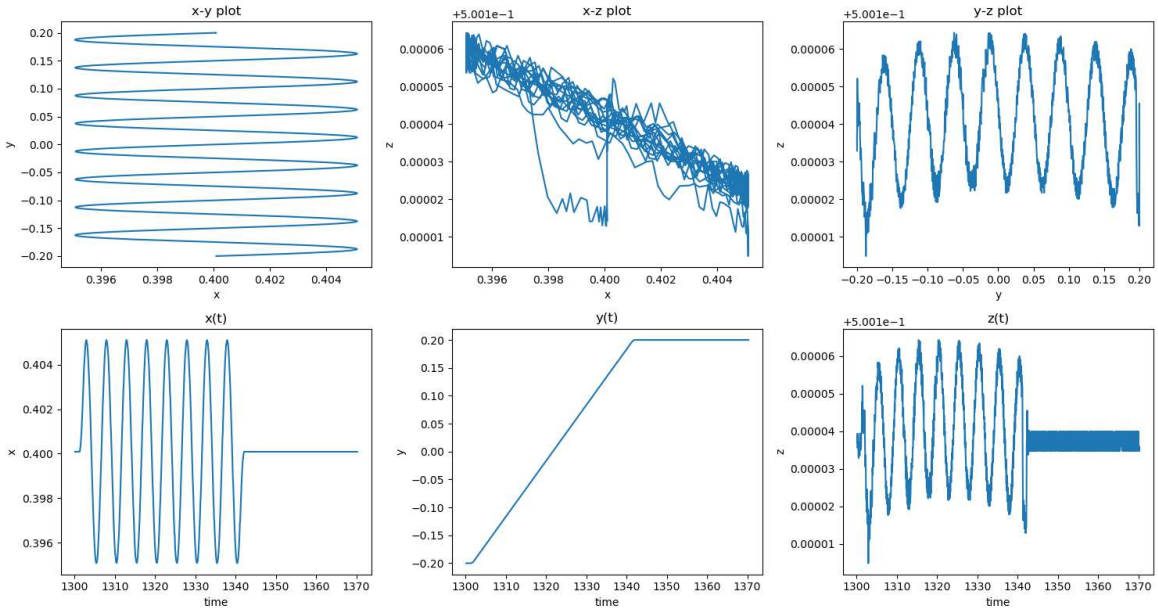
$$v_{\max} = 0.01, a_{\max} = 0.01, freq = 0.2, amplitude = 0.005$$



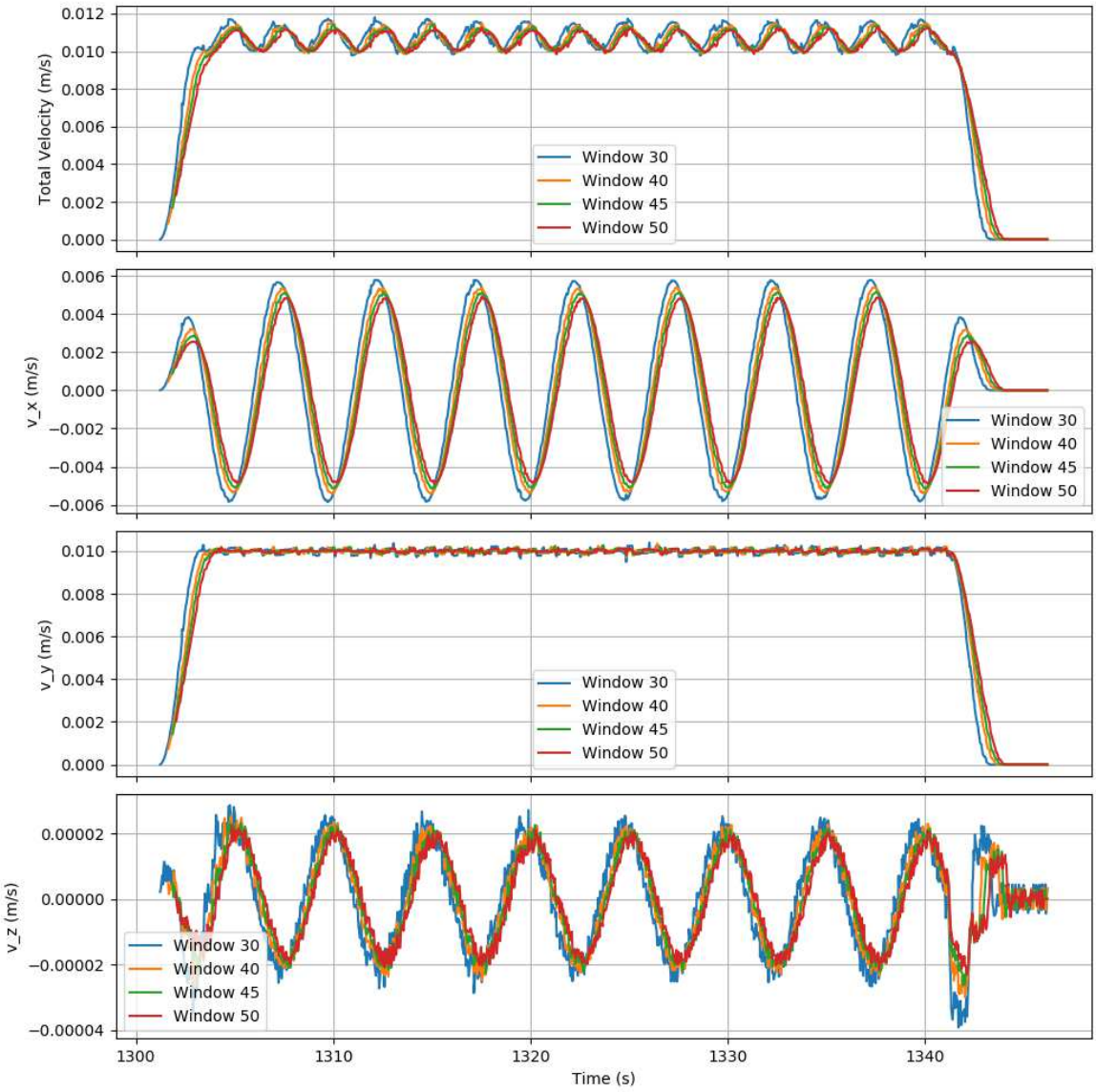
Slika 4 Prikaz isplanirane trajektorije



Slika 5 Prikaz filtrirane trajektorije



Slika 6 Prikaz izvršene trajektorije



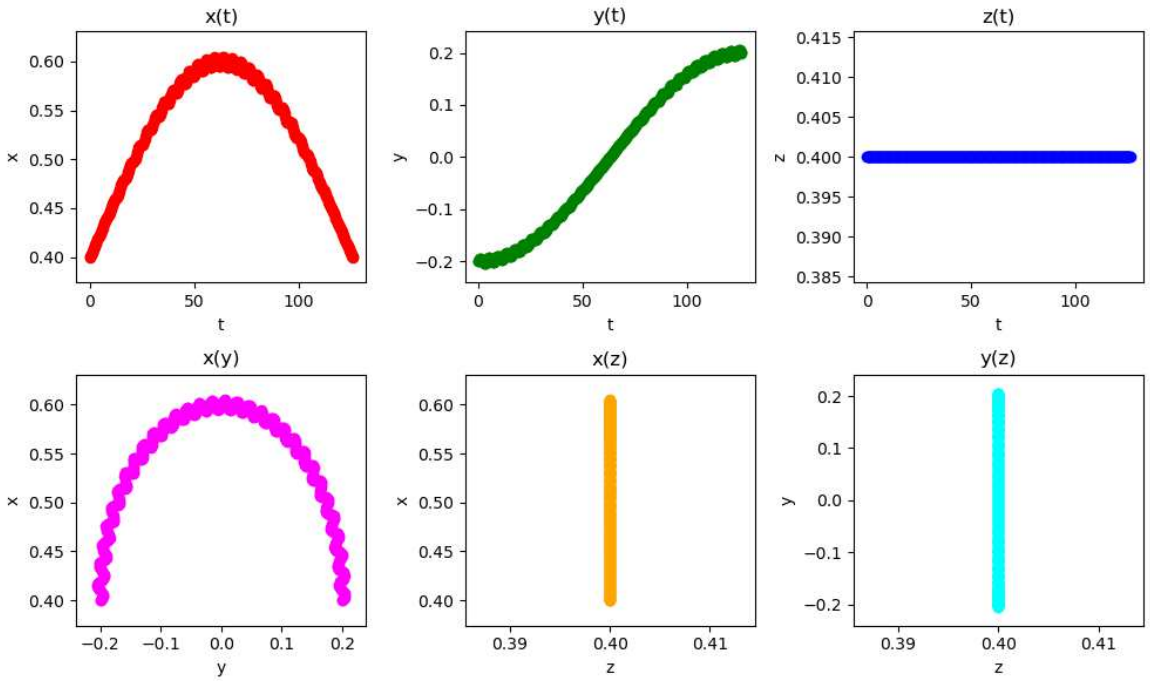
Slika 7 Prikaz brzina izvršene trajektorije

Na slikama 4, 5, 6, 7 su vidljivi rezultati planiranja i izvršavanja linearne trajektorije.

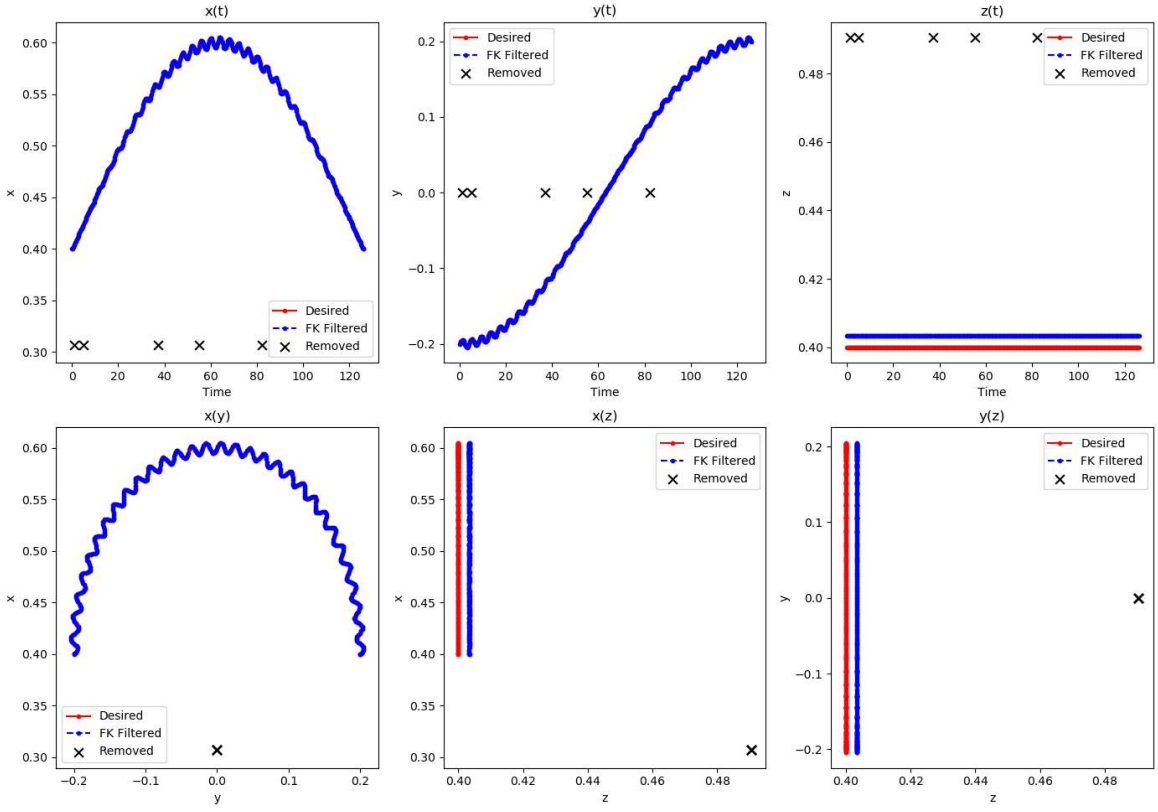
2. Kružna trajektorija zadana točkama (0.4, -0.2, 0.5), (0.6, 0, 0.5), (0.4, 0.2, 0.5) te istim kvaternionom orijentacije (-1, 0, 0, 0) koja predstavlja orijentaciju alata okomito na x-y ravninu.

Za trapezni profil su i oscilacije su parametri:

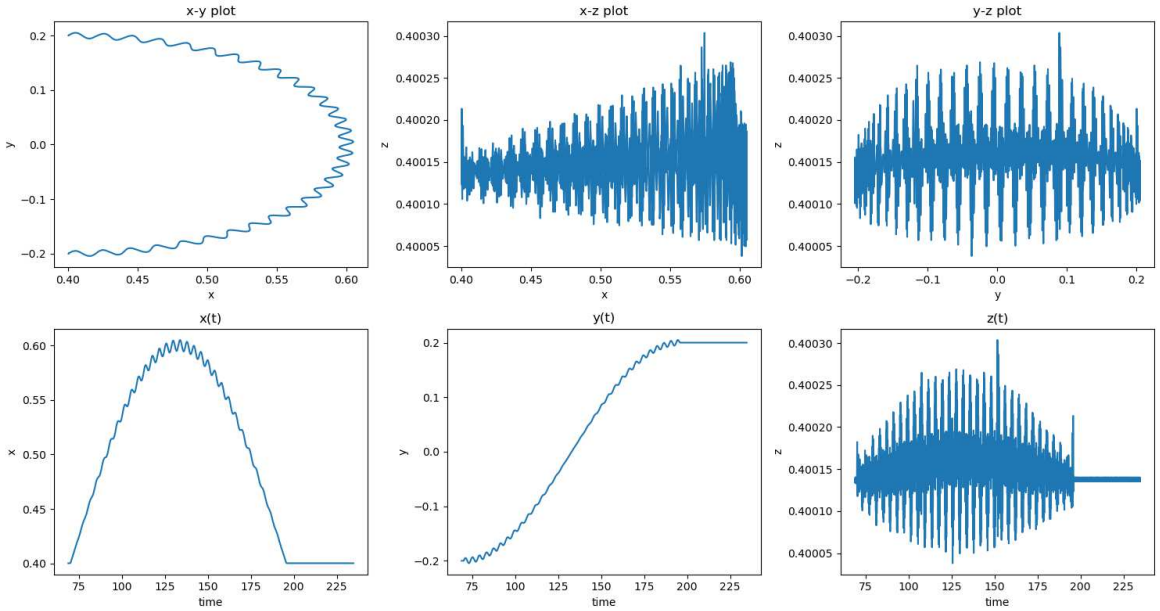
$$v_{\max} = 0.005,\ a_{\max} = 0.01, freq = 0.5, amplitude = 0.005$$



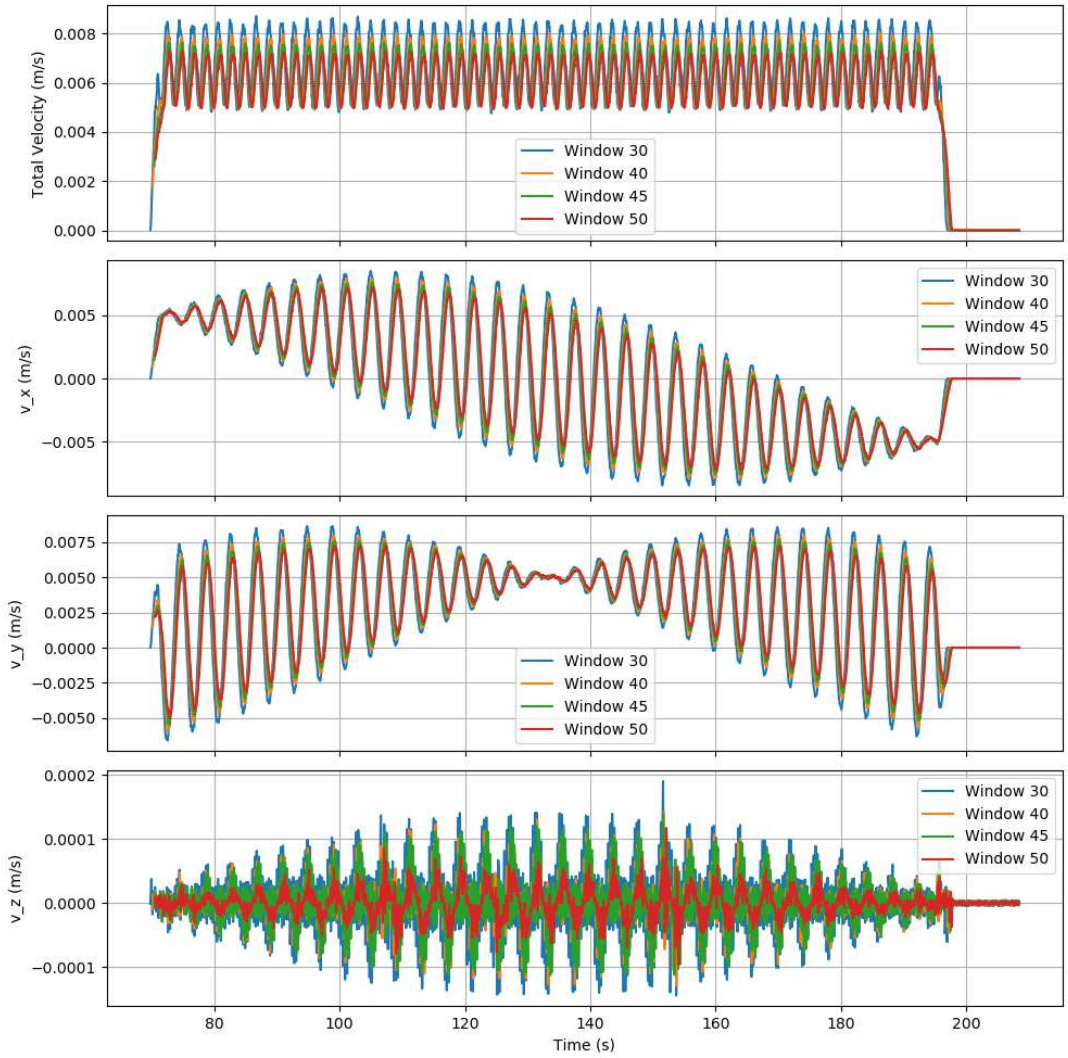
Slika 8 Prikaz isplanirana trajektorija



Slika 9 Prikaz filtrirana trajektorija



Slika 10 Prikaz izvršene trajektorije

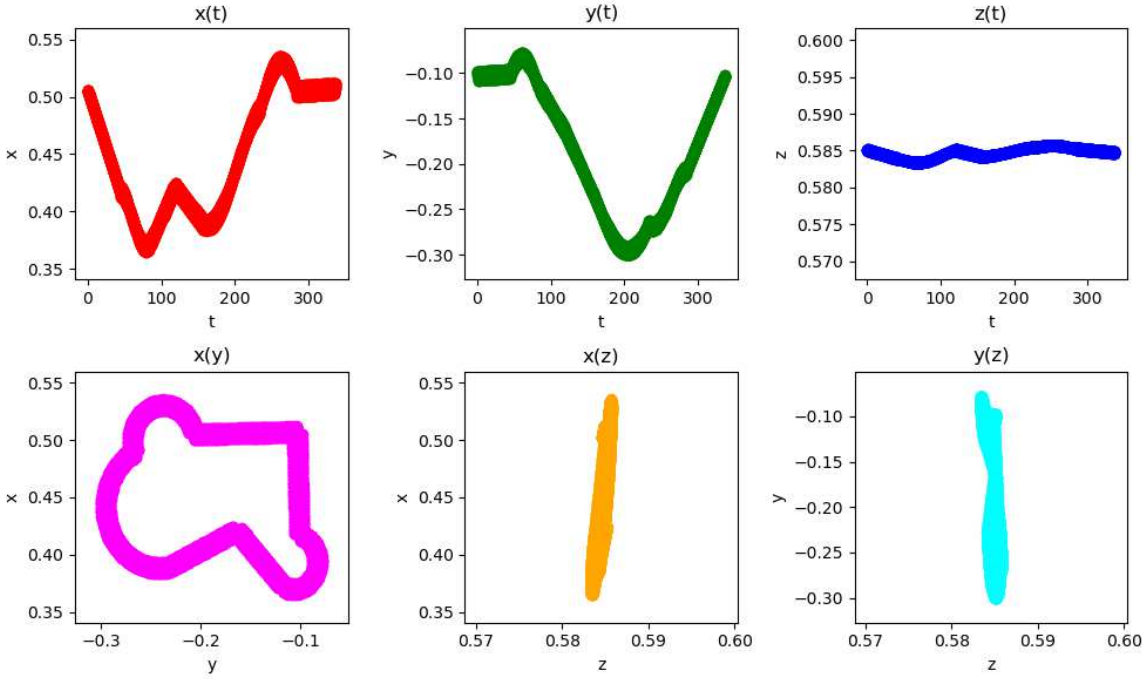


Slika 11 Prikaz brzina izvršene trajektorije

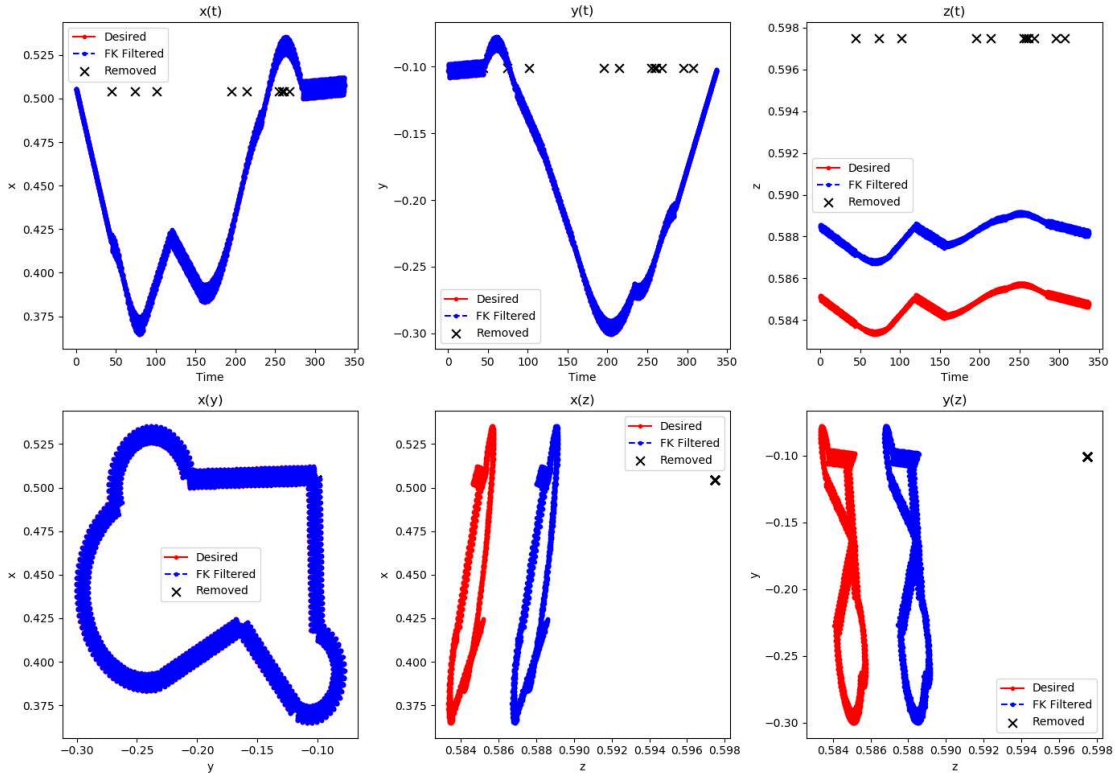
3. Trajektorija od niza kružnih i linearnih segmenata koja je ručno označena korištenjem vrha alata robota. Ova trajektorije je također izvršena u x-y ravnini na stolu koji je paralelan s bazom robota.

Za trapezni profil i oscilacije su parametri:

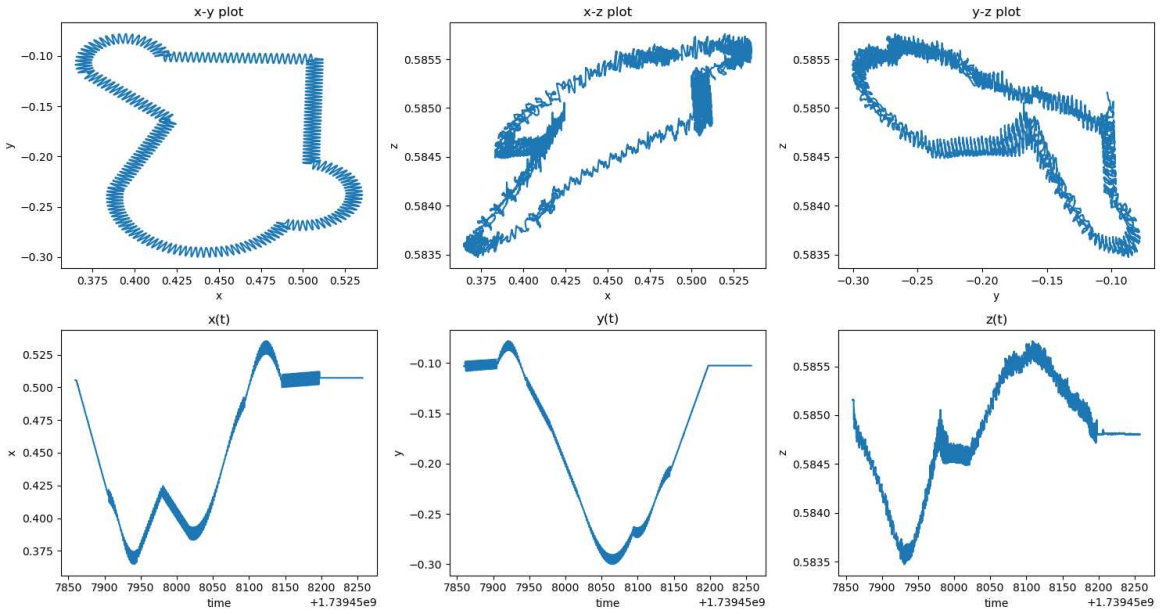
$$v_{\max} = 0.002,\ a_{\max} = 0.01, freq = 3, amplitude = 0.005$$



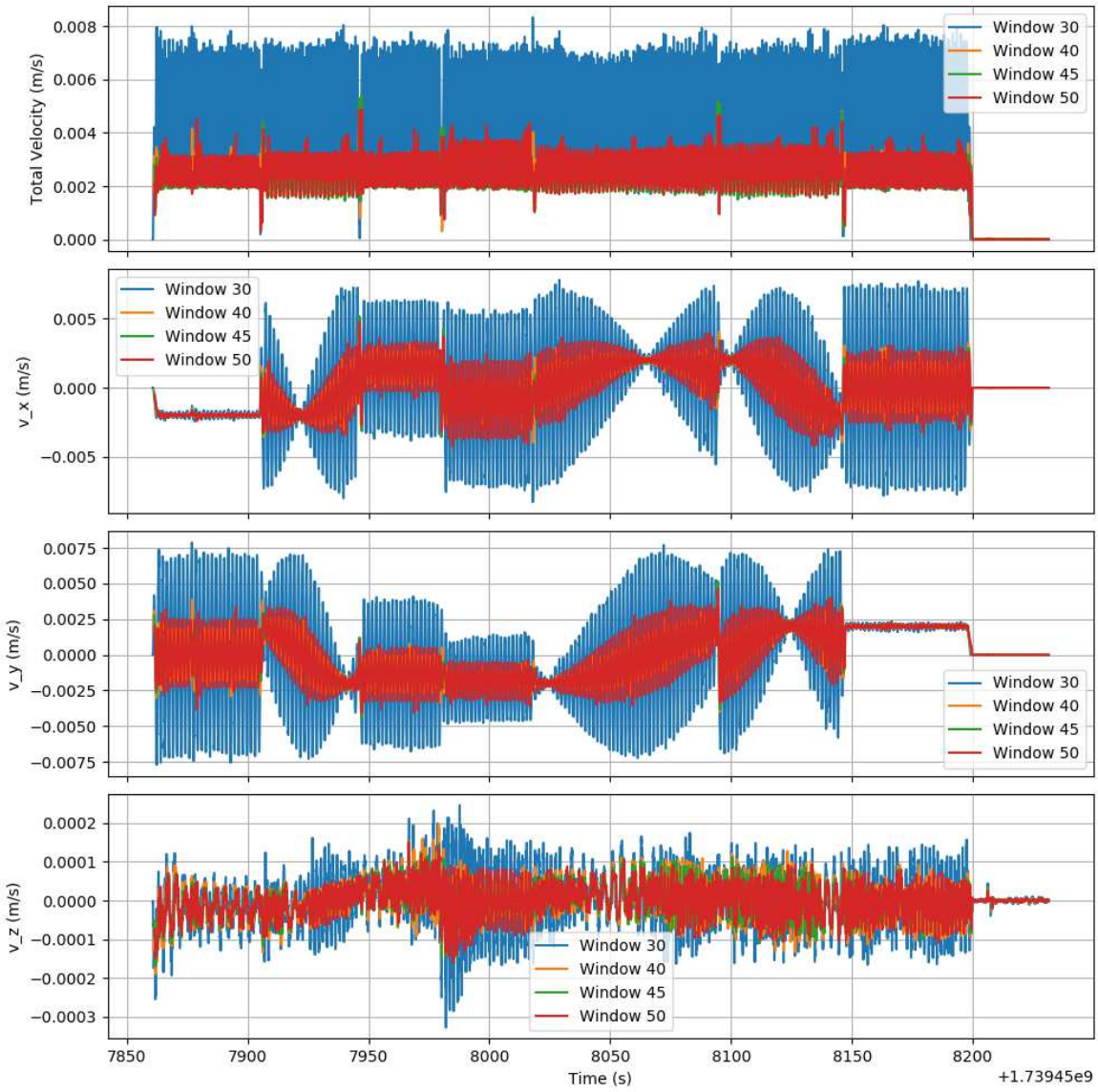
Slika 12 Prikaz isplanirane trajektorije



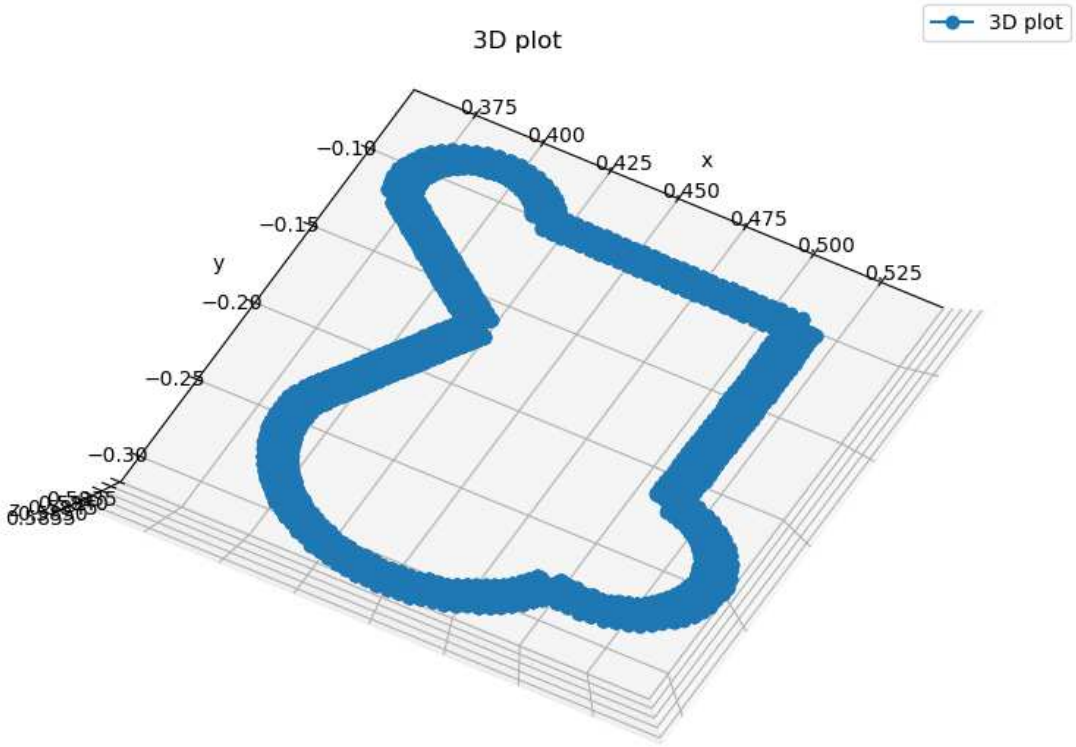
Slika 13 Prikaz filtrirane trajektorije



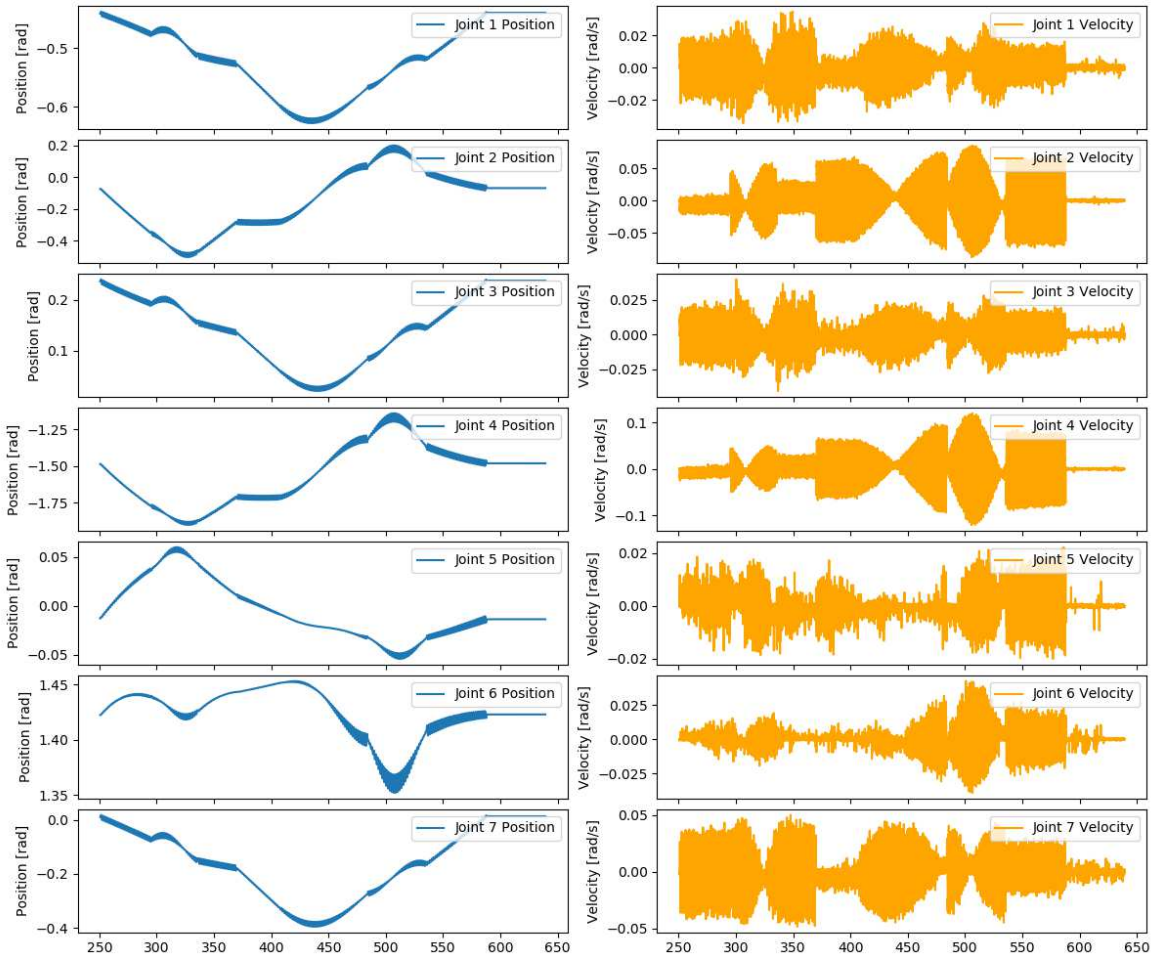
Slika 14 Prikaz izvršene trajektorije



Slika 15 Prikaz brzina izvršene trajektorije



Slika 16 Prikaz izvršene putanje u 3D



Slika 17 Prikaz pozicija i brzina zglobova tokom izvršavanja trajektroije

Zaključak

U ovom diplomskom radu predstavljen je čovjekom potpomognut sustav za označavanje okoline i generiranje trajektorija zavarivanja korištenjem robotskog manipulatora Franka. Sustav kombinira intuitivno definiranje ključnih točaka u radnom prostoru pomoću virtualne olovke (Kalipen) s naprednim algoritmima za generiranje linearnih i kružnih trajektorija. Posebna pažnja posvećena je kalibraciji, pri čemu su razvijene metode za određivanje transformacijskih matrica između koordinatnih sustava Optitrack-a, virtualne olovke i robota.

Primjenom trapeznog profila brzine i dodatnih sinusnih oscilacija postiže se konstantna brzina kretanja, čime se osigurava ravnomjerna kvaliteta zavarenih spojeva. Eksperimenti provedeni u simulacijskom okruženju te na stvarnom robotskom sustavu potvrdili su preciznost izračunate trajektorije, što ukazuje na primjenjivost u dinamičnim proizvodnim okruženjima. Sustav smanjuje potrebu za specijalizirane stručnjake, omogućujući jednostavnije programiranje i brzu prilagodbu promjenjivim radnim uvjetima.

Unatoč postignutim rezultatima, postoje mogućnosti za daljnja istraživanja, osobito u spajanju segmenata trajektorije tako da ne dolazi do stajanja u međutočkama. Također je u generiranim trajektorijama orijentacija alata konstanta što nije slučaj kod pravih poslova zavarivanja.

Programski kod je ostvaren s mogućnostima za daljnju nadogradnju i promjenom svih dijelova za generiranje trajektorije. Omogućava primjenu na različitim robotskim sustavima koji su implementirani u ROS okruženju. Moguće je dodavanje novih vrsta putanja, nadogradnja u smislu preciznije inverzne kinematike ili programske dijelove koristiti za provjeru i objavljivanje trajektorija.

Zaključno, ovaj rad predstavlja važan doprinos u razvoju intuitivnih i prilagodljivih robotskih sustava, otvarajući nove mogućnosti za implementaciju robotike u industrijskim i malim proizvodnim postrojenjima.

Literatura

[1] B. Marić, F. Zorić, A. Ivanović, F. Petrić, i M. Oršag, **"Human-friendly robot systems deployment using Virtual Pen"**. Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, Laboratory for Robotics and Intelligent Control Systems (LARICS).

[2] B. Marić, M. Polic, T. Tabak, i M. Oršag, **"Unsupervised optimization approach to in situ calibration of collaborative human-robot interaction tools"**, u: Proceedings of the 2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), 2020

[3] https://www.optitrack.com/

[4] https://github.com/larics/for_franka_ros ROS paket za pozivanje inverzne i direktne kinematike

[5] https://github.com/andrejjurman/Diplomski-Rad/tree/main Repozitorij programskog koda.

Sažetak

Čovjekom potpomognut robotizirani sustav zavarivanja

Ovaj rad predstavlja razvoj čovjekom potpomognutog sustava za označavanje okoline i generiranje trajektorija zavarivanja pomoću robotskog manipulatora. Sustav omogućava intuitivno definiranje ključnih točaka u radnom prostoru, čime se prikupljaju podaci potrebni za optimizaciju putanja zavarivanja s konstantnom brzinom. Implementirane su metode za kalibraciju olovke za označavanje prostora te se primjenjuju algoritmi za generiranje linearnih i kružnih trajektorija zavarivanja. Rezultati su testirani u simulacijskoj okolini te na stvarnom robotskom manipulatoru Franka.

Robotski manipulator; Zavarivanje; Generiranje trajektorije;

Summary

Human-assisted robotic welding system

This paper presents the development of a human-assisted system for environment marking and welding trajectory generation using a robotic manipulator. The system enables the intuitive definition of key points in the workspace, thereby gathering the data necessary for optimizing welding trajectories at a constant speed. Methods for calibrating the marking pen have been implemented, and algorithms for generating both linear and circular welding trajectories are applied. The results were tested in a simulation environment and on the real Franka robotic manipulator.

Robotic manipulator; Welding; Trajectory generation;