Implementacija programskog rješenja virtualnog modela robota u stvarnoj okolini

Topić, Marta

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:260242

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Marta Topić

Zagreb, 2024. godina.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Izv. prof. dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Marta Topić

Zagreb, 2024. godina.

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

Htjela bih izraziti svoju iskrenu zahvalnost svima koji su mi pomogli i podržali me tijekom izrade ovog diplomskog rada.

Prije svega, zahvaljujem svojem mentoru na neizmjernom strpljenju, podršci i vođenju. Vaši savjeti bili su od neprocjenjive vrijednosti te sam iznimno zahvalna što sam imala priliku učiti od Vas.

Također, želim se zahvaliti svojoj obitelji i prijateljima koji su me uvijek podržavali i ohrabrivali, kako tijekom izrade ovoga rada, tako i tijekom cijelog mojeg školovanja. Bez vaše podrške, ovo putovanje ne bi bilo moguće.

Još bih se htjela zahvaliti Bogu uz čiju sam milost i vodstvo uspješno završila još jedno poglavlje svoga života. Hvala Ti.

Marta Topić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Procesno-energetski, konstrukcijski, inženjersko modeliranje i računalne simulacije i brodostrojarski

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 24 - 06 / 1**	
Ur.broj: **15 - 24 -**	

DIPLOMSKI ZADATAK

Student: **Marta Topić** JMBAG: 0035214018

Naslov rada na hrvatskom jeziku: **Implementacija programskog rješenja virtualnog modela robota u stvarnoj okolini**

Naslov rada na engleskom jeziku: **Implementation of a software solution for a virtual robot model in a real environment**

Opis zadatka:

Model robota kao entitet koji je realiziran u sklopu virtualnog svijeta moguće je povezati s fizičkim robotom u sklopu stvarne okoline. Upravljanje robotom moguće je provesti programskim rješenjima koja se odvijaju sinkrono iz virtualne okoline koristeći princip digitalnih blizanaca ili asinkrono tako da se postojeći programi koji su testirani u virtualnoj okolini učitaju na fizičkog robota.
U radu je potrebno:

- koristeći RoboDK simulacijsko okruženje odabrati i definirati odgovarajućeg robota sa 6 stupnjeva slobode gibanja
- modelirati u simulacijskom okruženju radnu okolinu robota za poslove automatske robotske montaže
- koristeći Python programski jezik načiniti programsko rješenje za upravljanje robotom kod automatske robotske montaže
- razvijeno rješenje prenijeti i implementirati na stvarnom robotu te eksperimentalno evaluirati.

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

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
18. siječnja 2024.	21. ožujka 2024.	25. – 29. ožujka 2024.

Zadatak zadao:
Izv. prof. dr. sc. Tomislav Stipančić

Predsjednik Povjerenstva:
Prof. dr. sc. Tanja Jurčević Lulić

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
x	m	Udaljenost točke P od ishodišta O u smjeru x osi
ρ	m	Euklidska udaljenost između točke P i osi z
φ	Rad	Kut između osi x i vektora usmjerenog od ishodišta prema projekciji točke P na referentnu ravninu
y	m	Udaljenost točke P od ishodišta O u smjeru y osi
z	m	Udaljenost točke P od ishodišta O u smjeru z osi
θ	Rad	Kut između osi z i vektora usmjerenog od ishodišta prema točki P

SAŽETAK

U ovom diplomskom radu napravljena je simulacija automatske robotske montaže u simulatoru RoboDK te je implementirana na stvarnog UR5 robota. Najprije je opisana povijest programiranja robota, zatim koordinatni sustavi bitni u robotici te podjele robota. Također su navedeni i opisani izvršni članovi robota s naglaskom na hvataljke. Date su osnovne karakteristike robota UR5 i programa RoboDK te su opisani koraci nastanka simulacije, a na kraju je pokazano kako je simulacija uspješno spojena sa stvarnim robotom.

Ključne riječi: RoboDK, UR5, robotika, montaža, programiranje

SUMMARY

In this thesis, a simulation of automatic robotic assembly was made in the RoboDK simulator and implemented on a real UR5 robot. First, the history of robot programming is described, then the coordinate systems important in robotics and the division of robots. The executive members of the robot are also listed and described with an emphasis on grippers. The basic characteristics of the UR5 robot and the RoboDK program are given, and the steps of creating the simulation are described, and at the end it is shown how the simulation was successfully connected to the real robot.

Keywords: RoboDK, UR5, robotics, assembly, programming

1. UVOD

Programiranje robota može se koristiti u različitim područjima, uključujući industrijsku automatizaciju, medicinske aplikacije, istraživanje i razvoj, vojne svrhe, zabavu i obrazovanje. U svakom od tih područja programiranje robota omogućuje automatizaciju zadatka ili procesa, povećanje učinkovitosti i preciznosti, kao i obavljanje zadataka koji su opasni ili teško dostupni za ljude.

Zadatak ovog diplomskog rada je offline programirati robota te taj program učitati na fizičkog robota. Offline programiranje robota je proces programiranja robota koji se odvija izvan radnog okruženja robota, odnosno izvan samog proizvodnog postrojenja ili radnog mjesta. Ovaj pristup omogućuje programerima da razvijaju, testiraju i optimiziraju robotske zadatke u kontroliranom i sigurnom okruženju, često na računalu ili drugom programabilnom uređaju. Takav način programiranja robota donosi niz prednosti, uključujući smanjenje vremena zastoja proizvodnje, poboljšanu sigurnost, smanjenje troškova obuke osoblja te povećanu fleksibilnost i preciznost u programiranju. Osim toga, omogućuje programerima da istraže različite strategije i scenarije bez utjecaja na stvarnu proizvodnju.

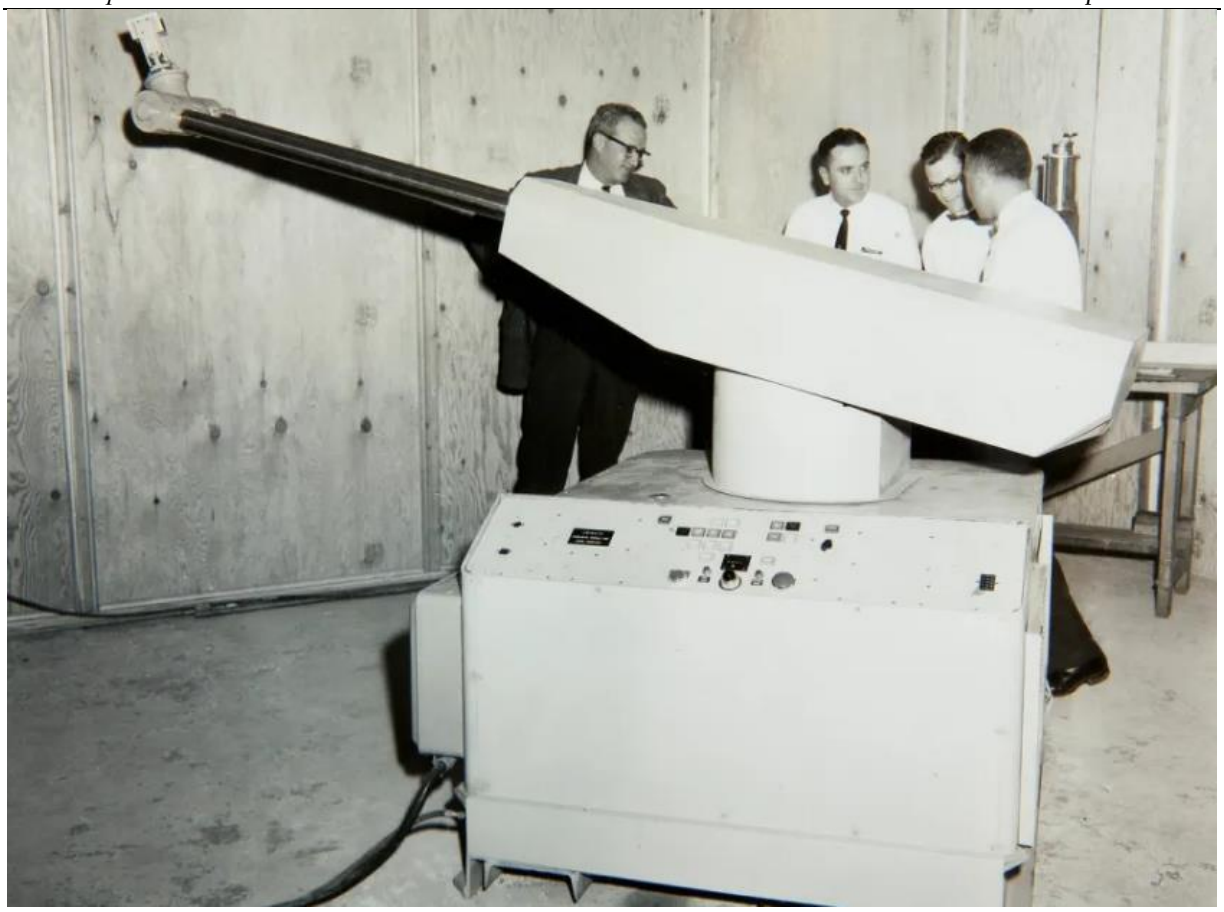
Zadatak koji robot treba obaviti je automatska robotska montaža. To je proces u kojem se roboti koriste za automatizaciju montaže proizvoda ili komponenti. Roboti se programiraju tako da obavljaju različite zadatke montaže, kao što su spajanje dijelova, pričvršćivanje vijaka, lemljenje, zavarivanje ili drugi postupci povezivanja dijelova kako bi se stvorio konačni proizvod. Automatska robotska montaža često se koristi u industrijskim postrojenjima za masovnu proizvodnju različitih proizvoda, kao što su elektronika, automobili, električni uređaji, medicinska oprema i druge vrste proizvoda. Ova tehnologija omogućuje proizvodnim tvrtkama povećanje učinkovitosti, smanjenje troškova proizvodnje i poboljšanje kvalitete proizvoda.

2. POVIJEST

Povijest programiranja robota izuzetno je složena, a posebno kada se uzme u obzir koliko je programiranje danas postalo jednostavno. Nedavni napretci u integraciji robotskog hardvera s popularnim jezicima, autonomnim robotima i upotrebom softverskih modula za uobičajene funkcije dodatno su olakšali proces.

Prvi pravi programski jezik, Plankalkül, razvio je njemački inženjer Konrad Zuse oko 1945. godine, ali on je bio za računala, a ne za robote. Programiranje robota započelo je nekoliko desetljeća kasnije.

Povijest programiranja robota može se podijeliti u tri generacije te one označavaju ključne faze u razvoju programskih jezika za upravljanje robotima. Nakon Drugog svjetskog rata, industrija je počela težiti automatizaciji radi smanjenja troškova proizvodnje, što je dovelo do razvoja prvog industrijskog robota, nazvanog Unimate, koji je 1961. godine razvio Joseph Engelberger, patentiravši ga 1954. godine zajedno s Georgom Devolom. Pojava Unimatea označava početak prve generacije programiranja robota. Unimate je bio korišten u proizvodnji automobila tvrtke General Motors, gdje je manipulirao vrućim komadima lijevanog željeza, a prikazan je na Slici 1. Ovaj robot učio je putem ručnog pomicanja u određene pozicije, koje bi kasnije replicirao, što je slično modernom konceptu učenja robota.



Slika 1. Prvi industrijski robot – Unimate [4]

Prvi poznati robotski jezik bio je MHI, razvijen 1960. godine. Tek u 1970-ima pojavili se se prvi općeniti programski jezici za robote. Kategorizacija programskih jezika u tri generacije pomaže u razumijevanju njihovog evolucijskog puta, iako je važno napomenuti da su se jezici unutar iste generacije često razlikovali.

Prva generacija robotskih jezika, poput VAL-a koji je razvijen 1973. na Sveučilištu Stanford, omogućila je programiranje robota na primitivnoj razini. Jezici ove generacije temeljili su se na postojećim programskim jezicima poput COBOL-a, ALGOL-a i Fortrana te su bili ograničeni u interakciji sa senzorima i drugim robotima.

Jezici druge generacije, kao što je VAL II, razvijenog 1982., donijeli su napredniju kontrolu pokreta i mogućnosti sučelja senzora. Iako je industrijska robotika ostala prilično konzervativna u korištenju jezika druge generacije, sami su jezici nastavili evoluirati, te su usvajali značajke iz drugih područja programiranja.

Treća generacija robotskog programiranja više se ne fokusira na specifične jezike. Ova generacija više se percipira kao skup programskih ideja koje se neprestano razvijaju, nego kao

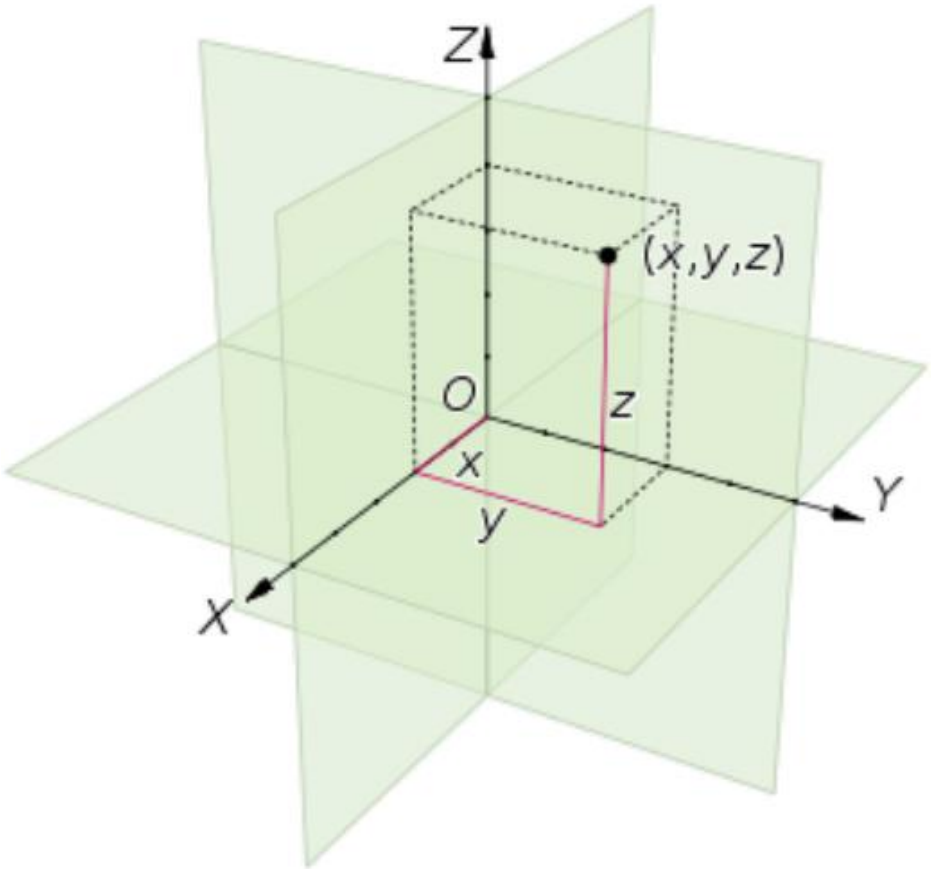
određeni set jezika. Značajke treće generacije uključuju automatske 3D modele okoline te dinamičko razumijevanje okoline. Osim treće generacije jezika, možemo razmotriti i treću generaciju robotike koja uvodi poučavanje demonstracijom, strojno učenje i nadzornu kontrolu. U budućnosti, vidimo trend prema robotima koji sami uče i šire svoje sposobnosti što postavlja nove izazove i mogućnosti kako za korisnike tako i za proizvođače robota.

3. KOORDINATNI SUSTAVI

Koordinatni sustavi omogućuju opisivanje točaka na krivulji, pravcu, plohi, u ravnini ili prostoru pomoću brojeva. Oni su ključni za razumijevanje trodimenzionalnog prostora te su nužni za ispravno i precizno djelovanje robota. U 3D prostoru postoje tri osnovna koordinatna sustava: kartezijev, cilindrični i sferni.

3.1. Kartezijev koordinatni sustav

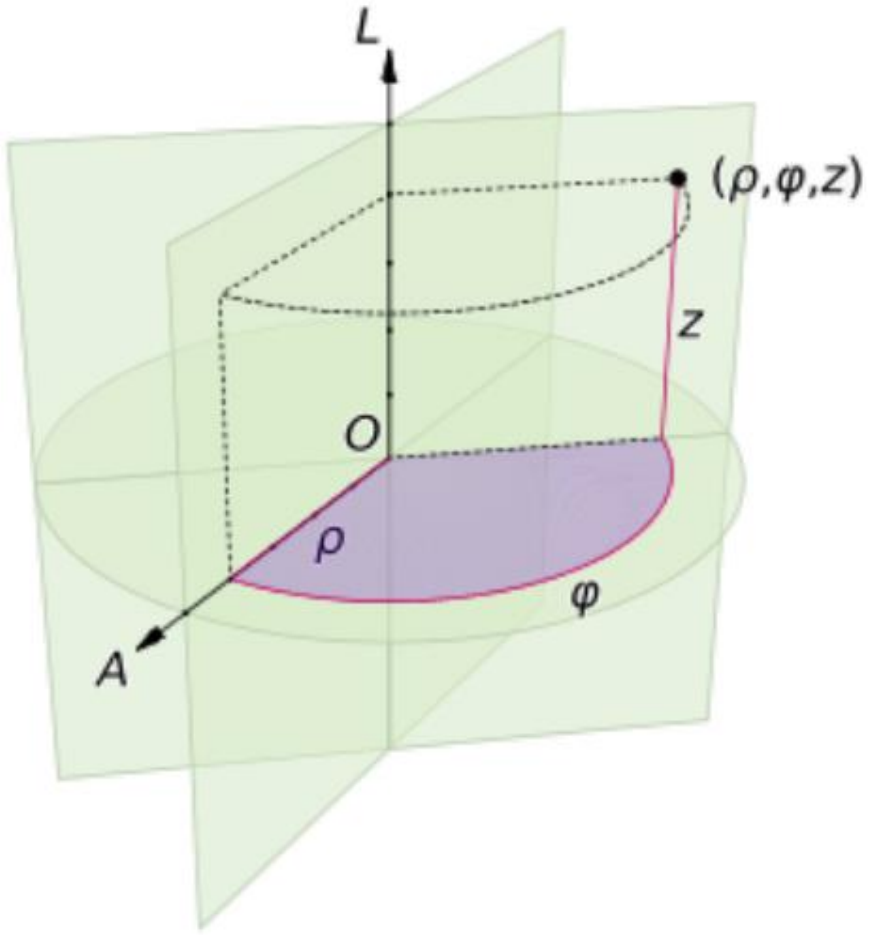
Kartezijev koordinatni sustav u trodimenzionalnom prostoru određen je sa tri međusobno okomita pravca x, y i z, koji se sijeku u ishodištu O. Koordinate na ovim osima nazivaju se apscisa (na osi x), ordinata (na osi y) i aplikata (na osi z). Na taj način svakoj točki u prostoru pridružena su tri realna broja (x, y, z). Kartezijev koordinatni sustav najčešće se primjenjuje u robotici. Na Slici 2. prikazan je kartezijev koordinatni sustav.



Slika 2. Kartezijev koordinatni sustav [5]

3.2. Cilindrični koordinatni sustav

Cilindrični koordinatni sustav u prostoru određen je ishodištem O, zrakom p s početkom u ishodištu i pravcem z koji je okomit na zraku p i prolazi kroz ishodište. Koordinate točke P u ovom sustavu označavaju se sa ρ, φ i z, gdje ρ predstavlja udaljenost okomitu na pravac z od točke P do ishodišta, φ je kut koji projekcija vektora OP zatvara na ravninu u kojoj se nalazi zraka p sa zrakom p, dok z predstavlja udaljenost paralelnu s osi z od točke P do ishodišta. Na sljedećoj slici vidi se cilindrični koordinatni sustav.



Slika 3. Cilindrični koordinatni sustav [5]

Prijelaz iz kartezijevih u cilindrične koordinate računa se prema sljedećim izrazima:

$$\rho = \sqrt{x^2 + y^2}$$

$$\varphi = arctan\frac{y}{x}$$

$$z = z$$

Dok se prijelaz iz cilindričnih u kartezijeve koordinate računa prema:

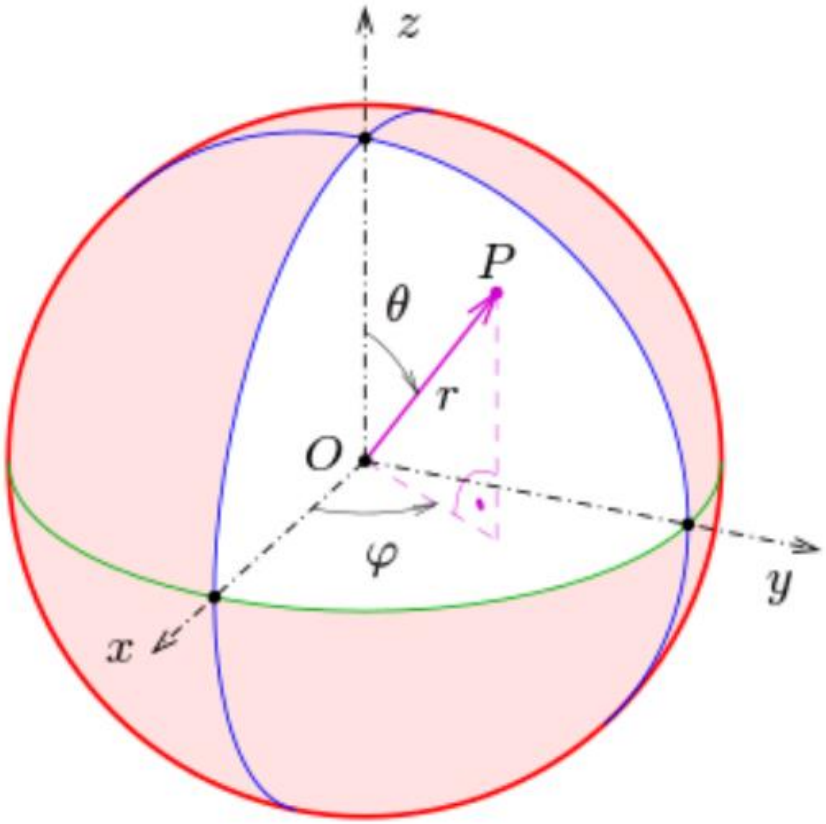
$$x = \rho \cdot \cos\varphi$$

$$y = \rho \cdot \sin\varphi$$

$$z = z$$

3.3. Sferni koordinatni sustav

Sferni koordinatni sustav u prostoru određen je ishodištem O i međusobno okomitim polupravcima p i z. Koordinate točke P u ovom sustavu označavaju se s r, φ i θ, gdje je r udaljenost od ishodišta do točke P, φ je kut od projekcije vektora OP na ravninu okomitu na poluos z i koordinatu θ, dok je θ kut koji vektor OP zatvara s poluosi z. Na Slici 4. prikazan je sferni koordinatni sustav.



Slika 4. Sferni koordinatni sustav [5]

Jednadžbe koje opisuju prijelaz iz kartezijevih u sferne koordinate su:

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$\theta = arccos\frac{z}{\sqrt{x^2 + y^2 + z^2}} = arccos\frac{z}{r}$$

$$\varphi = arctan\frac{y}{x}$$

A prijelaz iz sfernih u kartezijeve koordinate računa se prema izrazima:

$$x = r \cdot sin\theta \cdot cos\rho$$

$$y = r \cdot sin\theta \cdot sin\rho$$

$$z = r \cdot cos\theta$$

3.4. Koordinatni sustavi robota

3.4.1. Globalni koordinatni sustav

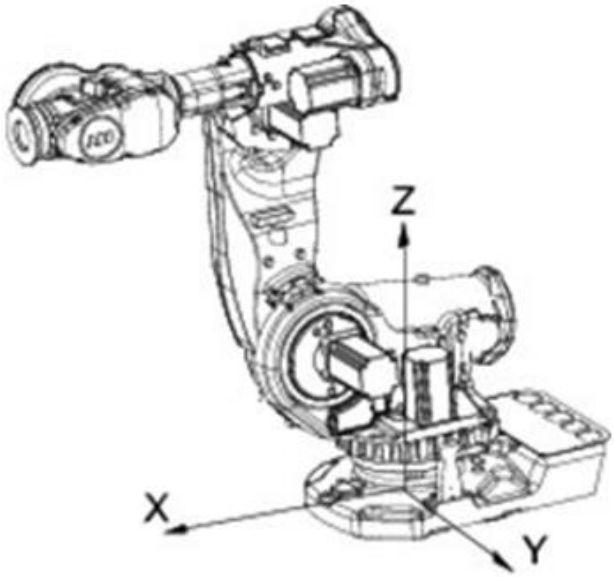
Globalni koordinatni sustav fiksiran je u prostoru i može biti zajednički za više uređaja. Također se može podudarati s neki drugim koordinatnim sustavom.

3.4.2. Lokalni koordinatni sustav

Lokalni koordinatni sustav definira se u odnosu na globalni koordinatni sustav na dijelovima mehanizma robota koji se relativno kreću.

3.4.3. Koordinatni sustav baze

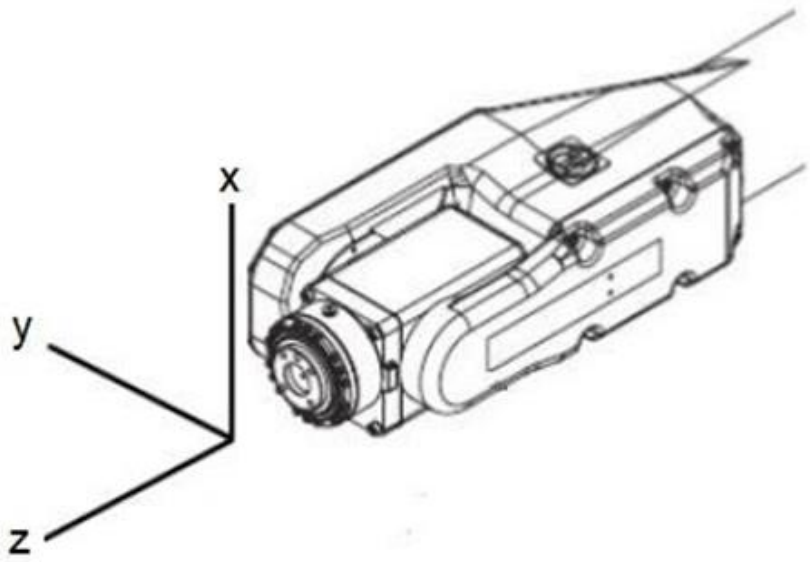
Koordinatni sustav baze zadani je koordinatni sustav robota koji definira njegovu poziciju u odnosu na globalni koordinatni sustav, a ishodište mu je fiksirano za bazu robota. Prikazan je na Slici 5.



Slika 5. Koordinatni sustav baze [17]

3.4.4. Koordinatni sustav prirubnice

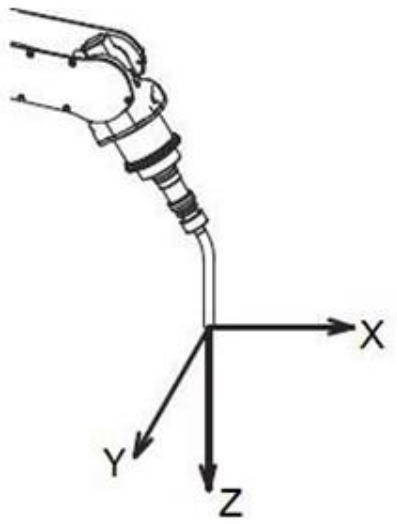
Koordinatni sustav prirubnice definiran je na prirubnici robota, iznad hvataljke te mu se ishodište nalazi u geometrijskoj sredini prirubnice. Tvornički je postavljen što znači da se ne može mijenjati, a njegova svrha je određivanje pozicije i orijentacije koordinatnog sustava alata. Na sljedećoj slici vidimo koordinatni sustav prirubnice.



Slika 6. Koordinatni sustav prirubnice [17]

3.4.5. Koordinatni sustav alata

Koordinatni sustav alata definira se u odnosu na koordinatni sustav prirubnice te se može pozicionirati ovisno o potrebama korisnika. Najčešće se postavlja u TCP točki (središnjoj točki alata). Na Slici 7. je prikazan ovaj koordinatni sustav.



Slika 7. Koordinatni sustav alata [17]

3.4.6. Koordinatni sustav objekta

Koordinatni sustav objekta definiran je na objektima kojima robot manipulira.

4. PODJELA ROBOTA

Roboti se dijele prema vrsti pogona, geometriji radnog prostora i načinu upravljanja kretanjem.

4.1. Vrsta pogona

- Električni motor

Električni motori (istosmjerni, izmjenični i koračni) danas se upotrebljavaju za pogon većine robota. Prednost im je što su relativno jeftini, imaju veliku brzinu i točnost te je kod ovakvog pogona omogućena primjena složenih algoritama upravljanja.

- Hidraulički pogon

Roboti s hidrauličkim pogonom primjenjuju se kod manipulacije velikim teretima. Brzina rada im je zadovoljavajuća, a zbog nestlačivosti ulja pružaju mogućnost mirnog održavanje položaja. Nedostatci ovih robota su onečišćenje okoliša radi opasnosti od istjecanja ulja, buka te visoke cijene.

- Pneumatski pogon

Roboti s pneumatskim pogonom prikladni su za laboratorijski rad zbog toga što imaju veliku brzina rada i ne onečišćuju okoliš, a i cijena im je niska. Također, upotrebljavaju se kod rada s lomljivim predmetima. S druge strane, nisu najbolji izbor za rad s velikim teretima jer nije moguće položaj održavati mirnim zbog stlačivosti zraka. Nadalje, proizvode veliku buku te je potrebno i dodatno filtrirati i sušiti zrak.

4.2. Geometrija radnog prostora

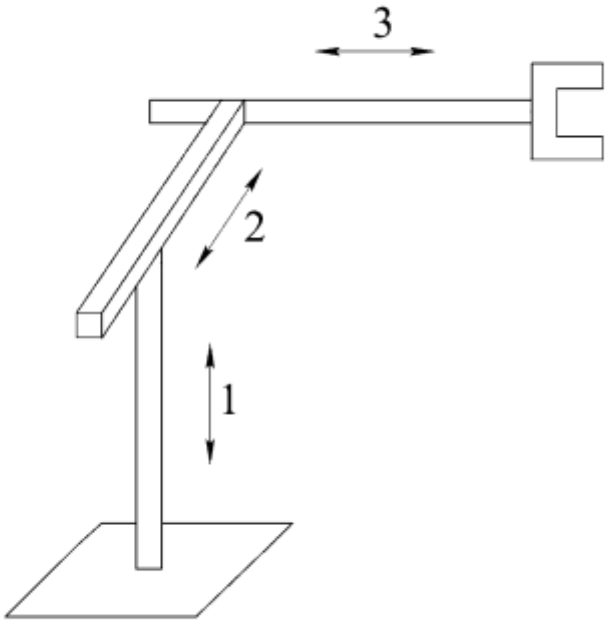
Skup točaka u trodimenzionalnom prostoru koje je moguće dohvatiti ručnim zglobom robota nazivaju se radnim prostorom robota. Veličina tog radnog prostora ovisna je o broju i tipu zglobova, duljinama članaka te o postojećim fizičkim ograničenjima koja su neposredno povezana s konkretnom građom i izgledom robota. [6]

Za industrijske robote upotrebljavaju se dva tipa zglobova: rotacijski (rotira oko osi) i translacijski (giba se linearno duž osi). Kombiniranjem rotacijskih i translacijskih zglobova za prve tri osi dobivene su različite konfiguracije robota:

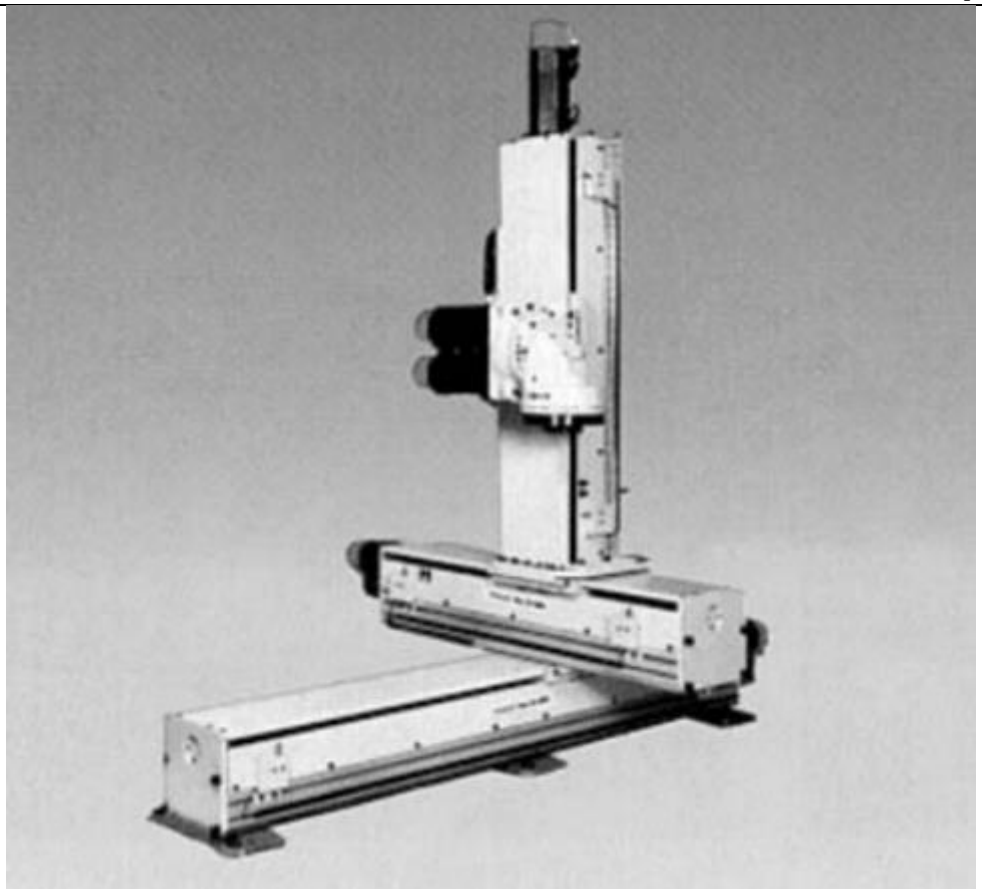
- Pravokutna
- Cilindrična
- Sferna
- Rotacijska
- Robot tipa SCARA

4.2.1. Pravokutna konfiguracija robota

Kod robota s pravokutnom konfiguracijom koriste se tri translacijska zgloba te im je radni prostor prizma. Na Slici 8. vidimo shematski prikaz takvog robota, a na Slici 9. vidimo industrijskog robota pravokutne konfiguracije.



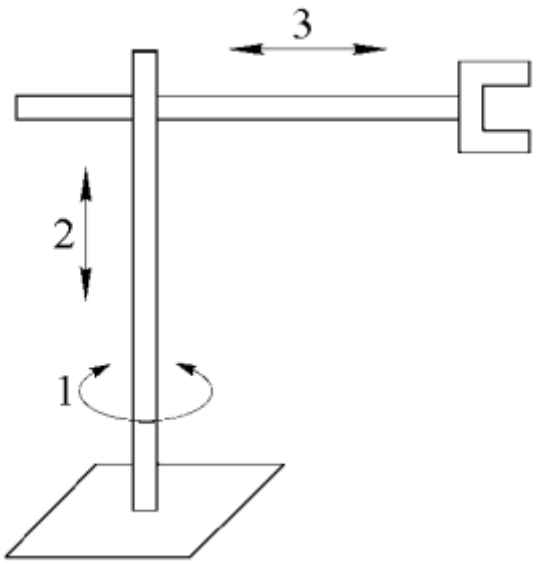
Slika 8. Shematski prikaz robota s pravokutnom konfiguracijom [6]



Slika 9. Prikaz industrijskog robota s pravokutnom konfiguracijom [6]

4.2.2. Cilindrična konfiguracija robota

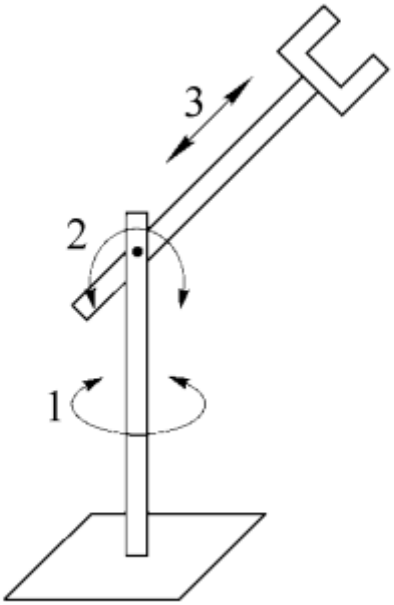
Kako bi se dobila cilindrična konfiguracija, koristi se jedan rotacijski zglob i dva translacijska zgloba, a kako je translacijsko gibanje ograničeno, radni prostor robota je volumen između dva vertikalna koncentrična plašta valjaka. Slika 10. prikazuje robota s cilindričnom konfiguracijom.



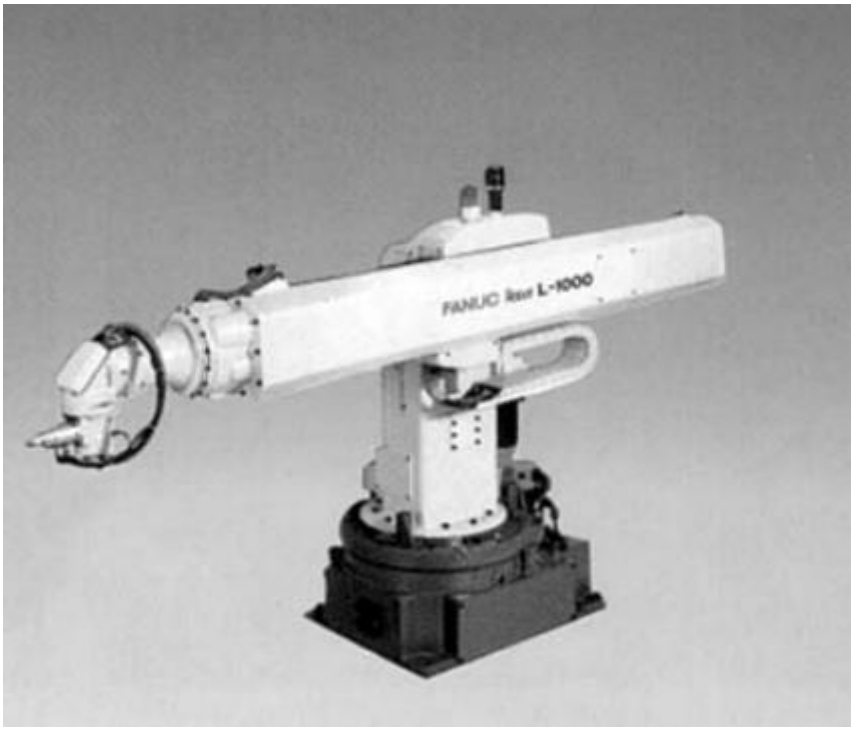
Slika 10. Shematski prikaz robota s cilindričnom konfiguracijom [6]

4.2.3. Sferna konfiguracija robota

Robot sa sfernom konfiguracijom ima dva rotacijska i jedan translacijski zglob. Ukoliko je translacijsko gibanje ograničeno, radni prostor robota je volumen između dviju koncentričnih sfera, a ako su ograničena sva gibanja, onda je radni prostor dio volumena između dviju koncentričnih sfera. Na sljedećim slikama prikazan je robot sa sfernom konfiguracijom.



Slika 11. Shematski prikaz robota sa sfernom konfiguracijom [6]

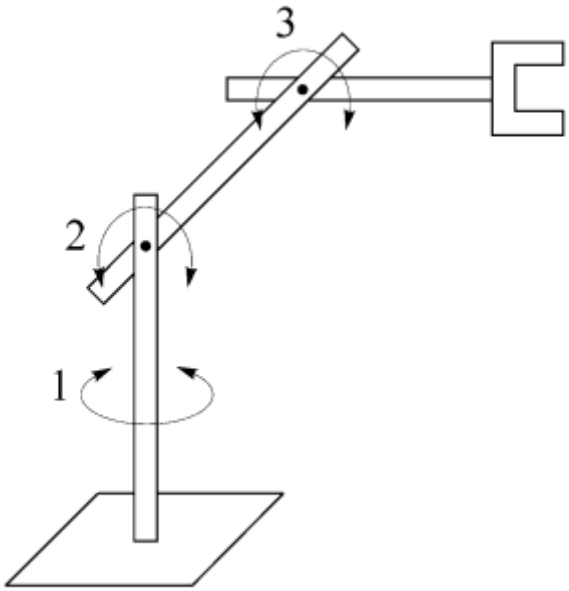


Slika 12. Prikaz industrijskog robota sa sfernom konfiguracijom [6]

4.2.4. Rotacijska konfiguracija robota

Robot s ovim tipom konfiguracije ima sva tri rotacijska zgloba. Rotacijska konfiguracija još se može zvati laktasta, antropomorfna ili zglobna. Kada ne postoje ograničenja gibanja, radni prostor je kugla, a ukoliko ograničenja postoje, radni prostor je dio kugle složenog oblika.

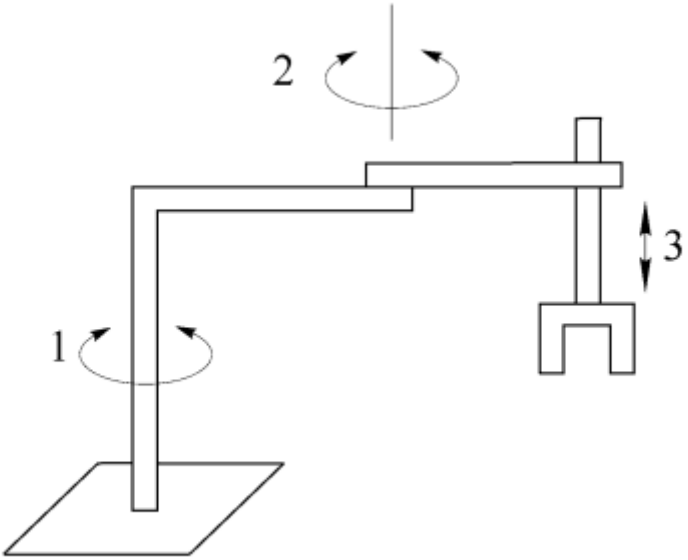
Na slici 13. vidi se prikaz robota s rotacijskom konfiguracijom.



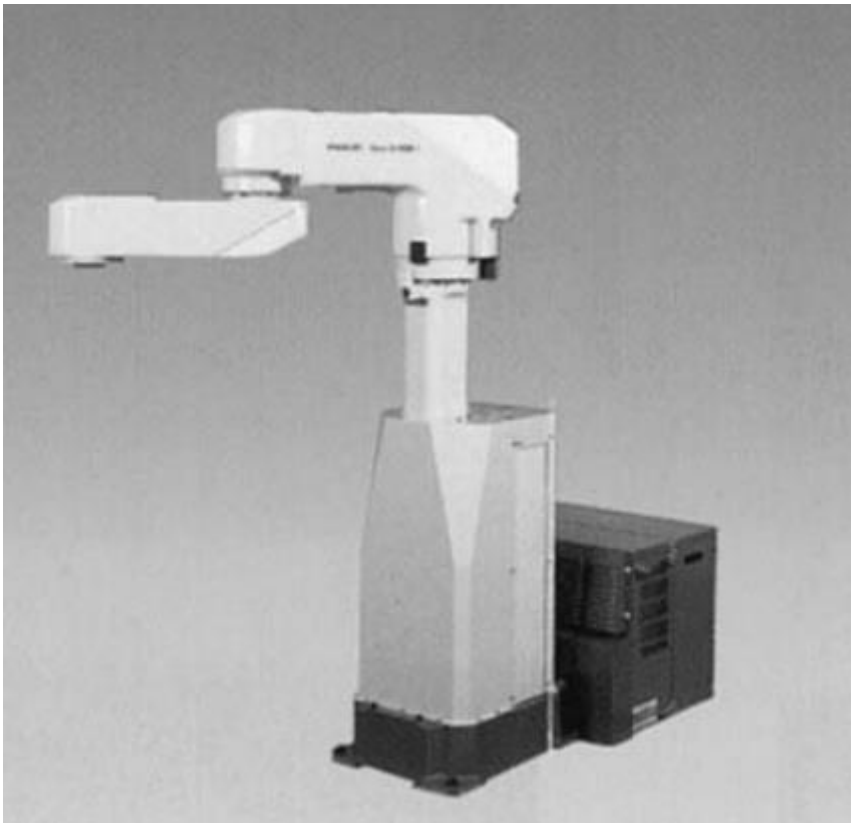
Slika 13. Shematski prikaz robota s rotacijskom konfiguracijom [6]

4.2.5. Robot tipa SCARA

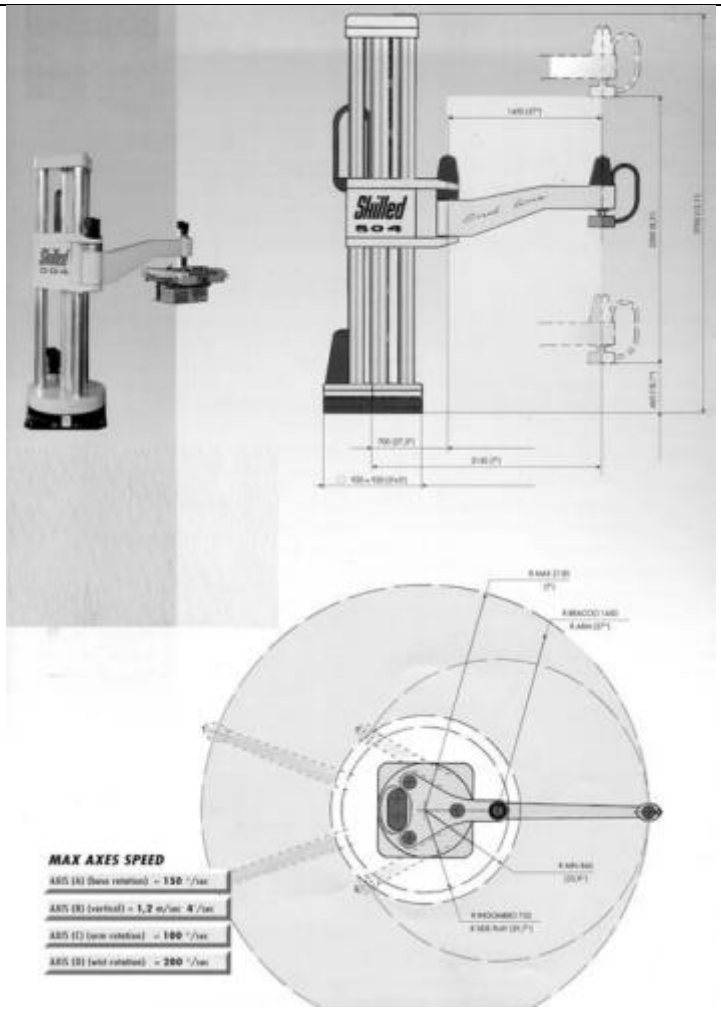
Ovaj tip robota ima dva rotacijska i jedan translacijski zglob. Ovisno o rasporedu tih zglobova, kod ovog robota razlikuju se tri vrste konfiguracije: RTR, TRR i RRT. Na sljedećim slikama vide se sve tri vrste konfiguracija.



Slika 14. Robot tipa SCARA (RRT konfiguracija) [6]



Slika 15. Robot tipa SCARA (TRR konfiguracija) [6]



Slika 16. Robot tipa SCARA (RTR konfiguracija) [6]

4.3. Način upravljanja kretanjem

Dva osnovna načina kretanja završnog mehanizma su:

- Kretanje od točke do točke

Koristi se kada je bitna točnost pozicioniranja, a putanja između točaka je manje važna. Primjeri korištenja su točkasto zavarivanje te podizanje i spuštanje predmeta.

- Kontinuirano gibanje po putanji

Ovaj tip kretanja koristi se kada su podjednako bitni i točnost pozicioniranja, a i putanja kojom se robot giba. Koristi se za bojanje, šavno zavarivanje i lijepljenje.

5. IZVRŠNI ELEMENTI ROBOTA

Izvršni elementi robota su dijelovi ili komponente robota koji izravno izvršavaju zadatke ili operacije koje su definirane u programu robota. Ovi elementi odgovorni su za fizičko izvođenje pokreta, manipulaciju predmetima, ili obavljanje drugih specifičnih zadataka koji su programirani u robotskom sustavu. Odabiru se ovisno o zadatku kojeg robot obavlja. Neki od mogućih izvršnih članova su: hvataljke, magneti, kamere, alati za bušenje, odvijači, alati za zavarivanje, pištolji za boju, senzori sile, alati za rezanje, alati za brušenje, itd. Najčešće korišteni izvršni elementi su hvataljke. Hvataljke možemo podijeliti na:

- Krute hvataljke

  Krute hvataljke koriste se za hvatanje i manipulaciju predmetima čvrstih i stabilnih oblika. Dizajnirane su tako da imaju čvrstu i stabilnu konstrukciju, često od čvrstih materijala što ima daje sposobnost nošenja težih predmeta. Primjeri predmeta kojima se manipulira čvrstim hvataljkama su kutije, palete, metalni dijelovi, itd.

- Mekane hvataljke

  Mekane hvataljke koriste se za hvatanje predmeta koji imaju različite oblike, veličine i teksture jer se mogu prilagoditi obliku predmeta kojim manipuliraju. Izrađene su od fleksibilnih materijala kao što su guma, silikon ili drugi elastomeri zbog čega je smanjena mogućnost oštećenja predmeta kojeg hvataju. To ih čini prikladnima za rukovanje osjetljivim ili krhkim predmetima bez rizika od oštećenja.

- Posebne hvataljke

  Posebne hvataljke robota su specijalizirane hvataljke koje su dizajnirane za obavljanje specifičnih zadataka ili manipulaciju određenim vrstama predmeta. Ove hvataljke često imaju jedinstvene značajke ili funkcionalnosti koje ih razlikuju od standardnih hvataljki.

Hvataljke se mogu podijeliti i prema načinu rada pa tako razlikujemo vakuumske, pneumatske, hidrauličke i električne.

5.1. Vakuumske hvataljke

Vakuumske hvataljke robota su vrsta hvataljki koje koriste vakuumsku snagu za hvatanje i manipulaciju predmetima. Koriste razliku u tlaku između unutarnje i vanjske strane hvataljke kako bi stvorile vakuumsku silu koja drži predmet na mjestu. Uglavnom se primjenjuju na predmetima ravnih površina i pravilnih oblika, a pogodne su za rukovanje osjetljivim i lomljivim predmetima kao što su staklo, keramika ili eketroničke komponente. Na sljedećoj slici prikazane su vakuumske hvataljke.

Slika 17. Vakuumske hvataljke [8]

5.2. Pneumatske hvataljke

Pneumatske hvataljke robota su hvataljke koje koriste komprimirani zrak kao izvor energije za pokretanje prstiju ili čeljusti kako bi hvatale i manipulirale predmetima. Ove hvataljke često se koriste u industrijskim postrojenjima zbog svoje jednostavnosti, brzine i pouzdanosti te mogućnosti prilagođavanja obliku i veličini predmeta koje hvataju. Negativna strana im je to što je za njihovo korištenje potreban pristup izvoru komprimiranog zraka. Također, zbog njihove brzine i snage, trebaju se koristiti u skladu s propisanim sigurnosnim mjerama kako bi se spriječile ozljede ili oštećenja predmeta koji se hvataju. Slika 18. prikazuje pneumatske hvataljke.

Slika 18. Pneumatske hvataljke [9]

5.3. Hidrauličke hvataljke

Hidrauličke hvataljke robota su hvataljke koje koriste hidraulični sustav za pokretanje prstiju ili čeljusti kako bi hvatale i manipulirale predmetima. Ove hvataljke koriste hidraulični tlak za generiranje potrebne sile za pokretanje i držanje predmeta. Imaju veliku snagu i preciznost pa se koriste za hvatanje teških predmeta. Nedostatak im je potreba za pristupom hidrauličnom izvoru energije i to što takav sustav zahtijeva redovno održavanje kako bi se osigurala njegova pouzdanost i učinkovitost. Sljedeća slika prikazuje hidrauličke hvataljke.

Slika 19. Hidrauličke hvataljke [17]

5.4. Električne hvataljke

Električne hvataljke robota su hvataljke koje koriste električnu energiju za pokretanje prstiju ili čeljusti kako bi hvatale i manipulirale predmetima. Ove hvataljke koriste elektromotore, elektromagnete ili druge električne aktuatore kako bi generirale potrebnu silu za hvatanje i držanje predmeta. Omogućuju visoku preciznost i kontrolu nad pokretima prstiju ili čeljusti, također imaju veliku snagu te prilagodljivost. Prikazane su na Slici 20.

Slika 20. Električne hvataljke [10]

6. ROBOT UR5

Robot UR5 pripada skupini kolaborativnih robota. Kolaborativni roboti (eng. cobot) su robotske ruke male težine koje imaju širok raspon primjene te su dizajnirani za rad s ljudima u zajedničkom radnom prostoru što je suprotno tradicionalnim industrijskim robotima koji su obično izolirani ili ljudi rade s njima pod zaštitom. Iako su sporiji i manje snažni od industrijskih robota, to se ne smatra nedostatkom jer moraju djelovati prilagođenom snagom i brzinom kako bi osigurali sigurnost ljudi u blizini.

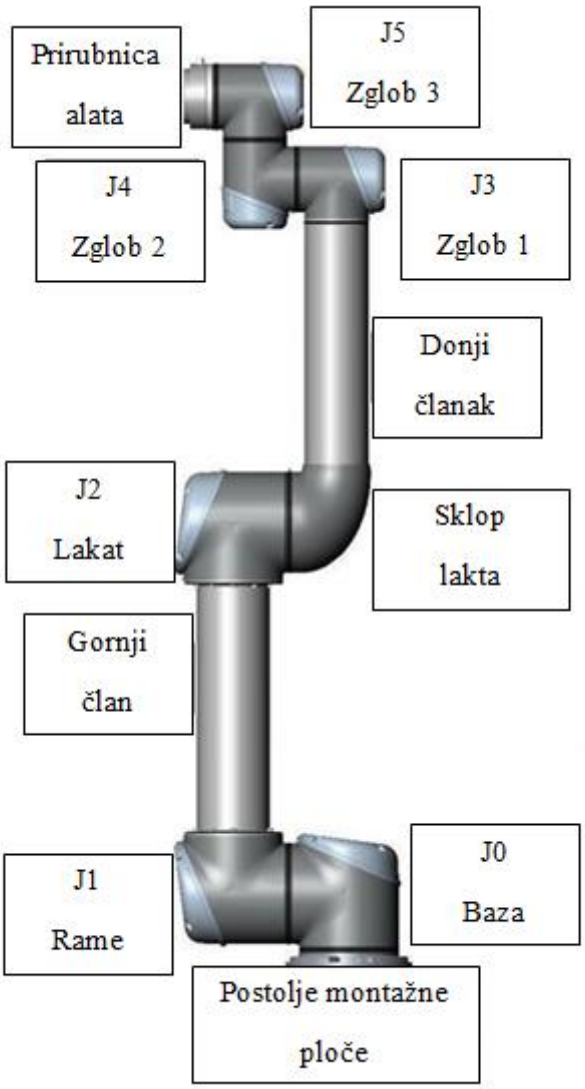
Koriste se za automatizaciju novih područja, posebno u blizini ljudi gdje je korištenje tradicionalnih industrijskih robota prezahtjevno i skupo. Neki primjeri korištenja robotske ruke su odabiranje i umetanje komponenti, sastavljanje komponenti, injekcijsko prešanje, upravljanje CNC strojem, paletiranje, probne kontrole, poliranje, lijepljenje, itd. Slika 21. prikazuje neke primjene robota UR5.

Slika 21. Primjeri primjene robota UR5 [18]

Glavne karakteristike kolaborativnih robota su sigurnost, jednostavnost korištenja i pristupačnost. Sigurnost kolaborativnog robota može se postići upotrebom laganih materijala, zaobljavanjem rubova, ograničenjem brzine i sile te upotrebom senzora i softvera koji kontroliraju sigurno ponašanje.

Robot UR5 u usporedbi s drugim robotima je vrlo lagan zato što se u potpunosti sastoji od aluminijskih cijevi i karika. Također je i vrlo fleksibilan te se može koristiti za obavljanje ponavljajućih zadataka i podizanje tereta do 5 kg. Ima 6 osi kojima može postići tražene položaje i orijentacije. Zbog svoje lagane konstrukcije prikladan je za primjene koje ne zahtijevaju velika opterećenja. Robot je opremljen senzorom sile koji se može podesiti po želji u grafičkom korisničkom sučelju i koristiti u određenim aplikacijama.

Na sljedećoj slici prikazan je sklop robotske ruke. J0, J1, J2, J3, J4, J5 su pomični zglobovi robota.



Slika 22. Sklop robotske ruke [18]

Sljedeća tablica prikazuje neke karakteristike robota UR5.

Tablica 1. Karakteristike robota UR5 [18]

Masa	18,4 kg
Opterećenje	5 kg
Doseg	850 mm
Pokretljivost zglobova	+/- 360^0 na svim zglobovima
Brzina	Zglob: najviše 180^0/s, Alat: približno 1 m/s
Ponovljivost	+/- 0,1 mm
Baza	Ø149 mm
Stupnjevi slobode	6 rotacijskih zglobova
Materijali	Aluminij, plastika PP

7. PROGRAMIRANJE ROBOTA

Robot se može programirati na tri različita načina:

- Izravno ili online programiranje

Robot se u proizvodnji programira direktno pomoću ručnog programatora koji koristi jednostavno i intuitivno grafičko korisničko sučelje nazvano Polyscope što čini ovu opciju vrlo popularnom među korisnicima UR robota. Prednost ove metode programiranja je što se osoba nalazi pokraj robota, a to omogućuje praćenje njegovih reakcija i brzo zaustavljanje u slučaju grešaka ili neočekivanih situacija. Ova metoda prikladna je za složenije primjene, poput zavarivanja pod teško dostupnim kutovima jer omogućuje ručno upravljanje robotom do željenih točaka.

- Neizravno ili offline programiranje

Programiranje robota je moguće korištenjem robotskog jezika URScript putem Ethernet TCP/IP veze u različitim programskim okruženjima. Također se često koristi kombinacija programiranja s RPN-om, gdje se određeni dijelovi programa uređuju na računalu, primjerice u Notepadu, te zatim importiraju u RPN.

Naredbe URScripta za kretanje i druge akcije robota mogu se slati s glavnog računala direktno robotu UR5 preko mreže, bez potrebe za RPN-om.

Prednost korištenja skriptnih naredbi i offline programiranja leži u tome što se program može razvijati i mijenjati dok robot obavlja druge zadatke u proizvodnji. Novi program se može učitati na robota kada se očekuje zastoj u proizvodnji, čime se omogućuje nesmetan rad ostatka proizvodnje.

- Simulacijski programi

Simulacijski programi postaju sve popularniji jer omogućuju planiranje kompletnih proizvodnih linija i integraciju robota u njih. Ovo omogućuje offline programiranje, a neki programi čak omogućuju istovremeno izvođenje simulacije i programa na stvarnom robotu. Primjeri takvih programa su ArtiMinds RPS i RoboDK. ArtiMinds RPS je posebno certificiran za UR robote i nudi dodatne module za korištenje senzora sile, robotske hvataljke, itd. S druge strane, RoboDK je univerzalniji simulator koji podržava korištenje više različitih robota i omogućuje sofisticiranije planiranje virtualnih simulacija.

8. RoboDK

RoboDK je softverski alat za simulaciju koji omogućuje dizajniranje, programiranje i simulaciju industrijskih robota u virtualnom okruženju. Objavljen je u siječnju 2015. godine. U današnjem dinamičnom i izrazito konkurentnom industrijskom okruženju, simulacija i testiranje robotskih aplikacija bez stvarnog pokretanja stroja velika je prednost. RoboDK nudi niz prednosti svojim korisnicima, uključujući povećanje produktivnosti, smanjenje vremena i troškova proizvodnje te poboljšanje sigurnosti. Softver se ističe korisnički prijateljskim sučeljem i širokom bibliotekom dostupnih robota i alata što ga čini pristupačnim i jednostavnim i za manje iskusne korisnike. Programiranje u RoboDK-u moguće je koristeći različite programske jezike poput Pythona, C# i C++ čime se olakšava pristup širokom spektru korisnika. Također podržava rad s više robota istovremeno, omogućujući korisnicima simulaciju složenih industrijskih procesa. Mogućnost testiranja i validacije robotskih programa bez fizičkog pokretanja stroja značajno smanjuje vrijeme i troškove povezane s programskim ispravcima, dok istovremeno eliminira potrebu za zaustavljanjem proizvodnje i strojeva što pozitivno utječe na produktivnost i rentabilnost.

Još jedna prednost korištenja RoboDK-a je sposobnost simuliranja kompleksnih industrijskih procesa i prepoznavanja potencijalnih sudara i drugih problema prije nego što se dogode u stvarnom okruženju. Ova mogućnost doprinosi povećanju sigurnosti stroja i smanjenju rizika od nesreća ili oštećenja opreme. Dodatno, softver generira detaljne izvještaje za svaki korak simulacije, što korisnicima omogućuje brzo i jednostavno rješavanje problema ili nedostataka u programu. Također pruža uvid u korake procesa i omogućuje korisnicima da se usredotoče na ključne aspekte. Nadalje, podržava integraciju s različitim CAD i CAM softverima, što olakšava proces izrade simulacije.

Njegova knjižnica sadrži više od osamsto robota različitih proizvođača. Koristeći virtualne modele u formatima .step, .iges ili .stl, možemo izgraditi kompletno radno okruženje robota i stvoriti virtualnu simulaciju. Pomoću simulatora RoboDK, možemo kreirati razne offline simulacije poput glodanja, crtanja, bojanja, zavarivanja, 3D ispisa i slično. U lokalnoj knjižnici programa nalazi se mnogo primjera ovih simulacija koje možemo preuzeti i proučavati njihovu izvedbu.

Slika 23. Logo tvrtke RoboDK [18]

U programu RoboDK moguće je kretanje robota programirati s tri različita pokreta:

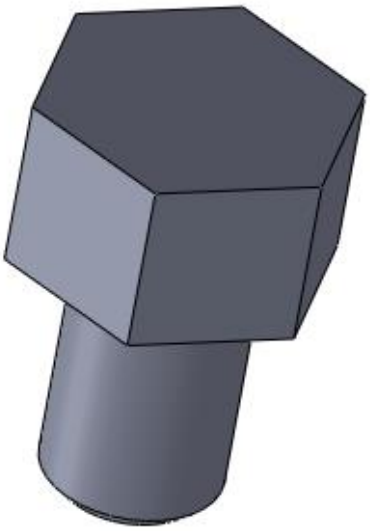
- Zglobni pomak (PTP) je osnovno i najčešće korišteno kretanje. Koristi se u slučajevima kada robot nema većih prepreka u radnom okruženju i nije bitno kojom rutom dolazi od točke A do točke B. Oznaka za takvo gibanje je MoveJ.
- Linearni pomak (LIN) je preciznije kretanje kod kojeg se robot kreće ravno po osi, najčešće se koristi pri približavanju i udaljavanju od objekta koji treba uhvatiti hvataljkom. Za ovo gibanje koristi se oznaka MoveL.
- Kružni pomak (CIRC) je rijetko korišteno gibanje. S njim se robot kreće u određenom krugu. Najprije se mora odrediti početna točka, a zatim središnja i krajnja točka, gdje će krug završiti. Oznaka je MoveP.

9. MODELIRANJE DIJELOVA

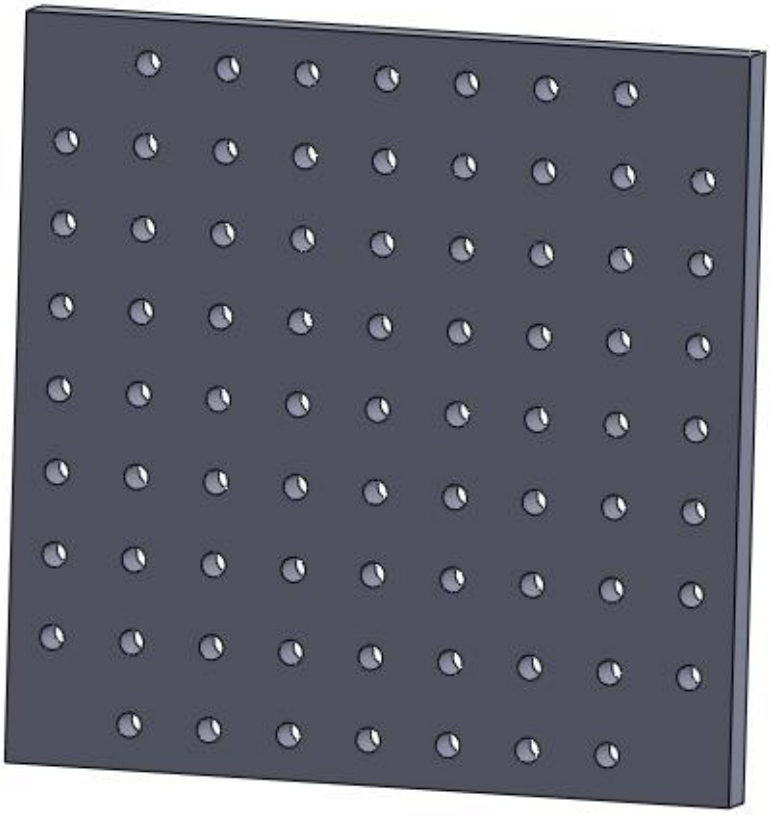
Prije nego što se započne sa simulacijom u programu RoboDK, potrebno je modelirati dijelove u Solidworksu. Prednosti modeliranja dijelova su:

- Preciznost simulacije
  Solidworks omogućuje detaljno modeliranje dijelova s visokom preciznošću što rezultira točnijom simulacijom u RoboDK-u.
- Interakcija s okolinom
  Modelirani dijelovi mogu se integrirati s okolinom, poput radnih stolova ili strojeva, a to omogućuje realističniju simulaciju kretanja robota u stvarnom radnom okruženju.
- Kolizije i provjere stvarnosti
  Modelirani dijelovi omogućuju provjeru mogućih kolizija i interakcija s drugim objektima u radnom okruženju prije stvarne implementacije.
- Optimizacija procesa
  Solidworks omogućuje dizajniranje i optimizaciju dijelova prije nego što se simulacija izvede u RoboDK-u te se tako može ubrzati proces proizvodnje, a mogu se i smanjiti troškovi.
- Fleksibilnost i prilagodljivost
  Korištenje Solidworksa omogućuje korisnicima veću fleksibilnost i prilagodljivost u dizajniranju dijelova pa to rezultira boljom simulacijom i kvalitetnijim rezultatima u RoboDK-u.

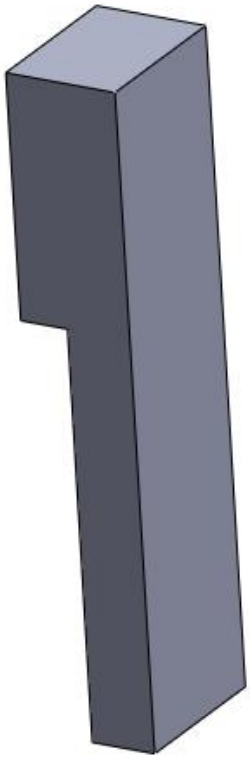
Na sljedećim slikama prikazani su dijelovi koje je bilo potrebno modelirati kod ovog zadatka.



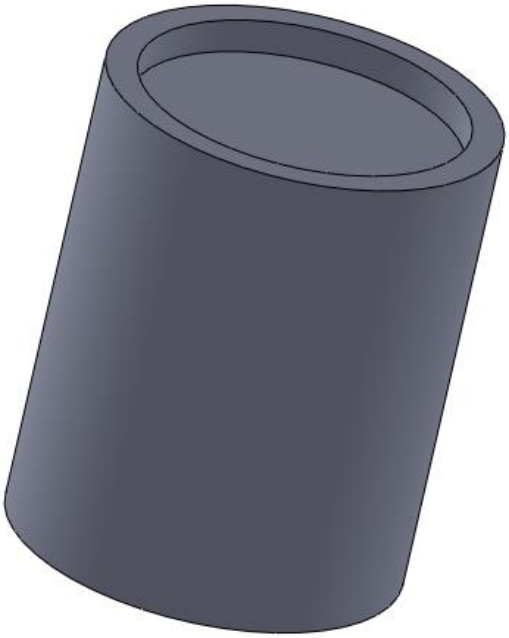
Slika 24. Model vijka u programu Solidworks



Slika 25. Model ploče u programu Solidworks



Slika 26. Model prsta hvataljke u programu Solidworks



Slika 27. Model hvataljke u programu Solidworks

10. PYTHON SKRIPTE

Kako bismo napravili simulaciju u programu RoboDK, potrebno je napraviti Python skripte. One su korisne iz nekoliko razloga:

- Automatizacija procesa

  Python skripte omogućuju automatizaciju ponavljajućih ili složenih zadataka u simulaciji. Umjesto ručnog izvođenja svake radnje u sučelju programa, može se napraviti skripta pomoću koja će to učiniti za nas.

- Prilagodljvost

  Python je svestrani programski jezik koji nudi mnoge biblioteke i alate. Koristeći Python, simulacija se može prilagoditi prema specifičnim potrebama korisnika.

- Razvoj složenih algoritama

  Za rješavanje složenih problema u simulaciji često su potrebni složeni algoritmi. Python omogućuje razvoj takvih algoritama na učinkovit i intuitivan način.

- Komunikacija s vanjskim sustavima

  Python skripte omogućuju komunikaciju s vanjskim sustavima, poput drugih softverskih alata, senzora ili uređaja. To je posebno korisno kada je potrebno integrirati simulaciju s drugim dijelovima proizvodnog procesa.

- Brži razvoj

  Python je jezik visoke razine koji omogućuje brz i efikasan razvoj aplikacija. Korištenjem Python skripti, možemo brzo iterirati i testirati različite ideje u simulaciji.

Ukratko, Python skripte su ključne za poboljšanje funkcionalnosti i prilagodljivosti simulacije u programu RoboDK, omogućujući korisnicima da učinkovito automatiziraju procese, prilagode simulaciju prema svojim potrebama i razvijaju složene algoritme.

Za zadatak koji je zadan u ovom diplomskom radu potrebne su tri Python skripte. Jedna koja generira nove vijke, jedna koja briše stare te skripta koja omogućuje otvaranje i zatvaranje hvataljke. Dijelovi tih skripti objašnjeni su u daljnjem tekstu.

10.1. New screw

```
# You can also use the new version of the API:
from robolink import *    # RoboDK API
from robodk import *      # Robot toolbox
RDK = robolink.Robolink()

# Program example:
item = RDK.Item('Screw_1')
item.Copy()
item_copy1 = RDK.Paste()
item_copy1.setName('Screw_2')
item_copy1.setPose(transl(-650,0,20)) # Coordinates relative to UR5 base
```

Slika 28. Dio Python skripte „New screw“

Najprije imamo uvoz svih funkcija i metoda iz modula robodk te robolink. Modul robodk pruža funkcionalnost za rad s objektima unutar RoboDK projekta, kao što su roboti, alati, ciljevi, putanje, itd, dok modul robolink pruža funkcionalnost za interakciju s RoboDK-om kao cjelinom, kao što su otvaranje, spremanje i manipulacija s projektima. Zatim se stvara instanca klase robolink koja predstavlja vezu s RoboDK-om te omogućuje pristup svim funkcijama i metodama dostupnim putem API-ja.

Sada se poziva metoda Item() na instanci RDK kako bi se dohvatila referenca na objekt s imenom 'Screw_1' pa se objekt dohvaćen u prethodnom koraku kopira. Zatim se kopirani objekt lijepi u RoboDK projektu, a rezultirajuća referenca na novi objekt pohranjuje se u varijablu item_copy1 te se novi objekt imenuje kao 'Screw_2'. Slijedi postavljanje položaja novog objekta koristeći funkciju setPose() čije su koordinate definirane s obzirom na koordinatni sustav baze UR5e robota. Konkretno, pozicija novog objekta translatirana je za 650mm u negativnom smjeru x osi i 20mm u pozitivnom smjeru z osi u odnosu na ishodište koordinatnog sustava robota.

10.2. Delete screws

```
from robodk import *      # RoboDK API
from robolink import *    # Robot toolbox
# Link to RoboDK
RDK = robolink.Robolink()

list_items = RDK.ItemList() # list all names
for item in list_items:
    if item.Name().endswith('_2'):
        item.Delete()
```

Slika 29. Dio Python skripte "Delete screws"

U ovoj Python skripti također se najprije napravi uvoz svih funkcija i metoda iz modula robodk te robolink, a zatim stvaranje instance klase robolink koja predstavlja vezu s RoboDK-om.

Sada se poziva metoda ItemList() na instanci RDK kako bi se dobio popis svih objekata u RoboDK projektu. Slijedi iteriranje kroz popis svih objekata spremljenih u listi list_items svaki objekt u listi list_items te se provjerava završaval li ime objekta sa _2. Ako je ime objekta završava sa _2, poziva se metoda Delete() nad tim objektom kako bi se objekt obrisao iz RoboDK projekta.

10.3. Gripper

```
import sys # allows getting the passed argument parameters
from robodk import *

if len(sys.argv) < 2:
    print('Invalid parameters. This function must be called as MoveGripper(Gripper_Id, Distance)')
    print('Number of arguments: ' + str(len(sys.argv)))
    #raise Exception('Invalid parameters provided: ' + str(sys.argv))
    entry = mbox('Move gripper. Type:\n1 to open, 0 to close\n\nNote: this can be called as a program.\nExample: Gripper(1)', entry='0')
    if not entry:
        raise Exception('Operation cancelled by user')

    GRIPPER_VALUE = int(entry)*4.25

else:
    GRIPPER_VALUE = int(sys.argv[1])*4.25
```

Slika 30. Dio Python skripte "Gripper" (1)

Uvozi se modul sys koji omogućuje dobivanje proslijeđenih argumenata, a zatim sve funkcije i metode iz modula robodk.

Sada slijedi provjera broja proslijeđenih argumenata. Ako je broj argumenata manji od 2, ispisuje se poruka o neispravnim parametrima, a ako broj argumenata nije manji od 2, postavlja se vrijednost hvataljke na temelju proslijeđenog argumenta.

```
from robolink import *    # API to communicate with RoboDK
RDK = Robolink()


finger_P = RDK.Item('FingerB1')
finger_N = RDK.Item('FingerA1')

if not finger_P.Valid() or not finger_N.Valid():
    raise Exception('Invalid Gripper ID')

finger_P.setPose(transl(+GRIPPER_VALUE,0,0))
finger_N.setPose(transl(-GRIPPER_VALUE,0,0))

# Set the station parameter
RDK.setParam('Gripper', GRIPPER_VALUE)
```

Slika 31. Dio Python skripte "Gripper" (2)

Ponovno imamo uvoz svih funkcija i metoda iz modula robolink te stvaranje instance klase Robolink za komunikaciju s RoboDK-om.

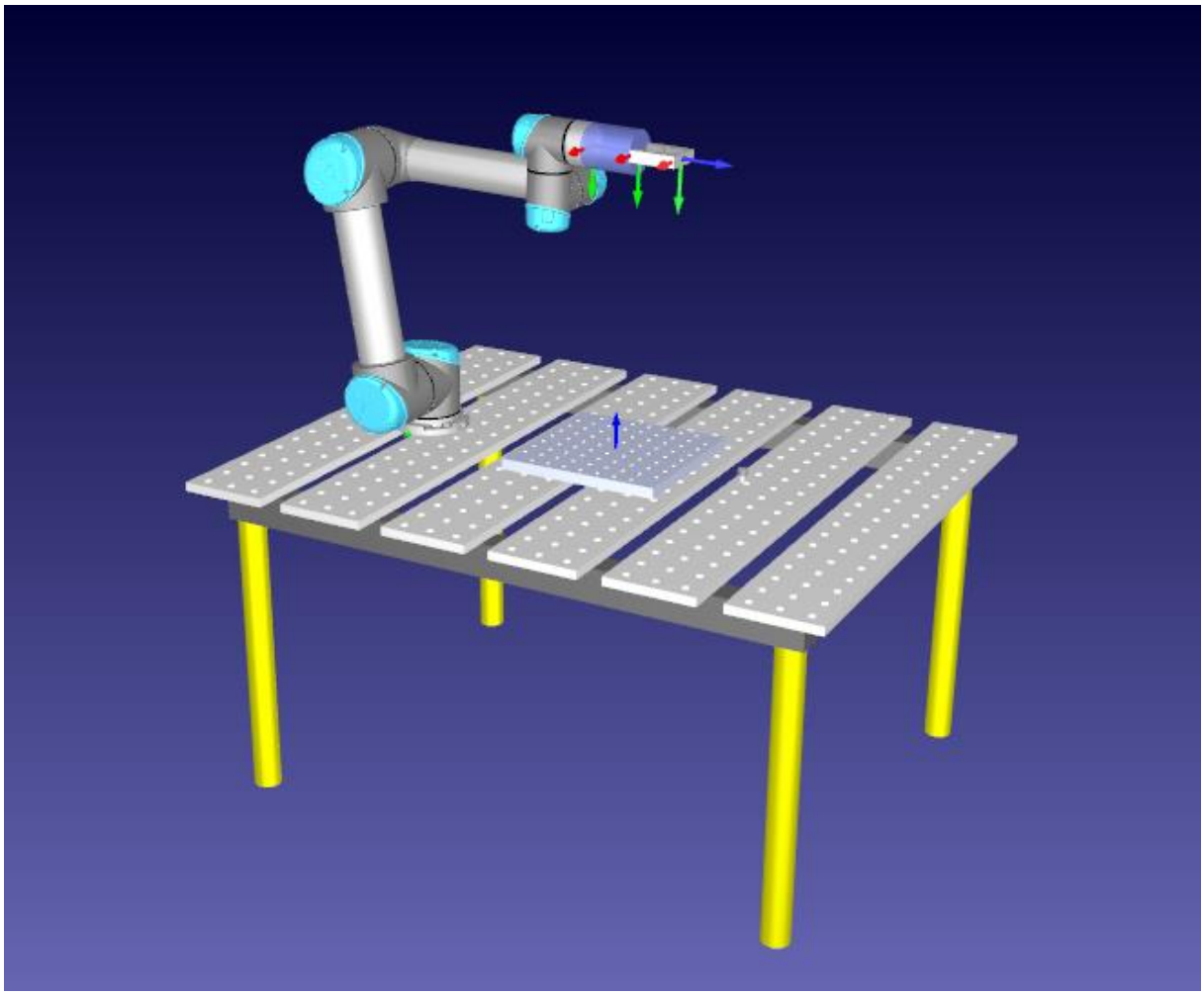
Sada se pomoću metode Item() dohvaćaju reference na prste hvataljke te se provjerava njihova valjanost.

Pomoću funkcije setPose() postavlja se položaj prstiju hvataljke. Najprije se postavlja položaj prvog prsta hvataljke pomaknutog za vrijednost GRIPPER_VALUE duž osi x, a zatim se postavlja položaj drugog prsta hvataljke pomaknutog za vrijednost GRIPPER_VALUE u negativnom smjeru osi x.

Zatim se postavlja parametar stanice Gripper na vrijednost GRIPPER_VALUE.

11. SIMULACIJA U RoboDK

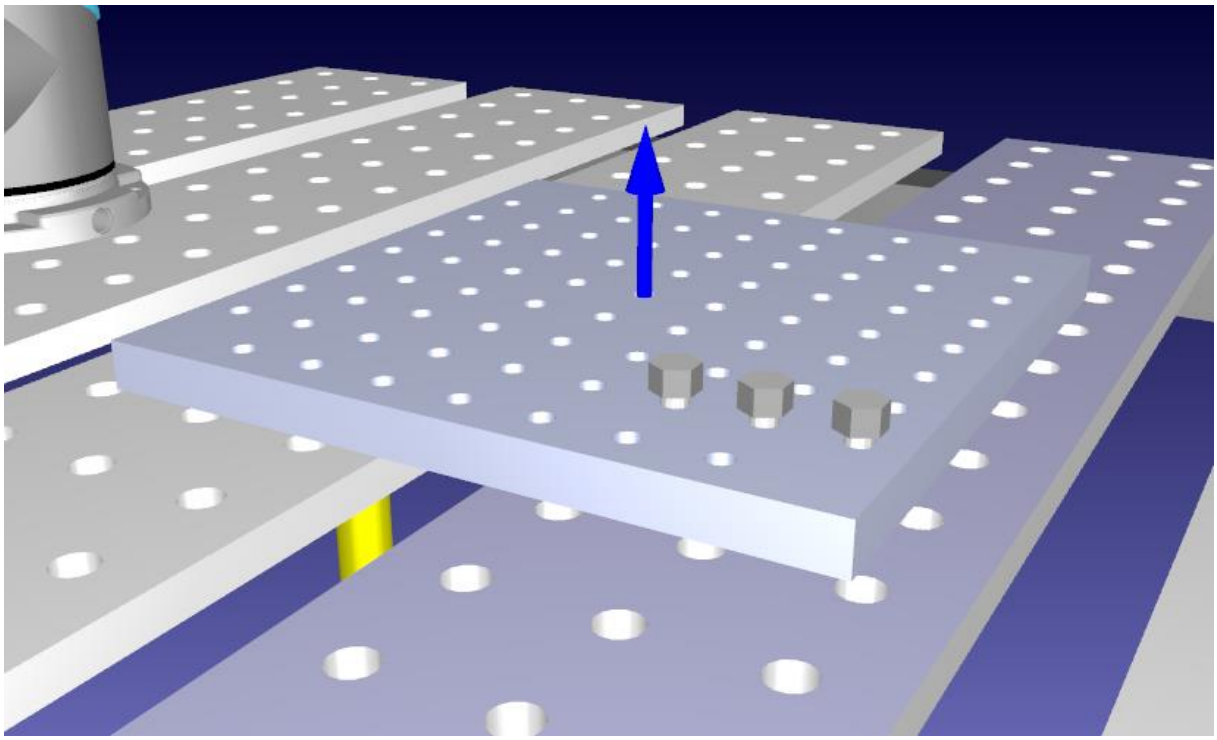
Simulaciju u programu RoboDK započinjemo učitavanjem robota UR5 koji se nalazi u knjižnici programa. U programu su, također, dostupni i različiti stolovi na kojima stoje robot i predmeti s kojima će robot raditi pa je izabran i učitan jedan od tih stolova. Nakon toga dodaju se hvataljka i prsti te ploča i vijak, a pritom je potrebno paziti na koordinatne sustave tih dijelova. Na Slici 32. vide se robot u početnoj poziciji i svi ostali dodani dijelovi.



Slika 32. Robot UR5, stol, hvataljka i prsti te ploča i vijak

Kada su svi dijelovi dodani, potrebno je označiti pozicije koje će robot trebati dosegnuti, ubaciti prethodno objašnjene Python skripte te napraviti programe koji opisuju gibanje robota.

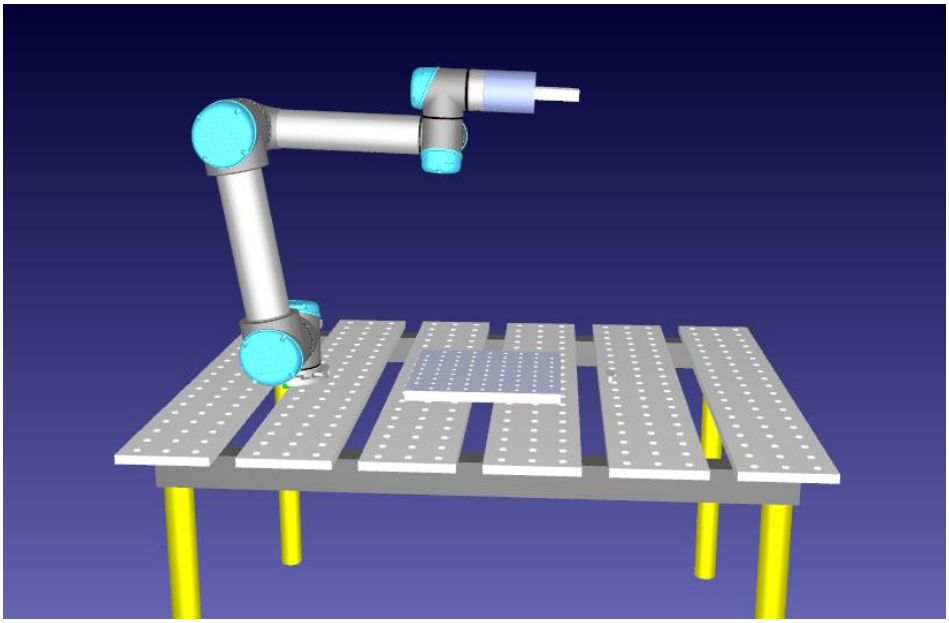
Kako je u ovom diplomskom radu korištena Pick and place metoda, na sljedećoj slici prikazani su vijci montirani na ploču koja se nalazi na stolu.



Slika 33. Prikaz vijaka montiranih na ploču

12. POVEZIVANJE S UR5 ROBOTOM

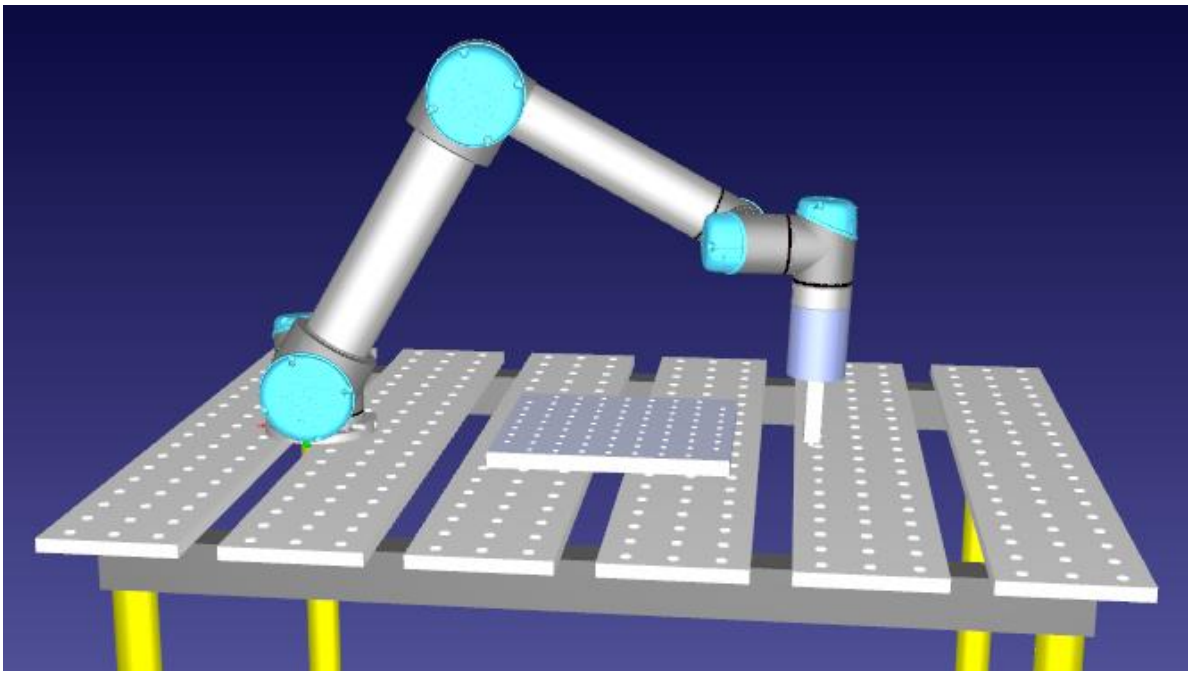
Nakon što je simulacija napravljena i testirana u programu RoboDK, potrebno je učitati simulaciju na stvarnog robota. Slike 34. i 35. prikazuju početni položaj robota.



Slika 34. Početni položaj u programu RoboDK

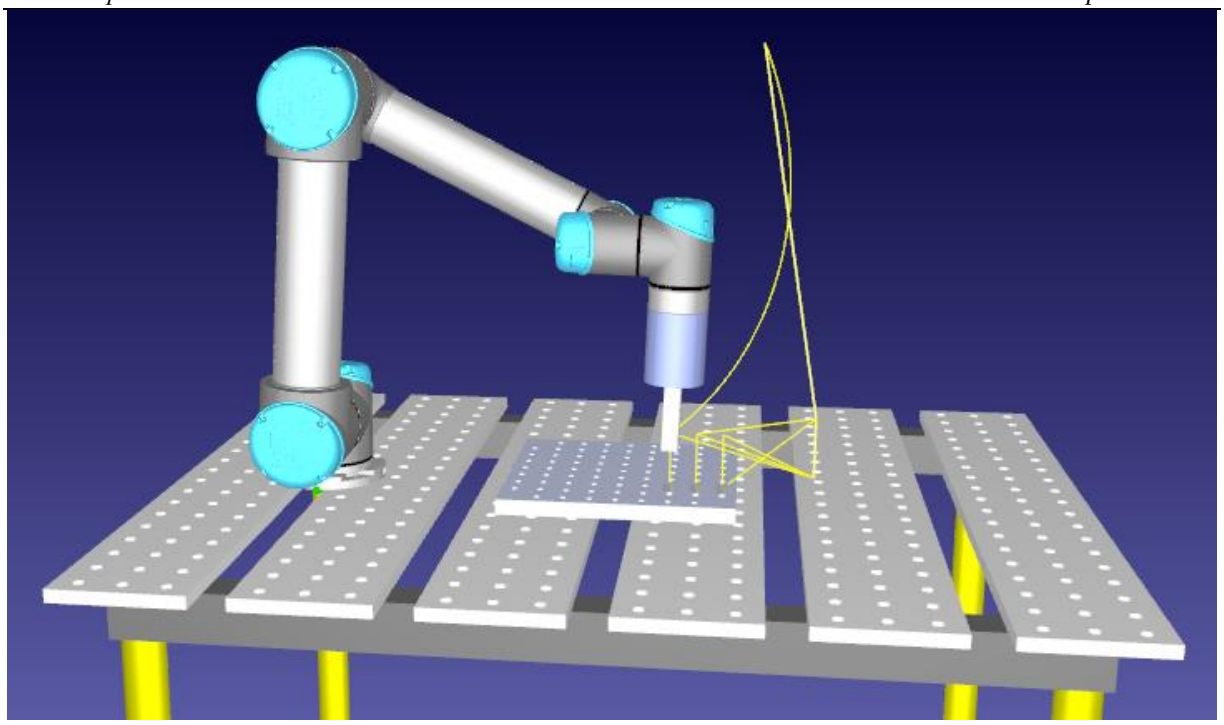
Slika 35. Početni položaj u stvarnosti

Na sljedećim slikama prikazano je kako robot hvata prvi vijak (Slike 36. i 37.) te položaj robota nakon stavljanja tri vijka u ploču (Slike 38. i 39.).



Slika 36. Robot hvata prvi vijak u programu RoboDK

Slika 37. Robot hvata prvi vijak u stvarnosti



Slika 38. Prikaz robota nakon što je ostavio treći vijak u programu RoboDK

Slika 39. Prikaz robota nakon što je ostavio treći vijak u stvarnosti

13. KRITIČKI OSVRT

S obzirom na to da izrada ovog diplomskog rada nije prošla bez određenih poteškoća, u ovom poglavlju bit će opisani problemi koji su se pojavili.

Kako bi se napravila simulacija robotske montaže u RoboDK-u, bilo je potrebno najprije izmjeriti predmete rada te hvataljku i prste, a potom ih modelirati u programu Solidworks. Zatim je pomoću tih dijelova napravljena simulacija u relativno kratkom vremenu i bez većih problema.

Nakon toga je bilo potrebno učitati simulaciju na stvarnog robota i tada je došlo do nekoliko poteškoća koje su bile ispravljane „u hodu“. Najprije je uočeno kako su pozicija i koordinatni sustav stvarnog robota drugačije postavljeni u odnosu na simulaciju, to jest, u simulaciji je bilo potrebno okrenuti koordinatni sustav za 180^0. Kada je koordinatni sustav postavljen ispravno, sve pozicije robota su se pomaknule pa ih je bilo potrebno ispočetka odrediti. Nakon toga, nastao je problem s udaljenosti do koje je robot trebao doći prema simulaciji i s prevelikim kutovima zakreta u zglobovima robota. Pri izradi simulacije nije se previše obraćala pažnja na te parametre, ali kod spajanja sa stvarnim robotom, postoje određene udaljenosti te kutovi zakreta koje robot ne može doseći pa zbog toga staje s radom. Kada su se promijenili ti parametri, bilo je potrebno ponovno postaviti nove pozicije robota. Otklanjanjem ovih poteškoća, robot je bez problema odradio zadatak.

Kod izrade simulacije smatrano je kako je najvažnije što vjernije modelirati dijelove i osigurati da se alat i postolje u simulaciji podudaraju sa stvarnom situacijom. Kako je u simulaciji sve radilo kako treba, nisu bili očekivani nikakvi problemi pri učitavanju simulacije na robota. Međutim, ranije navedeni problemi koji su se pojavili prilikom spajanja simulacije na stvarnog robota dovode do zaključka kako je potrebno puno više pažnje usmjeriti na kalibraciju samog robota te pažljivo određivanje pozicije i orijentacije.

14. ZAKLJUČAK

Programiranje robota u programu RoboDK pruža korisnicima intuitivan i efikasan način za simuliranje, programiranje i optimizaciju robota za različite primjene u industriji. Korištenje ovog programa omogućuje virtualno testiranje i optimizaciju simulacije prije implementacije na stvarnom robotu što rezultira uštedom vremena i resursa.

Jedna od najvažnijih prednosti korištenja programa RoboDK je mogućnost spajanja simulacije na stvarnog robota. Tako je korisnicima omogućeno prenošenje razvijenih programa bez problema s virtualnog okruženja na stvarni robot, osiguravajući dosljednost i pouzdanost u radu. Integracija simulacije sa stvarnim robotom pruža korisnicima mogućnost bržeg razvoja i testiranja programa, smanjujući potrebu za eksperimentalnim pristupima i minimizirajući rizik od grešaka.

U ovom diplomskom radu napravljena je simulacija automatske robotske montaže za robot UR5 u programu RoboDK metodom offline programiranja te je nakon toga povezana sa stvarnim robotom. Tako su pokazane ranije spomenute prednosti programiranja u RoboDK-u, kao što je mogućnost programiranja robota bez potrebe da se korisnik fizički nalazi kraj robota, a također, simulacija rada robota u virtualnom okruženju omogućuje ispravljanje mogućih grešaka bez oštećenja opreme i zastoja u proizvodnji.

LITERATURA

[1] Why use Offline Robot Programming Software and how to get started – Robotmaster, http://www.robotmaster.com/en/newsroom/offline-robot-programming-software, pristupljeno: 4.3.2024.

[2] Automating with an aseembly robot – Reeco, https://www.reeco.co.uk/automation-robotics/assembly-robot/, pristupljeno: 5.3.2024.

[3] A History of Robot Programming Languages – Robotiq, https://blog.robotiq.com/the-history-of-robot-programming-languages, pristupljeno: 5.3.2024.

[4] Unimate – Robots, https://robotsguide.com/robots/unimate, pristupljeno: 5.3.2024.

[5] Koordinatni sustav – Wikipedija, https://hr.wikipedia.org/wiki/Koordinatni_sustav, pristupljeno: 5.3.2024.

[6] Z. Kovačić, S. Bogdan, V. Krajči, Osnove robotike, Zagreb, 2002.

[7] Robot Grippers - Robots.com, https://www.robots.com/articles/grippers-for-robots, pristupljeno: 4.3.2024.

[8] OnRobot, Vakuumska hvataljka VGP20 – Memidos, https://www.memidos.com/hr/proizvod/onrobot-vakuumska-hvataljka-vgp20/, pristupljeno: 4.3.2024.

[9] OnRobot, 2FG7 Hvataljka za prste – Memidos, https://www.memidos.com/hr/proizvod/onrobot-2fg7-hvataljka-za-prste-2/, pristupljeno: 4.3.2024.

[10] Electric Grippers – Robotics tomorrow, https://www.roboticstomorrow.com/article/2018/04/electric-grippers/11628/, pristupljeno: 4.3.2024.

[11] Robotika - Kobot vs Robot – Mreža, https://mreza.bug.hr/robotika/robotika-kobot-vs-robot-32282, pristupljeno: 6.3.2024.

[12] UR5 – RoboDK, https://robodk.com/robot/Universal-Robots/UR5, pristupljeno: 6.3.2024.

[13] How are Robots Programmed? – DevOpsSchool, https://www.devopsschool.com/blog/how-are-robots-programmed/, pristupljeno: 10.3.2024.

[14] RoboDK Add-In for SolidWorks – RoboDK, https://robodk.com/doc/en/Plugin-SolidWorks.html, pristupljeno: 7.3.2024.

[15] RoboDK API – RoboDK, https://robodk.com/doc/en/RoboDK-API.html#RoboDKAPI, pristupljeno: 7.3.2024.

[16] Python API - RoboDK, https://robodk.com/doc/en/RoboDK-API-Python-API.html, pristupljeno: 7.3.2024.

[17] L. Kukurin, Koncept povezivanja virtualnog modela robota s robotom u sklopu stvarne okoline, diplomski rad, Zagreb, 2022.

[18] T. Pušnik, Montaža z industrijskim robotom UR5 v simulacijskem okolju RoboDK, Maribor, 2016.

PRILOZI

I. Programski kod za generiranje novih vijaka

II. Programski kod za brisanje vijaka

III. Programski kod za otvaranje i zatvaranje hvataljke

I. Programski kod za generiranje novih vijaka

```
# Type help("robodk.robolink") or help("robodk.robomath") for more information
# Press F5 to run the script
# Documentation: https://robodk.com/doc/en/RoboDK-API.html
# Reference:     https://robodk.com/doc/en/PythonAPI/robodk.html
# Note: It is not required to keep a copy of this file, your Python script is saved with your RDK project

# You can also use the new version of the API:
from robolink import *    # RoboDK API
from robodk import *      # Robot toolbox
RDK = robolink.Robolink()

# Program example:
item = RDK.Item('Screw_1')
item.Copy()
item_copy1 = RDK.Paste()
item_copy1.setName('Screw_2')
item_copy1.setPose(transl(-650,0,20)) # Coordinates relative to UR5 base
```

II. Programski kod za brisanje vijaka

```
# Type help("robodk.robolink") or help("robodk.robomath") for more information
# Press F5 to run the script
# Documentation: https://robodk.com/doc/en/RoboDK-API.html
# Reference:     https://robodk.com/doc/en/PythonAPI/robodk.html
# Note: It is not required to keep a copy of this file, your Python script is saved with your RDK project

# Forward and backwards compatible use of the RoboDK API:
# Remove these 2 lines to follow python programming guidelines
from robodk import *      # RoboDK API
from robolink import *    # Robot toolbox
# Link to RoboDK
RDK = robolink.Robolink()

list_items = RDK.ItemList() # list all names
for item in list_items:
    if item.Name().endswith('_2'):
        item.Delete()
```

III. Programski kod za otvaranje i zatvaranje hvataljke

```
# This program allows moving gripper fingers
import sys # allows getting the passed argument parameters
from robodk import *

if len(sys.argv) < 2:
    print('Invalid parameters. This function must be called as MoveGripper(Gripper_Id, Distance)')
    print('Number of arguments: ' + str(len(sys.argv)))
    #raise Exception('Invalid parameters provided: ' + str(sys.argv))
    entry = mbox('Move gripper. Type:\n1 to open, 0 to close\n\nNote: this can be called as a program.\nExample: Gripper(1)', entry='0')
    if not entry:
        raise Exception('Operation cancelled by user')

    GRIPPER_VALUE = int(entry)*4.25

else:
    GRIPPER_VALUE = int(sys.argv[1])*4.25

from robolink import *    # API to communicate with RoboDK
RDK = Robolink()


finger_P = RDK.Item('FingerB1')
finger_N = RDK.Item('FingerA1')

if not finger_P.Valid() or not finger_N.Valid():
    raise Exception('Invalid Gripper ID')

finger_P.setPose(transl(+GRIPPER_VALUE,0,0))
finger_N.setPose(transl(-GRIPPER_VALUE,0,0))

# Set the station parameter
RDK.setParam('Gripper', GRIPPER_VALUE)
```