SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Diplomski sveučilišni studij Računarstvo

Planiranje putanje alata robotskog manipulatora za obradu površine objekta na temelju 3D snimke

Diplomski rad

Adrian Čičić

Osijek, 2019.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 20.09.2019.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Adrian Čičić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-924R, 26.09.2018.
OIB studenta:	25648661405
Mentor:	Prof.dr.sc. Robert Cupec
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Damir Filko
Član Povjerenstva:	Doc.dr.sc. Emmanuel-Karlo Nyarko
Naslov diplomskog rada:	Planiranje putanje alata robotskog manipulatora za obradu površine objekta na temelju 3D snimke
Znanstvena grana rada:	**Automatizacija i robotika (zn. polje elektrotehnika)**
Zadatak diplomskog rada:	Izraditi računalni program koji na temelju informacije o prisustvu objekta od interesa na sceni te njegovom položaju dobivene iz slike snimljene 3D kamerom planira putanju alata robotskog manipulatora, koja omogućuje gibanje vrha alata po površini objekta. Tema rezervirana za studenta Adrijana Čičića.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	20.09.2019.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 03.10.2019.

Ime i prezime studenta:	Adrian Čičić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-924R, 26.09.2018.
Ephorus podudaranje [%]:	15

Ovom izjavom izjavljujem da je rad pod nazivom**: Planiranje putanje alata robotskog manipulatora za obradu površine objekta na temelju 3D snimke**

izrađen pod vodstvom mentora Prof.dr.sc. Robert Cupec

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD

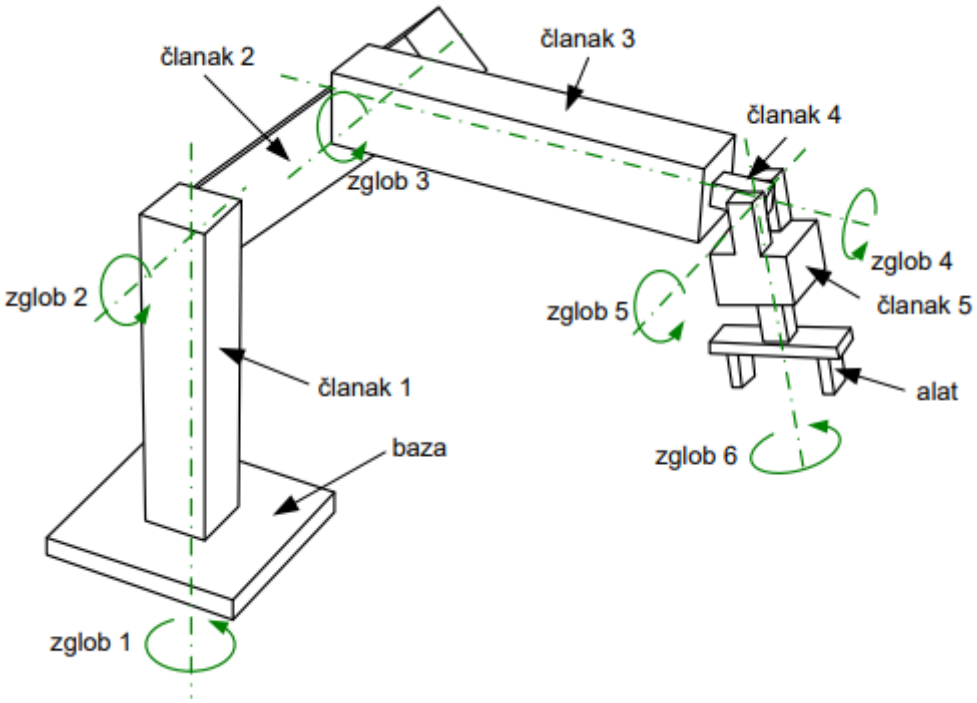
Robotski manipulator je posebna vrsta robota koja po svojoj funkciji odgovara funkcionalnostima ljudske ruke. Svrha robotskih manipulatora je manipuliranje predmetima u njihovoj okolini pomoću odgovarajućeg alata. Robotski manipulatori imaju široku primjenu u automatiziranim postrojenjima zbog svoje velike brzine i preciznosti pri izvođenju repetitivnih radnji. Kako bi se robotski manipulator mogao kretati kroz prostor potrebno je definirati putanju po kojoj želimo da se alata robotskog manipulatora kreće. Pri tome je potrebno definirati pomake svakog pojedinog zgloba manipulatora kako bi se izvelo željeno kretanje alata u prostoru. Ukoliko se robotski manipulator nalazi u dinamičkoj okolini, gdje se objekti od interesa ne nalaze uvijek na istom mjestu, potrebno je uvesti neki način detektiranja lokacije i orijentacije objekta od interesa, te na temelju tih informacija isplanirati putanju alata manipulatora. U svrhu izrade ovog rada razvijen je program unutar ROS (Robot Operating System) okvira koji pomoću informacija dobivenih iz 3D slike planira putanju vrha alata po objektu od interesa. Teorijski dio robotskih manipulatora, planiranja putanje alata manipulatora u prostoru i kalibracija sustava robot-kamera obrađen je u drugom poglavlju. U trećem poglavlju prikazane su neke od funkcionalnosti ROS programskog okvira, te je pobliže opisan MoveIt! programski paket. U četvrtom poglavlju detaljno se opisuju funkcionalnosti razvijenih programa te se u petom poglavlju testira preciznost razvijenog sustava.

1.1. Zadatak diplomskog rada

Izraditi računalni program koji na temelju informacije o prisustvu objekta od interesa na sceni te njegovom položaju dobivene iz slike snimljene 3D kamerom planira putanju alata robotskog manipulatora, koja omogućuje gibanje vrha alata po površini objekta.

2. ROBOTSKI MANIPULATOR

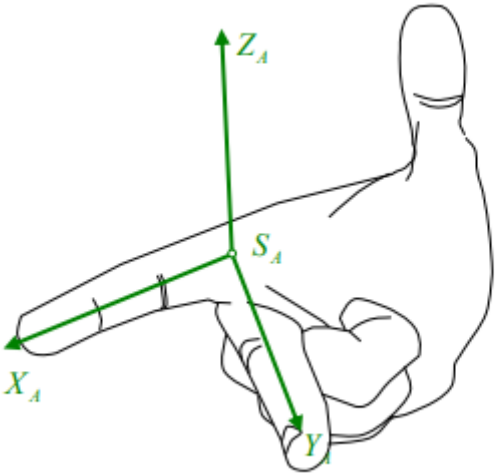
Robot je mehanički stroj koji omogućuje gibanje određenog alata ili predmeta u prostoru različitim putanjama pomoću odgovarajućih senzora. Robot se sastoji od više krutih tijela, koje zovemo člancima, koji su međusobno povezani pokretljivim zglobovima (slika 2.1.). Svrha robota može biti manipulacija fizičkim objektima, obrada površine, bušenje, zavarivanje i sl. Roboti se danas široko primjenjuju u industriji zbog svoje velike preciznosti i pozitivnog utjecaja na smanjenje troškova proizvodnje. Kako bi robot bio u mogućnosti odraditi zadani posao, potrebno je prvo postaviti odgovarajući alat na robota. Danas se u praktičnoj primjeni najčešće koriste roboti koji po funkciji odgovaraju ljudskoj ruci, te se oni nazivaju robotski manipulatori.



Slika 2.1. Robotski manipulator [1]

Kada se govori o robotskom manipulatoru najčešće se govori o robotskom manipulatoru se 6 stupnjeva slobode izvedbe kao manipulator prikazan na slici 2.1. Pomicanje članaka manipulatora izvedeno je pomoću električnih motora koji pokreću zglobove manipulatora. Kretanje manipulatora se definira upravljanjem svih motora u pojedinim zglobovima manipulatora.

Svrha robotskog manipulatora je pozicioniranje alata u prostoru što znači da je matematički opis položaja predmeta u prostoru osnova za matematičko modeliranje robota. Svaki predmet u trodimenzionalnom prostoru se može opisati kao skup točaka u navedenom prostoru odnosno svaka točka predmeta se može opisati pomoću 3 broja koji označavaju pomake u svakoj pojedinoj osi desnog koordinatnog sustava(slika 2.2.).



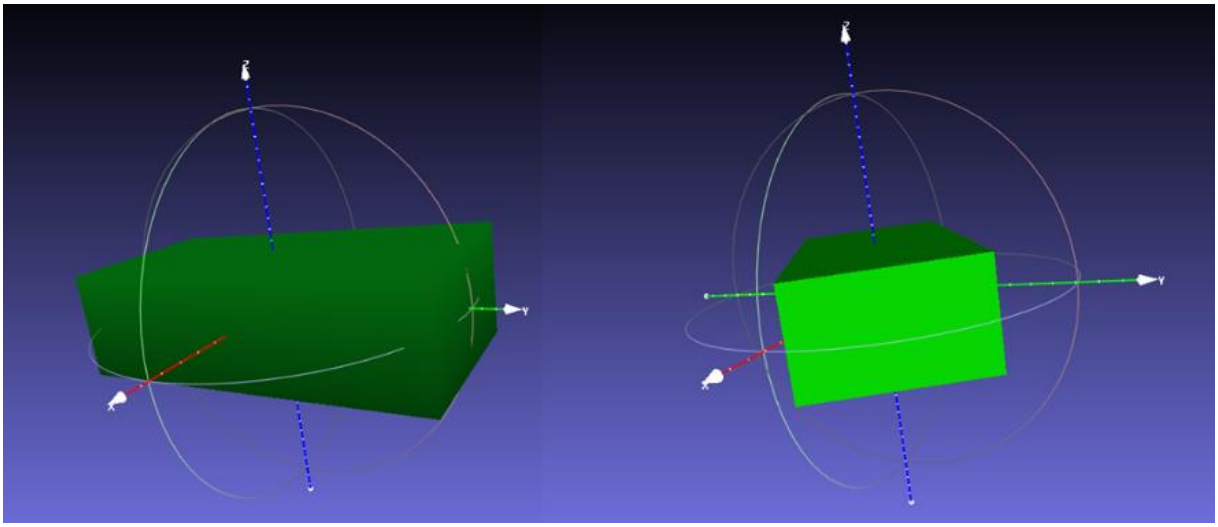
Slika 2.2. Desni koordinatni sustav

Određivanjem 3 točke na promatranom predmetu možemo odrediti gdje se nalaze sve ostale točke ukoliko znamo kako izgleda promatrani predmet u potpunosti. Jednostavniji način za određivanje pozicije i orijentacije predmeta u prostoru je pomoću transformacijskih matrica. Transformacijska matrica je 4x4 matrica koja sadrži informacije gdje se jedan koordinatni sustav nalazi u odnosu na drugi koordinatni sustav(2-1).

$$T = \begin{bmatrix} & R & & t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-1}$$

T je transformacijska matrica 4x4 čije su komponente matrica R koja je dimenzija 3x3 i translacijska matrica t koja je dimenzija 3x1.

Rotacijska matrica govori kako su pojedine osi koordinatnog sustava rotirane u odnosu na promatrani koordinatni sustav(slika 2.3).

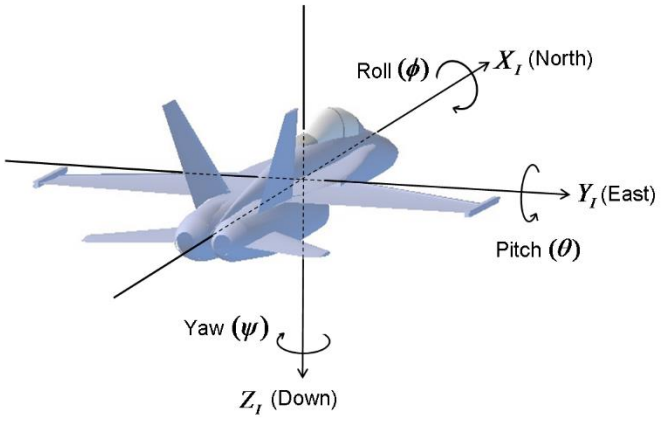


Slika 2.3. Rotacija predmeta oko z osi za 30 stupnjeva

Izrazom (2-2) je prikazano kako se rotacije oko pojedine osi računaju i prikazuju pomoću rotacijske matrice.

$$R = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2-2}$$

Kutovi ψ,θ,ϕ su rotacije oko pojedinih osi koordinatnog sustava. Ovi kutovi se zovu Eulerovi kutovi (slika 2.4).



Slika 2.4. Prikaz Eulerovih kutova [2]

Za predmet prikazan na slici 2.3. rotacijska matrica iz stanja prikazanog na lijevoj strani slike u stanje na desnoj strani bi izgledala kao izraz (2-3).

$$R = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos 30 & -\sin 30 & 0 \\ \sin 30 & \cos 30 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2\text{-}3)$$

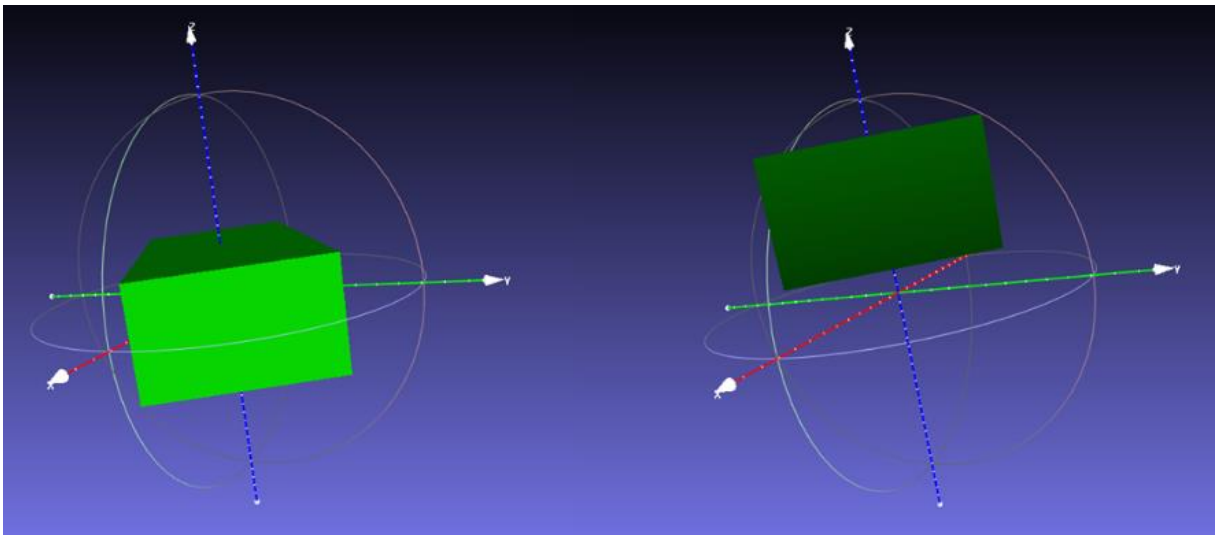
Uz rotacijsku matricu i Eulerove kutove rotaciju predmeta moguće je opisati pomoću kvaterniona. Kvaternioni predstavljaju proširenje kompleksnih brojeva i imaju oblik :

$$q = q_r + q_i \cdot i + q_j \cdot j + q_k \cdot k \quad (2\text{-}4)$$

Računanje transformacijske matrice pomoću kvaterniona je opisano izrazom (2-5).

$$R = \begin{bmatrix} 1 - 2(q_j^2 + q_k^2) & 2(q_i q_j - q_k q_r) & 2(q_i q_k + q_j q_r) \\ 2(q_i q_j + q_k q_r) & 1 - 2(q_i^2 + q_k^2) & 2(q_j q_k - q_i q_r) \\ 2(q_i q_k - q_j q_r) & 2(q_j q_k + q_i q_r) & 1 - 2(q_i^2 + q_j^2) \end{bmatrix} \quad (2\text{-}5)$$

Drugi dio transformacijske matrice je translacijska matrica koja ima 3 komponente koje označavaju pomake u x, y i z osima promatranog koordinatnog sustava. Translacija predmeta po z osi je prikazana na slici 2.5.



Slika 2.5. Translacijski pomak predmeta za 5 cm u z osi

Translacijska matrica za predmet prikazan na slici 2.5. bi imala oblik:

$$\mathrm{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0.05 \end{bmatrix} \tag{2-6}$$

Izraz transformacijske matrice za predmet koji je prvo rotiran kao na slici 2.3., a zatim translacijski pomaknut kao na slici 2.5. bi izgledao ovako:

$$T = \begin{bmatrix} 0.866 & -0.5 & 0 & 0 \\ 0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 1 & 0.05 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-7}$$

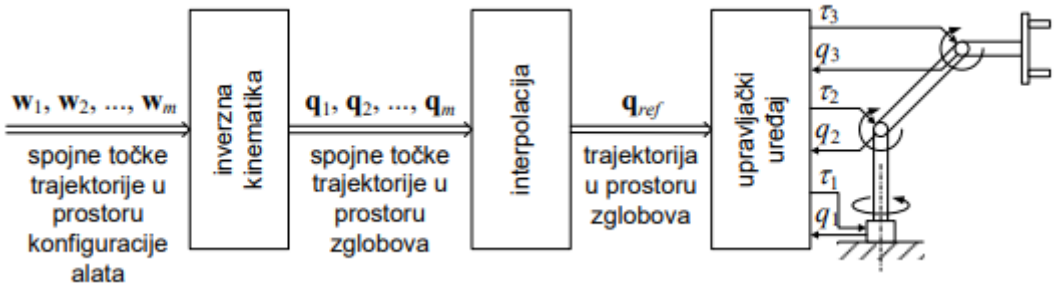
Pomoću ove transformacijske matrice možemo odrediti gdje se bilo koja točka na predmetu nalazi nakon pomicanja predmeta na novu poziciju i orijentaciju. Pozicija i orijentacija vrha alata manipulatora također se može opisati pomoću transformacijske matrice, a vrijednosti koje su upisane u matricu se određuju pomoću direktne kinematike. Produkt direktne kinematike je transformacijska matrica transformacije iz baznog koordinatnog sustava u koordinatni sustav zadnjeg zgloba manipulatora, najčešće 6. zgloba. Ovaj odnos je važan jer ukoliko se zna gdje se nalazi zadnji zglob manipulatora onda se može odrediti i pozicija i orijentacija alata koji je pričvršćen za njega. Transformacijska matrica iz baznog koordinatnog sustava se računa na način da se izračunaju transformacijske matrice transformacije iz i-tog zgloba u i+1 zglob te se sve matrice pomnože kao u izrazu (2-8) gdje je dan primjer za 6 osni manipulator.

$${}^{0}_{6}T = {}^{0}_{1}T \cdot {}^{1}_{2}T \cdot {}^{2}_{3}T \cdot {}^{3}_{4}T \cdot {}^{4}_{5}T \cdot {}^{5}_{6}T \tag{2-8}$$

${}^{0}_{6}T$ je transformacijska matrica iz baznog koordinatnog sustava u koordinatni sustav 6. zgloba manipulatora, ${}^{0}_{1}T$ je transformacijska matrica iz baznog koordinatnog sustava u koordinatni sustav 1. zgloba manipulatora,a sve ostale transformacijske matrice označavaju transformacije iz koordinatnog sustava jednog zgloba u koordinatni sustav sljedećeg zgloba. Pomoću direktne kinematike se može odrediti gdje će se alat manipulatora nalaziti u prostoru za promatrane kutove zglobova, a inverzna kinematika govori koji će kutovi zglobova rezultirati zadanom pozicijom i orijentacijom alata. Ne postoji jedan način rješavanja inverzne kinematike za sve izvedbe robotskog manipulatora.

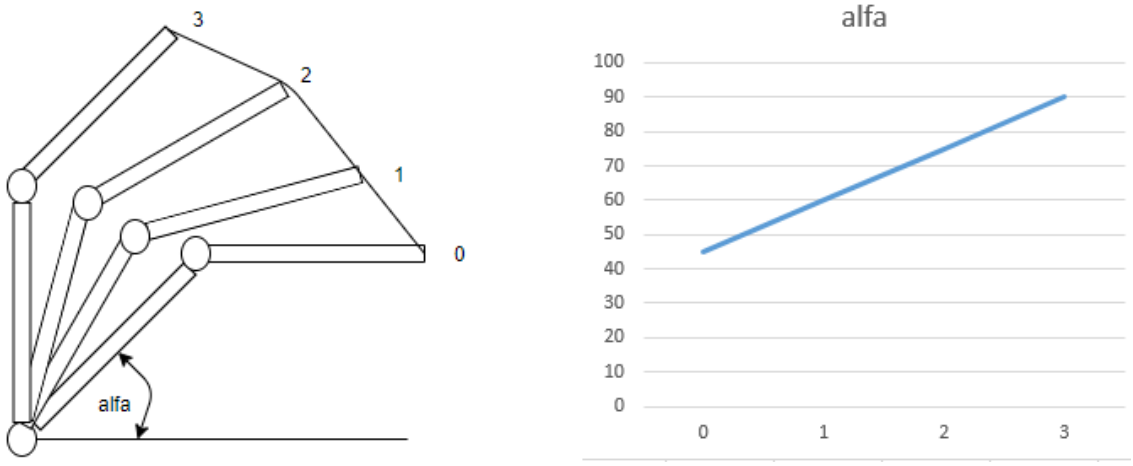
2.1. Planiranje putanje vrha alata manipulatora

Pri određivanju kretanja manipulatora kroz prostor potrebno je promatrati kako se manipulator kreće između dvije zadane točke, a ne samo poziciju i orijentaciju alata u početnoj i krajnjoj točki putanje. Računalno najjednostavniji način određivanje putanje po kojoj će se manipulator kretati je određivanje putanje u prostoru zglobova. Određivanje putanje u prostoru zglobova se izvršava tako što se uz poznate kutove zglobova izračuna inverzna kinematika krajnje pozicije. Svaki pojedini zglob se kreće od početne vrijednosti do vrijednosti izračunate pomoću inverzne kinematike. Ukoliko se vrh alata treba dovesti na više pozicija tokom putanje, onda se računa inverzna kinematika za svaku pojedinu točku te se vrijednosti kutova zglobova, mijenjaju iz vrijednosti i-te točke u vrijednost i+1 točke(slika 2.6.).



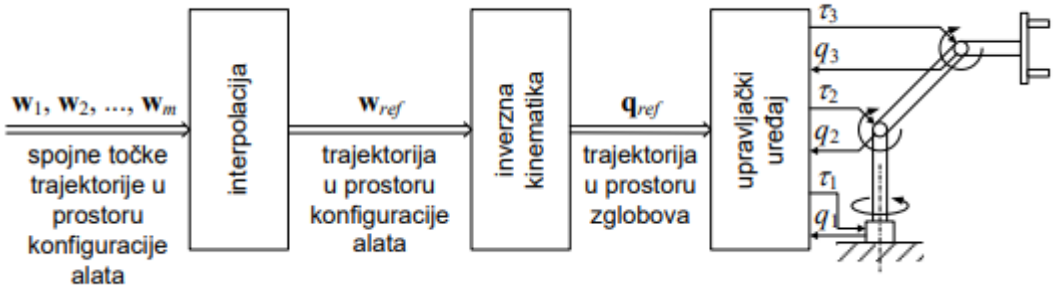
Slika 2.6. Planiranje putanje u prostoru zglobova [3]

Ovaj način je vrlo jednostavan za izračun, ali mana ovog pristupa je to što tokom izvršavanja zadane putanje, pozicija i orijentacija alata nije definirana u svakoj točki putanje što je potrebno u raznim funkcijama manipulatora, kao što je zavarivanje i obrada materijala. Linearna promjena kutova zglobova ne rezultira linearnim kretanjem vrha alata u prostoru(slika 2.7.).



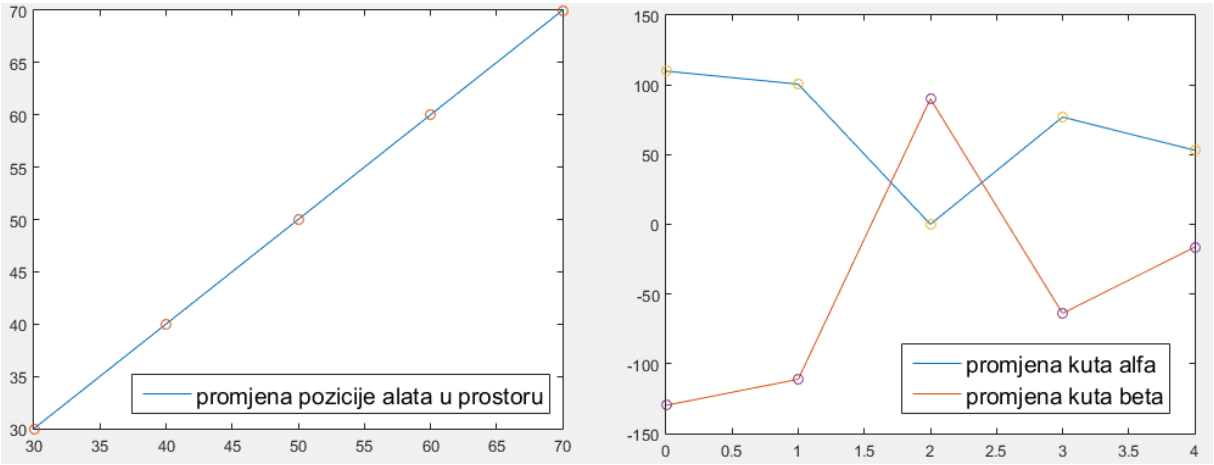
Slika 2.7. Prikaz putanje vrha alata uz linearnu promjenu vrijednosti jednog kuta zgloba

Ukoliko je potrebno vrh alata manipulatora kretati po definiranoj krivulji u prostoru, koristi se planiranje putanje u prostoru konfiguracije alata (slika 2.8.).



Slika 2.8. Planiranje putanje u prostoru konfiguracije alata [3]

Planiranje putanje u prostoru konfiguracije alata radi na način da se definirana putanja uzorkuje te se računa inverzna kinematika za svaki pojedini uzorak. Ovaj način osigurava kontroliranje pozicije i orijentacije vrha alata tijekom izvođenja putanje uz definiranu toleranciju, ali mana ovog tipa planiranja je problem rješavanja inverzne kinematike za svaki uzorak što je računalno zahtjevno. Na slici 2.9. je prikazan primjer planiranja putanje vrha alata u prostoru konfiguracije alata za manipulator sa dva stupnja slobode.



Slika 2.9. Promjena vrijednost kutova zglobova tokom planiranja putanje u prostoru konfiguracije alata

Putanja prikazana na slici 2.9. je uzorkovana 5 puta što znači da se inverzna kinematika računa 5 puta prije izvođenja kretnje manipulatora. Na grafu koji se nalazi na lijevoj strani slike je prikazana promjena pozicije alata. Točna pozicija alata je poznata jedino u uzorkovanim pozicijama, a na pozicijama između njih se manipulator neće kretati potpuno linearno što znači da nije moguće isplanirati potpunu linearnu kretnju bez odstupanja od zadane putanje. Ukoliko

je potrebno da manipulator što vjernije prati definiranu putanju onda je potrebno povećati uzorkovanje definirane putanje.

2.2. Kalibracija sustava robot-kamera

Vizualni servoing je metoda upravljanja manipulatorom na temelju informacija dobivenih iz percepcijskog senzora, u slučaju ovog diplomskog rada RGB-D kamere. Postoje dvije glavne vrste izvedbe vizualnog servoinga u robotici. Prva vrsta je izvedena na način da se kamera postavi na statičnu poziciju u okolini robota i to na način da je vidno polje kamere usmjereno u radni prostor manipulatora. Druga vrsta je izvedena na način da se kamera postavi na manipulator, uglavnom na alat manipulatora(slika 2.10.).

Slika 2.10. Pozicije kamere za vizualni servoing

Na lijevoj strani slike 2.10. je prikazana izvedba sa kamerom postavljenom na statičnoj poziciji, a na desnoj strani slike je prikazana kamera postavljena na alat manipulatora. U ovom diplomskom radu obrađivati će se problematika druge vrste odnosno kamera je postavljena na alat manipulatora. Kako bi ovaj način upravljanja manipulatorom bio moguć, potrebno je odrediti transformacijsku matricu (2-9), koja transformira vrijednosti koordinata iz koordinatnog sustava kamere u koordinatni sustav 6. zgloba manipulatora.

$$ {}^{C}_{6}T = \begin{bmatrix} & {}^{C}_{6}R & & {}^{C}_{6}t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-9} $$

${}^{C}_{6}T$ je transformacijska matrica iz koordinatnog sustava kamere u koordinatni sustav 6.zgloba, ${}^{C}_{6}R$ je rotacijska matrica, a ${}^{C}_{6}t$ je translacijska matrica iz koordinatnog sustava kamere u koordinatni sustav 6.zgloba.

Pomoću transformacijske matrice ${}^{C}_{6}T$ je moguće odrediti gdje se neki predmet detektiran od strane kamere nalazi u odnosu na koordinatni sustav 6. zgloba manipulatora pomoću izraza (2-10).

$$ {}^{P}_{6}T = {}^{C}_{6}T \cdot {}^{P}_{C}T \tag{2-10} $$

${}^{P}_{6}T$ je transformacijska matrica iz koordinatnog sustava predmeta u koordinatni sustav 6. zgloba manipulatora, a ${}^{P}_{C}T$ je transformacijska matrica iz koordinatnog sustava predmeta u koordinatni sustav kamere. Pozicija i orijentacija alata robotskog manipulatora se zadaju u baznom koordinatnom sustavu manipulatora zato je potrebno izraz (2-10) proširiti pomoću transformacijske matrice iz koordinatnog sustava 6. zgloba manipulatora u bazni koordinatni sustav manipulatora (izraz 2-11).

$$ {}^{P}_{0}T = {}^{6}_{0}T \cdot {}^{C}_{6}T \cdot {}^{P}_{C}T \tag{2-11} $$

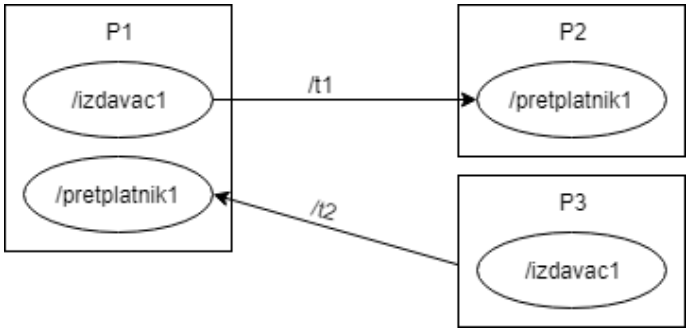
${}^{P}_{0}T$ je transformacijska matrica iz koordinatnog sustava predmeta u bazni koordinatni sustav manipulatora, a ${}^{6}_{0}T$ je transformacijska matrica iz koordinatnog sustava 6. zgloba manipulatora u bazni koordinatni sustav manipulatora. Postupak određivanja transformacijske matrice ${}^{C}_{6}T$ se zove kalibracija. Kalibracija se najčešće provodi pomoću kalibracijskog objekta poznatog oblika. Objekt se snima kamerom iz više pogleda, pri čemu se za svaki pogled zapisuju položaj objekta u odnosu na kameru te položaj 6. zgloba robota dobiven direktnom kinematikom. Na temelju tih podataka se računa transformacijska matrica. Zbog zašumljenosti podataka i rezolucije kamere točnija kalibracija se postiže kada se upotrijebi više mjerenja.

3. ROS – Robot Operating System

ROS (engl. *Robot Operating System*) je programski okvir otvorenog koda namijenjen razvoju robotske programske podrške. Pruža servise za apstrakciju sklopovlja, upravljanje sklopovljem na niskoj programskoj razini, komunikaciju između procesa te upravljanje paketima [4]. ROS je namijenjen korištenju na Linux operativnom sustavu mada postoje eksperimentalne verzije ROS programskog okvira za operativne sustave Windows i macOS. Podržava programske jezike C++, Python i LISP. Cilj razvoja ROS okvira je omogućiti standardizirani skup alata kako bi korisnici mogli brže i lakše razvijati svoja vlastita programska rješenja i kako bi vrlo lako mogli razmjenjivati kod.

Slika 3.1. Robot Operating System logo [5]

Radom ROS okvira upravlja jezgra ROS okvira (engl. *Roscore*) koja omogućuje komunikaciju između procesa i upravljanje s vremenom. Svi procesi pokretani koristeći ROS predstavljeni su kao čvorovi koji međusobno komuniciraju razmjenjivanjem poruka preko teme (engl. *Topic*) time čineći graf. Nit procesa koja šalje poruku naziva se izdavač (engl. *Publisher*) dok se nit koja prima poruku naziva pretplatnik (engl. *Subscriber*). Tema je naziv za tok poruka određenog tipa koji objedinjuje tip poruke koja se smije slati, vrijeme slanja poruke te same poruke. Temu kreira izdavač koji prvo oglašava naziv i tip teme pa se nakon toga pretplatnici mogu pretplatiti na tu temu kako bi primali poruke. Jezgra ROS okvira svakoj poruci dodjeljuje vrijeme slanja poruke i uklanja zastarjele poruke u temama. ROS ima implementirane neke od najčešće korištenih poruka (npr. trenutna pozicija i orijentacija vrha alata robota, kutovi zglobova robota, trajektorija vrha alata, slika kamere itd.), ali, ukoliko je potrebno, korisnik može definirati i svoj tip. Nad jednom temom nema ograničenja koliko može biti izdavača i pretplatnika te isto tako jedan proces može sadržavati više niti od kojih su neke izdavači a neke pretplatnici na različitim temama. Na slici 3.2 prikazan je graf s 3 procesa: P1 s jednim izdavačem i jednim pretplatnikom, P2 s jednim pretplatnikom i P3 s jednim izdavačem. Proces P1 šalje poruke procesu P2 preko teme t1 dok prima poruke od procesa P3 preko teme t2.



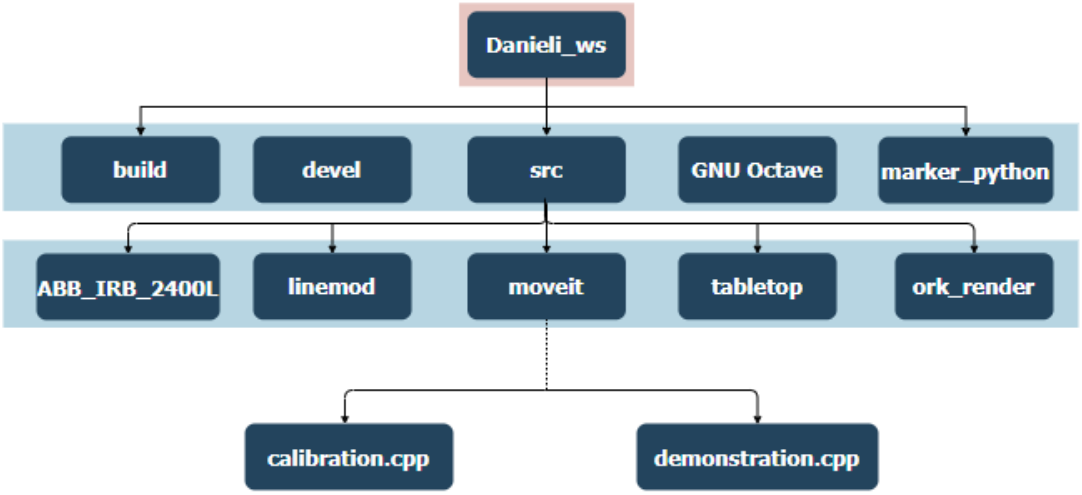
Slika 3.2 Primjer grafa čvorova ROS okvira

Za rad s konfiguracijskim paketima koristi se MoveIt!. Konfiguracijski paket je skup datoteka koji sadrži sve informacije potrebne za upravljanje manipulatorom na ROS platformi. MoveIt! je programski paket koji sadrži alate za modeliranje i upravljanje robotskim manipulatorima i mobilnim robotima što uključuje proračunavanje direktne i inverzne kinematike, planiranje trajektorije, vizualizaciju kretanja, simulaciju rada i navigaciju. MoveIt! paket omogućuje definiranje modela članaka manipulatora u konfiguracijskom paket čime se omogućuje vizualizacija kao i izbjegavanje sudara pojedinih članaka tijekom izvođenja kretnje manipulatora. MoveIt! također omogućava definiranje objekta u okolini manipulatora i izbjegavanje sudara s tim objektom tijekom planiranja i izvođenja putanje. Ovaj paket za rad koristi programsku datoteku tipa URDF (engl. *Universal Robot Description Format*) koju definira korisnik. URDF datoteka se temelji na XML jeziku i ona u potpunosti sadržava opis robotskog manipulatora s kojim se radi (npr. ovisnosti između zglobova manipulatora, međusobne udaljenosti pojedinih zglobova, materijal od kojega je napravljen pojedini članak itd.).

4. RAZVIJENI PROGRAMI

U svrhu izrade ovog diplomskog rada razvijena su dva C++ programa koja služe za kalibraciju sustava robot-kamera i za planiranje putanje vrha alata po objektu od interesa. Razvijeni programi su implementirane unutar ROS okvira. C++ programi koriste pomoćne skripte za proračune transformacijskih matrica i prepoznavanje objekata od interesa na slici dobivene od strane RGB-D kamere.

Programi kalibracije, planiranja putanje, prepoznavanja objekata, proračuna transformacijske matrice i pomoćni programi su objedinjeni u jedan radni prostor pod nazivom „Danieli_ws“(slika 4.1.).



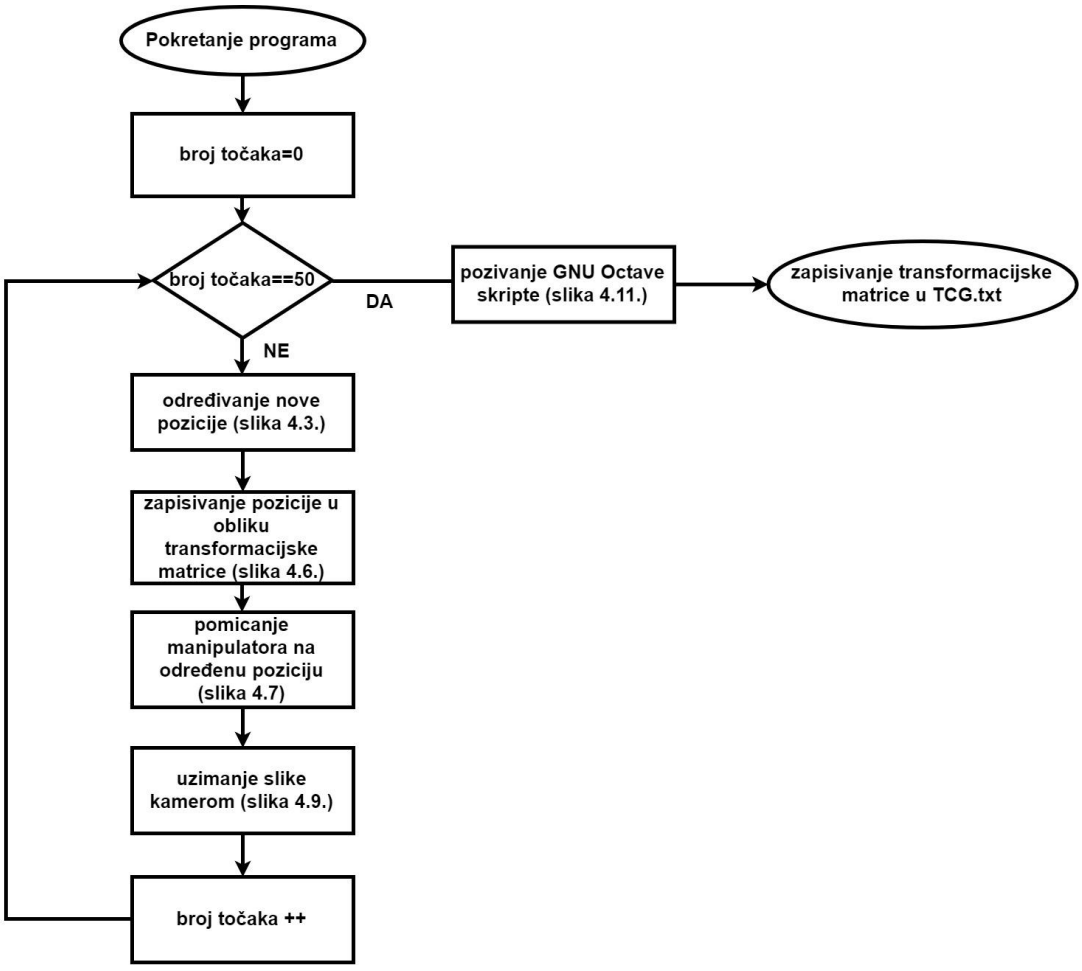
Slika 4.1. Radni prostor sa glavnim podmapama

U mapi src se nalaze ROS paketi koji se koriste prilikom kalibracije i planiranja putanje vrha alata manipulatora. „GNU Octave“ je mapa koja sadrži skriptu za proračunavanje transformacijske matrice iz koordinatnog sustava kamere u koordinatni sustav 6. zgloba manipulatora. „Marke_python“ sadrži python skriptu koja prepoznaje ArUco[6] markere na slici te vraća njihovu poziciju. „ABB_IRB_2400L“ mapa sadrži model robotskog manipulatora korištenog za testiranje razvijenog programa. Služi za uspostavljanje komunikacije između osobnog računala i upravljača robotskog manipulatora. „linemod“ i „tabletop“ su ROS paketi koji služe za detekciju objekta, a „ork_render“ služi za osposobljavanje kamere za rad unutar ROS okvira. Moveit mapa sadrži razvijene C++ programe za kalibraciju i planiranje putanje

vrha alata. Kalibracija se pokreće pomoću „calibration.cpp“, a planiranje putanje pomoću „demonstration.cpp“. Putanja do programa za kalibraciju i planiranje putanje vrha alata je: Danieli_ws/src/moveit/doc/move_group_interface/src.

4.1. Program za kalibraciju sustava robot-kamera

Kako bi sustav robot-kamera mogao raditi potrebno je odrediti odnose u translaciji i rotaciji koordinatnih sustava robota i kamere. Potrebno je odrediti transformacijsku matricu koja transformira vrijednosti pozicije i orijentacije detektiranog predmeta od strane kamere u bazni koordinatni sustav robotskog manipulatora. Informacije o poziciji i orijentaciji predmeta u baznom koordinatnom sustavu su nam potrebne jer se točke po kojima želimo da se 6. zglob manipulatora kreće zadaju u odnosu na bazni koordinatni sustav manipulatora. Na slici 4.2. je prikazan dijagram toka izvođenja algoritma za kalibraciju. Produkt izvršavanja ovog programa je tekstualna datoteka koja sadrži 4x4 transformacijsku matricu iz koordinatnog sustava kamere u koordinatni sustav 6. zgloba manipulatora.



Slika 4.2. Dijagram tok izvršavanje programa kalibracije

Program kalibracije sustava robot-kamera se sastoji od „main“ funkcije, funkcije „EyeOnHandCalib“ i pomoćnih funkcija. U funkciji „main“ se inicijaliziraju svi potrebni ROS čvorovi, te se poziva funkcija „EyeOnHandCalib“ koja izvršava cijeli kod kalibracije te poziva pomoćne funkcije opisane u nastavku ovog poglavlja. Za izvršavanje programa robotski manipulator se dovodi na 50 nasumično odabranih točaka u prostoru pod uvjetom da ArUco markeri(slika 5.5) budu u vidnom polju kamere postavljene na alat manipulatora. Odabrano je 50 točaka jer program za računanje radi na principu minimizacije pogreške te se povećanjem broja mjerenja dobivaju točniji rezultati. Rotacija alata manipulatora je ograničena kako bi se smanjio radni prostor izvedbe kalibracije. Kako bi markeri bili u vidnom polju kamere odabir pozicija je osmišljen na način da se virtualno povećava veličina alata manipulatora u z osi koordinatnog sustava 6. zgloba te se nakon toga vrh alata dovodi do centra markera uz dodana odstupanja kako bi dobili veću raznolikost podataka. Na slici 4.3. je prikazano postavljanje radnog prostora kalibracije.

```
318   float x_min=1.47;
319   float y_min=-0.100;
320   float z_min=0.6;
321
322   float x_max=1.54;
323   float y_max=-0.03;
324   float z_max=0.7;
325
326   float roll_min=-pi/8;
327   float pitch_min=pi/2-pi/8;
328   float yaw_min=pi-pi/8;
329
330
331   float roll_max=0+pi/8;
332   float pitch_max=pi/2+pi/8;
333   float yaw_max=pi;
```

Slika 4.3. Postavljanje radnog prostora kalibracije

„x_min“ i „x_max“ označavaju poziciju markera uz dodana odstupanja u odnosu na bazni koordinatni sustav manipulatora u x osi. Stvarna vrijednost sredine markera u x osi baznog koordinatnog sustava manipulatora je 1.505 metara.

„y_min“ i „y_max“ označavaju poziciju markera uz dodana odstupanja u odnosu na bazni koordinatni sustav manipulatora u y osi. Stvarna vrijednost sredine markera u y osi baznog koordinatnog sustava manipulatora je 0 metara.

„z_min“ i „z_max“ označavaju dužinu virtualnog alata odnosno duljinu alata po z osi koordinatnog sustava 6. zgloba manipulatora.

„roll_min“, „roll_max“, „pitch_min“, „pitch_max“, „yaw_min“ i „yaw_max“ označavaju maksimalne i minimalne vrijednosti u rotaciji alata manipulatora na kojem se nalazi kamera. Ove vrijednost su zadane u radijanima. Eulerovi kutovi se koriste samo zbog lakšeg razumijevanja zadanih ograničenja, odnosno prilikom zadavanja same željene orijentacije koriste se kvaternioni te je pretvorba iz Eulerovih kutova u kvaternione ostvarena pomoću funkcije „EulToQuant“(slika 4.4.).

```
464  void EulToQuant(float roll,float pitch,float yaw,float* quants){
465
466    float c1=cos(yaw*0.5);
467    float c2=cos(pitch*0.5);
468    float c3=cos(roll*0.5);
469    float s1=sin(yaw*0.5);
470    float s2=sin(pitch*0.5);
471    float s3=sin(roll*0.5);
472
473    quants[0]=c1*c2*c3-s1*s2*s3;
474    quants[1]=c1*c2*s3+s1*s2*c3;
475    quants[2]=s1*c2*c3+c1*s2*s3;
476    quants[3]=c1*s2*c3-s1*c2*s3;
477
478
479  }
```

Slika 4.4. Pretvorba Eulerovih kutova u kvaternione

Funkcija kao parametre prima 3 realna broja koji predstavljaju Eulerove kutove i pokazivač na polje koje predstavlja kvaternione koje želimo izračunati.

Kada zadajemo robotskom manipulatora poziciju na koju želimo da dođe onda zadajemo gdje će manipulator dovesti svoj 6. zglob, a kako je već ranije navedeno potrebno je dovesti vrh virtualnog alata do centra markera. Zbog toga je potrebno izračunati poziciju 6. zgloba manipulatora koja omogućuje dovođenje vrha virtualnog alata do željene točke. Računanje pozicije 6. zgloba manipulatora je ostvareno pomoću funkcije „jointPosCalc“(slika 4.5.) te se računa pomoću izraza:

$$jointPos = \begin{bmatrix} Eep\_x \\ Eep\_y \\ Eep\_z \end{bmatrix} - \begin{bmatrix} 1-2(q_y^2+q_z^2) & 2(q_xq_y-q_zq_w) & 2(q_xq_z+q_yq_w) \\ 2(q_xq_y+q_zq_w) & 1-2(q_x^2+q_z^2) & 2(q_yq_z-q_xq_w) \\ 2(q_xq_z-q_yq_w) & 2(q_yq_z+q_xq_w) & 1-2(q_x^2+q_y^2) \end{bmatrix} \cdot \begin{bmatrix} ToolTrans\_x \\ ToolTrans\_y \\ ToolTrans\_z \end{bmatrix} \quad (3\text{-}1)$$

```
MatrixXd jointPosCalc(MatrixXd ToolTrans,float Eep_x,float Eep_y,float Eep_z,
                      float q_w,float q_x,float q_y,float q_z){
 MatrixXd RotationTool(3,3);
 MatrixXd End_effector_pos(3,1);
 MatrixXd jointPos(3,1);
 End_effector_pos(0,0)=Eep_x;
 End_effector_pos(1,0)=Eep_y;
 End_effector_pos(2,0)=Eep_z;
 RotationTool(0,0)=1 - 2 * q_y*q_y - 2 * q_z*q_z;
 RotationTool(0,1)=2 * q_x*q_y - 2 * q_z*q_w;
 RotationTool(0,2)=2 * q_x*q_z + 2 * q_y*q_w;
 RotationTool(1,0)=2 * q_x*q_y + 2 * q_z*q_w;
 RotationTool(1,1)=1 - 2 * q_x*q_x - 2 * q_z*q_z;
 RotationTool(1,2)=2 * q_y*q_z - 2 * q_x*q_w;
 RotationTool(2,0)=2 * q_x*q_z - 2 * q_y*q_w;
 RotationTool(2,1)=2 * q_y*q_z + 2 * q_x*q_w;
 RotationTool(2,2)=1 - 2 * q_x*q_x - 2 * q_y*q_y;

 jointPos=End_effector_pos-RotationTool*ToolTrans;
 return jointPos;
}
```

Slika 4.5. Računanje pozicije 6. zglobа uz zadanu poziciju vrha alata

„Eep_x“, „Eep_y“, „Eep_z“, „q_w“, „q _x“, „q_y“ i „q_z“ su vrijednosti koje opisuju poziciju na koju želimo dovesti vrha alata manipulatora, a „ToolTrans“ je matrica 3x1 koja sadrži translaciju vrha alata od 6. zgloba alata u koordinatnom sustavu 6. zgloba robotskog manipulatora.

Kada je pozicija 6. zgloba manipulatora izračunata za željenu točku potrebno je zapisati vrijednosti te pozicije kako bi se poslije mogli koristiti za izračunavanje transformacijske matrice iz koordinatnog sustava kamere u koordinatni sustav 6. zgloba manipulatora i samim time u bazni koordinatni sustav manipulatora. Ovi se podaci zapisuju pomoću transformacijske matrice, što znači da trebamo podatak o poziciji i orijentaciji 6. zgloba koji je zapisan pomoću 7 brojeva (prva 3 označavaju poziciju, a zadnja 4 označavaju orijentaciju pomoću kvaterniona) prebaciti u oblik transformacijske matrice. Ovaj postupak je izveden pomoću funkcije „TransformationMatrixCalc“ (slika 4.6.).

```
967  MatrixXd TransformationMatrixCalc(
968    float x,float y,float z,float q_w,float q_x,float q_y,float q_z){
969   MatrixXd T_Matrix(4,4);
970   T_Matrix(0,0)=1 - 2 * q_y*q_y - 2 * q_z*q_z;
971   T_Matrix(0,1)=2 * q_x*q_y - 2 * q_z*q_w;
972   T_Matrix(0,2)=2 * q_x*q_z + 2 * q_y*q_w;
973   T_Matrix(0,3)=x;
974   T_Matrix(1,0)=2 * q_x*q_y + 2 * q_z*q_w;
975   T_Matrix(1,1)=1 - 2 * q_x*q_x - 2 * q_z*q_z;
976   T_Matrix(1,2)=2 * q_y*q_z - 2 * q_x*q_w;
977   T_Matrix(1,3)=y;
978   T_Matrix(2,0)=2 * q_x*q_z - 2 * q_y*q_w;
979   T_Matrix(2,1)=2 * q_y*q_z + 2 * q_x*q_w;
980   T_Matrix(2,2)=1 - 2 * q_x*q_x - 2 * q_y*q_y;
981   T_Matrix(2,3)=z;
982   T_Matrix(3,0)=0;
983   T_Matrix(3,1)=0;
984   T_Matrix(3,2)=0;
985   T_Matrix(3,3)=1;
986   return T_Matrix;
987
988  }
```

Slika 4.6. Računanje transformacijske matrice

Funkcija kao parametre prima 7 realnih brojeva. Prva 3 označavaju pomake u x, y i z osi, a zadnja 4 označavaju kvaternione koji predstavljaju orijentaciju. Funkcija vraća matricu 4x4 koja označava transformacijsku matricu za predane vrijednosti.

Nakon što se odredila pozicija na koju želimo dovesti 6. zglob manipulatora kretanje manipulatora se izvršava pomoću move_group objekta, funkcije „setPoseTarget“ i funkcije „move“(slika 4.7.) .

```
404    move_group.setPoseTarget(target_pose);
405
406    move_group.move();
```

Slika 4.7. Naredbe za pomicanje manipulatora na željenu poziciju

Funkciji „setPoseTarget“ se predaje parametar koji predstavlja poziciju i orijentaciju na koju želimo dovesti 6. zglob manipulatora(slika 4.8.).

```
384     target_pose.position.x=jointPos(0,0);
385     target_pose.position.y=jointPos(1,0);
386     target_pose.position.z=jointPos(2,0);
387
388     target_pose.orientation.w = quants[0];
389     target_pose.orientation.x = quants[1];
390     target_pose.orientation.y = quants[2];
391     target_pose.orientation.z = quants[3];
```

Slika 4.8. Dodjeljivanje vrijednosti pozicije i orijentacije

Vrijednosti pozicije se dodjeljuju „target_pose.position“ u obliku x, y, z, a orijentacija se dodjeljuje „target_pose.orientation“ u obliku kvaterniona qw, qx, qy, qz.

Kada je manipulator došao na željenu poziciju program odlazi u „sleep“ 1 sekundu kako bi se manipulator, a samim time i kamera koja se nalazi na manipulatoru prestala tresti. Kada je prošla 1 sekunda uzima se slika pomoću kamere. Ukoliko bi izdali naredbu za slikanje odmah nakon što je manipulator došao na željenu poziciju, slika bi mogla biti mutna.

Pokretanje naredbe za uzimanje slike s kamere je vidljivo na slici 4.9.

```
413     sleep(1);
414     system("rosservice call /image_saver/save");
415     sleep(1);
```

Slika 4.9. Uzimanje slike s kamere

Ovaj postupak se ponavlja za svih 50 nasumično odabranih pozicija, te se nakon toga poziva python skripta koja na slikama koje su uzete na svakoj pojedinačnoj poziciji traži ArUco markere, te sprema vrijednosti njihove pozicije u tekstualnu datoteku. Pozivanje skripte je vidljivo na slici 4.10.

```
562   system(" cd ~/Danieli_ws/marker_pyhton/Aruco_Tracker && python aruco_sve_u_mapi.py ");
```

Slika 4.10. Pozivanje skripte za prepoznavanje markera na slici

Računanje transformacijske matrice se izvršava pomoću GNU Octave[7] skripte koja se poziva unutar C++ programa za kalibraciju pomoću narede vidljive na slici 4.11.

```
565   system("cd /home/ferit/Danieli_ws/Matlab && octave handEyeCalibration3Demo.m");
```

Slika 4.11. Pozivanje GNU Octave skripte

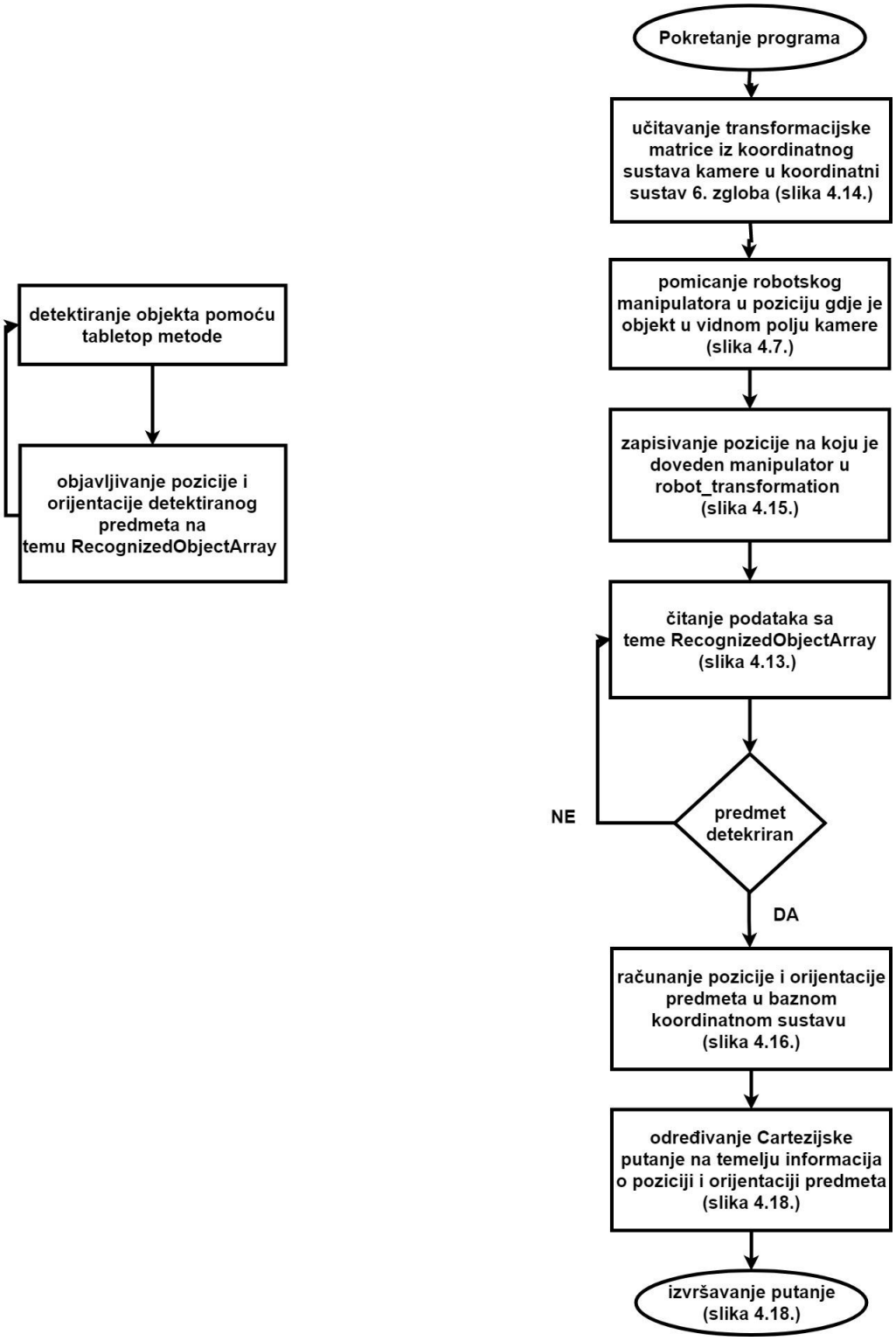
GNU Octave skripta nakon pokretanja stvara tekstualnu datoteku koja sadrži 16 brojeva koji označavaju transformacijsku matricu iz koordinatnog sustava kamere u koordinatni sustav 6. zgloba manipulatora.

4.2. Program za planiranje putanje vrha alata

Za potrebe obrade površine predmeta potrebno je omogućiti linearno kretanje vrha alata robotskog manipulatora po površini predmeta. Kako bi se ostvarila linearna kretnja vrha alata manipulatora unutra ROS okvira korišten je MoveIt! programski paket. Za ostvarivanje ovog tipa kretnje korištena je kartezijska putanja koja je u MoveIt!-u ostvarena na način da se zadaje skup točaka po kojima želimo da se vrha alata manipulatora kreće.

U ovom diplomskom radu obrađuje se tematika gdje robotski manipulator treba obrađivati površinu predmeta koji se ne nalazi uvijek na istoj poziciji i orijentaciji u odnosu na robotski manipulator već je potrebno isplanirati putanju vrha alata ovisno o informacijama dobivenih od RGB-D kamere. Informacije od trenutnoj poziciji i orijentaciji predmeta se šalje pomoću ROS poruka.

Program za planiranje putanje vrha alata radi na principu prikazanom na slici 4.12. Uz pokretanje programa za planiranje putanje vrha alata potrebno je pokrenuti i program za detekciju predmeta odnosno tabletop [8] ROS čvor. Tabletop ROS čvor konstanto objavljuje informacije o detektiranom predmetu na ROS temu „RecognizedObjectArray“.



Slika 4.12. Dijagram toka izvršavanja programa za planiranje putanje vrha alata

Program koji upravlja kretanjem robotskog manipulatora dohvaća podatke o detektiranom predmetu pomoću funkcije prikazane na slici 4.13. Na temu „RecognizedObjectArray“ se šalje polje svih potencijalnih predmeta. „Object_recognition_msgs“ sadrži ROS poruke i definicije

actionlib servera za prepoznavanje objekta [9]. Svakom potencijalnom detektiranom predmetu je dodijeljena vrijednost koja označava koliko je algoritam za prepoznavanje siguran da je na toj poziciji i orijentaciji stvarni predmet. „objects_msg“ sadrži informacije o poziciji i orijentaciji detektiranog predmeta kao i vrijednost sigurnosti. Pomoću for petlje se prolazi kroz sve elemente polja „objects“ te se uzima onaj koji ima najveću vrijednost sigurnosti.

```
71  void objectCallback(const object_recognition_msgs::RecognizedObjectArray objects_msg) {
72
73      if((int)objects_msg.objects.size() > 0){
74   int j=0;
75      double conf=0;
76
77
78      // pronađi predmet s najvecom vjerovatnosti
79      for (int i = 0; i < objects_msg.objects.size(); ++i) {
80          if(objects_msg.objects[i].confidence > conf){
81              conf= objects_msg.objects[i].confidence;
82              j=i;
83          }
84      }
85
86
87          object_qx = objects_msg.objects[j].pose.pose.pose.orientation.x;
88          object_qy = objects_msg.objects[j].pose.pose.pose.orientation.y;
89          object_qz = objects_msg.objects[j].pose.pose.pose.orientation.z;
90          object_qw = objects_msg.objects[j].pose.pose.pose.orientation.w;
91          object_x=objects_msg.objects[j].pose.pose.pose.position.x;
92          object_y=objects_msg.objects[j].pose.pose.pose.position.y;
93          object_z=objects_msg.objects[j].pose.pose.pose.position.z;
94     id = objects_msg.objects[j].type.key.c_str();
95
96
97      }
98
99
100 }
```

Slika 4.13. Dohvaćanje podataka o poziciji i orijentaciji predmeta

Informacije o poziciji i orijentaciji predmeta su zapisane u odnosu na koordinatni sustav kamere te ih je potrebno transformirati u bazni koordinatni sustav robotskog manipulatora. Kako bi ovo ostvarili potrebna je transformacijska matrica iz koordinatnog sustava kamere u koordinatni sustav 6. zgloba robotskog manipulatora. Što znači da je prvo potrebno obaviti kalibraciju sustava opisanu u poglavlju 4.1.

Kada smo odradili kalibraciju potrebno je učitati podatke transformacijske matrice u program za planiranje putanje vrha alata manipulatora(slika 4.14.).

```
1085    ifstream myfile;
1086    myfile.open("/home/ferit/Danieli_ws/GNU_Octave/TCG.txt");
1087
1088    for(int i=0;i<4;i++){
1089    for (int j=0;j<4;j++){
1090    myfile >> TCG(i,j);
1091    }
1092    }
```

Slika 4.14. Učitavanje transformacijske matrice

Nakon što je učitana transformacijska matrica potrebno je dovesti robotski manipulator u poziciju gdje kamera može vidjeti objekt od interesa. Pomicanje robota do željene pozicije se izvršava pomoću funkcije „move“ opisane u poglavlju 4.1. Kada je robotski manipulator došao u željenu poziciju potrebno je zabilježiti poziciju i orijentaciju 6. zgloba manipulatora jer nam ranije izračunata transformacijska matrica transformira poziciju i orijentaciju objekta iz koordinatnog sustava kamere u koordinatni sustav 6. zgloba manipulatora, a nama je potrebna informacija o poziciji i orijentaciji predmeta u baznom koordinatnom sustavu robotskog manipulatora. Za određivanje transformacije iz koordinatnog sustava 6. zgloba u bazni koordinatni sustav koristi se funkcija „TransformationMatrixCalc“(slika 4.15.) opisana u poglavlju 4.1. Parametri koje prima funkcija označavaju poziciji i orijentaciju 6. zgloba manipulatora u trenutku kada je objekt detektiran, odnosno robot_x, robot_y i robot_z su koordinate 6. zgloba u pojedinim osima, a robot_qw, robot_qx, robot_qy i robot_qz su vrijednosti kvaterniona koji označavaju orijentaciju 6. zgloba manipulatora.

```
572    robot_transformation= TransformationMatrixCalc(robot_x,robot_y,robot_z,robot_qw,robot_qx,robot_qy,robot_qz);
```

Slika 4.15. Računanje transformacijske matrice iz koordinatnog sustava 6 zgloba u bazni koordinatni sustav

Kada je zapisana transformacijska matrica iz koordinatnog sustava 6. zgloba manipulatora u bazni koordinatni sustav, manipulator stoji na zadanoj poziciji sve dok kamera ne prepozna predmet. Kada je predmet prepoznat, izračunava se gdje se predmet nalazi u baznom koordinatnom sustavu. Rezultat ovog procesa je transformacijska matrica koja govori gdje se predmet nalazi u odnosu na bazni koordinatni sustav.

Nakon što je određeno gdje se predmet nalazi u odnosu na bazni koordinatni sustav, može se početi planirati putanja vrha alata po predmetu. Točke po kojima se manipulator treba kretati moraju biti zadane u koordinatnom sustavu predmeta odnosno ukoliko se koordinatni sustav predmeta prilikom izrade modela predmeta postavio u sredinu predmeta, onda bi vrh alata manipulatora doveli do sredine predmeta zadajući koordinate:

- X=0
- Y=0
- Z=0

Kada su točke po kojima se vrh alata treba kretati određene, potrebno je ih pomnožiti s transformacijskom matricom „object_transformation“ koja nam govori gdje se predmet nalazi u odnosu na bazni koordinatni sustav(slika 4.16.), odnosno transformacijskom matricom iz koordinatnog sustava predmeta u bazni koordinatni sustav manipulatora.

```
140    BoxPoint_base=object_transformation*BoxPoint;
```

Slika 4.16. Računanje pozicije točke na predmetu u baznom koordinatnom sustavu

„BoxPoint_base“ je matrica koja sadrži podatke gdje se odabrana točka nalazi u baznom koordinatnom sustavu, a „BoxPoint“ je matrica koja sadrži podatke o željenoj točki u koordinatnom sustavu predmeta.

Kada bi zadali točku koja se nalazi u matrici „BoxPoint_base“ robotski manipulator bi došao sa 6. zglobom na zadanu točku. Zato je potrebno izračunati gdje je potrebno dovesti 6. zglob manipulatora da vrh alata bude na zadanoj točki. Ovaj proračun se odrađuje postupkom opisanim u poglavlju 4.1. odnosno pomoću funkcije „jointPosCalc“. Funkciji se predaje točka na koju želimo dovesti vrha alata manipulatora, orijentacija alata i translacijski pomak vrha alata u odnosu na 6. zglob manipulatora u koordinatnom sustavu 6. zgloba manipulatora.

Točke po kojima želimo da se manipulator kreće se zadaju pomoću „waypoint“ objekta(slika 4.17.), a zadaju se pomoću „target_pose“ objekta na način opisan na slici 4.8.

```
451        waypoints.push_back(target_pose);
```

Slika 4.17. Zadavanje točaka po kojima će se vrha alata kretati

Kada su sve točke odabrane, prije nego se manipulatoru može zadati naredba za kretanje po zadanoj putanji potrebno je postaviti nekoliko parametara kretnje. Obrada površine je funkcija

koja zahtjeva manju brzinu kretanja manipulatora stoga je maksimalna brzina postavljena na 10% maksimalne brzine manipulatora. Kako bi se putanja mogla isplanirati potrebno je odrediti rezoluciju uzorkovanja između dvije zadane točke. Rezolucija je postavljena na 1 cm(slika 4.18).

```
458   move_group.setMaxVelocityScalingFactor(0.1);
459   const double jump_treshold=0.0;
460   const double eef_step=0.01;
461   double fraction = move_group.computeCartesianPath(waypoints,eef_step,jump_treshold, trajectory);
462   my_plan.trajectory_=trajectory;
463   move_group.execute(my_plan);
```

Slika 4.18. Izvršavanje kretnje manipulatora po zadanoj putanji

„move_group“ je objekt kojem je pridružen konfiguracijski paket manipulatora, „setMaxVelicityScalingFactor“ je funkcija koja omogućuje ograničavanje maksimalne brzine manipulatora, „computeCartesianPath“ je funkcija koja na temelju odabranih točaka i odabrane rezolucije putanje stvara putanju te je zapisuje u „trajectory“. „trajectory“ je MoveIt! poruka, a my_plan je objekt koji služi za izvršavanje putanje. Pomoću funkcije „execute“ se izvršava plan putanje koji je izračunat pomoću odabranih točaka i parametara putanje.

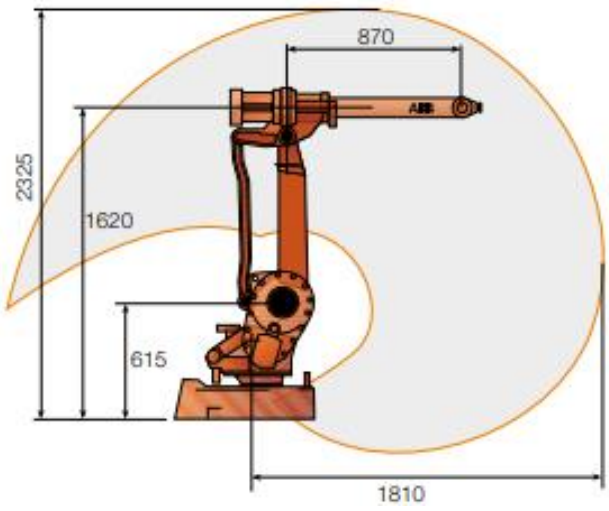
5. TESTIRANJE SUSTAVA

Manipulator na kojem je ispitana funkcionalnost razvijenog programa je manipulator tvrtke ABB. ABB je svjetski poznati proizvođač robotskih manipulatora koji je proizveo i pustio u pogon više od 300000 robotskih manipulatora. Model korištenog manipulatora je IRB 2400L (slika 5.1.).

Slika 5.1. Robotski manipulator IRB 2400L

Navedeni manipulator je na korištenje ustupila tvrtka DANIELI SYSTEC d.o.o., koja surađuje s Fakultetom elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

Model IRB 2400L ima dohvat 1.8 metara i nosivost od 7 kg, te su njegove dimenzije vidljive na slici 5.2.



Slika 5.2. dimenzije manipulatora IRB 2400L [10]

Navedeni manipulator ima široku primjenu u industriji, a primarne funkcije su mu lučno zavarivanje, prijenos i obrada materijala. Manipulator je veliku nosivost mijenjao za veliko radno područje, uske članke i zglobove, što ga čini optimalnim za ranije navedene primjene.

Robusna konstrukcija i minimalistički pristup prilikom dizajna dijelova manipulatora doprinosi visokoj pouzdanosti i dugačkim intervalima između održavanja.

Na 6. članak manipulatora je postavljena hvataljka proizvođača Schunk model PGN 100-1-AS prikazana na slici 5.3.

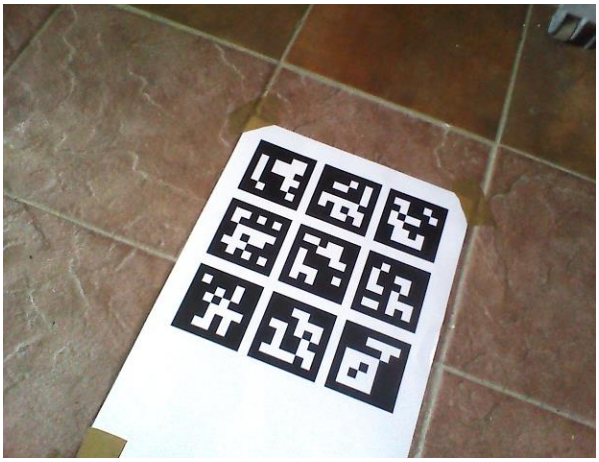
Slika 5.3. Schunk hvataljka PGN 100-1-AS [6]

Za detekciju objekta od interesa koristi se RGB-D kamera postavljena na hvataljku manipulatora. Korištena je kamera Orbbec Astra S koja je učvršćena na hvataljku pomoću 3D isprintanog postolja. Kamera postavljena na hvataljku manipulatora vidljiva je na slici 5.4.

Slika 5.4. RGB-D kamera postavljena na hvataljku manipulatora

5.1. Provedba pokusa kalibracije

Kako bi proveli pokus kalibracije prvo je potrebno odabrati objekt koji će služiti kao referenta točka kalibracije odnosno taj objekt će biti statičan u prostoru dok se robotski manipulator, a zajedno s njim i kamera kreću u prostoru. U svrhu provedbe kalibracije za taj objekt je odabran A3 format papira sa 9 ArUco markera zalijepljen na pod ispred manipulatora (slika 5.5).



Slika 5.5. ArUco markeri

Nakon što je promatrani objekt odabran potrebno je definirati gdje se on nalazi u odnosu na bazni koordinatni sustav, te pomoću postupka opisanog na slici 4.3. odrediti radni prostor robotskog manipulatora tijekom kalibracije. Kada je određen radni prostor kalibracije potrebno je manipulator dovesti na 50 pozicija u zadanom radnom prostoru na način da ArUco markeri budu vidljivi kameri u svakoj poziciji. Manipulator tijekom izvođenja kalibracije je vidljiv na slici 5.6.

Slika 5.6. Izvođenje kalibracije

U svakoj poziciji se manipulator zadrži 1 sekundu prije uzimanja slike kamerom kako bi se otklonila mogućnost da slika bude mutna jer se manipulator još kretao. Nakon odrađenih 50 točaka pokreće se GNU Octave skripta koja računa transformacijsku matricu, te je zapisuje u tekstualnu datoteku. Produkt kalibracije je transformacijska matrica:

$$ {}^{C}_{6}T = \begin{bmatrix} -0.9992831329303 & 0.0057784603851095 & 0.03741429722111013 & 0.0057770372546468 \\ -0.004076710693673 & -0.9989605015499271 & 0.045401506286006 & 0.0762434280212597 \\ 0.03763775592263618 & 0.04521643217565852 & 0.9982679367736961 & 0.06511862098841585 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5\text{-}1) $$

5.2. Testiranje kalibracije i programa za planiranje putanje alata

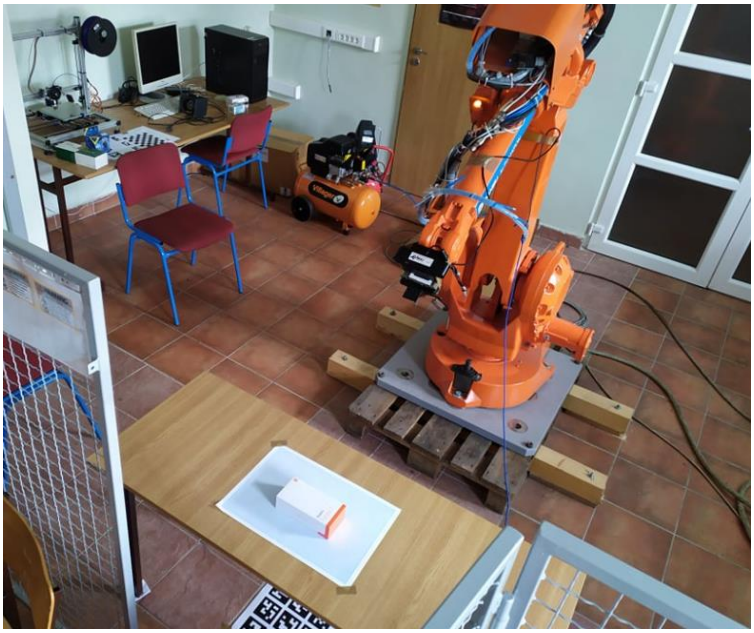
Kako bi se testirala kalibracija kao i rad cjelokupnog sustava planiranja putanje alata na temelju 3D snimke osmišljen je pokus gdje se objekt prikazan na slici 5.7. postavlja u radni prostor manipulatora te se laser koji je postavljen na manipulator(slika 5.10.) dovodi na vrh kocke te preko bridova dolazi do ostalih vrhova.

Slika 5.7. Predmet nad kojim se vrši manipulacija

Kako bi se pogreške detektiranja predmeta i planiranja putanje vrha alata mogle mjeriti predmet sa slike 5.7. je postavljen na milimetarski papir te je naznačeno gdje se predmet nalazi na papiru(slika 5.8.).

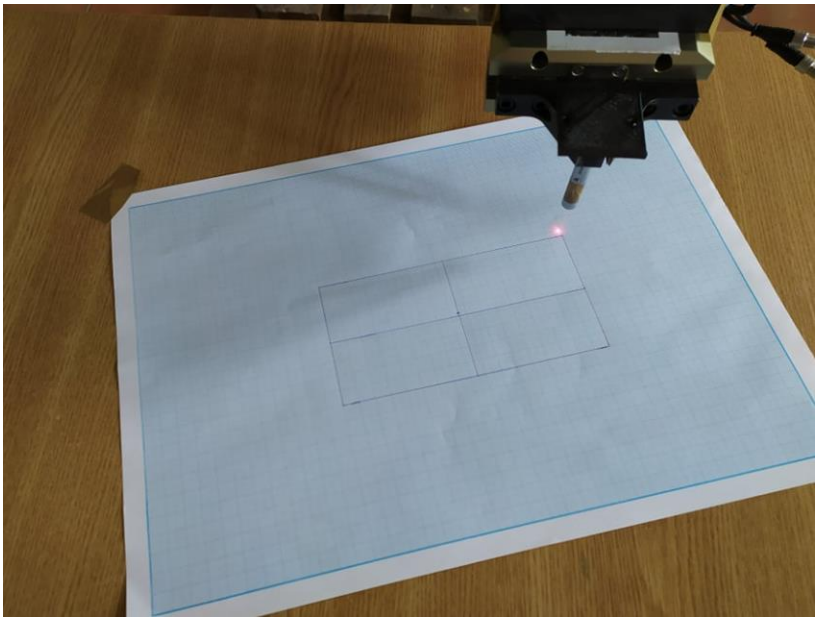
Slika 5.8. Predmet nad kojim se vrši manipulacija postavljen na milimetarski papir

Nakon što je predmet postavljen na milimetarski papir te je njegova pozicija označena pokreće se program za planiranje putanje vrha alata (Prilog A). Manipulator prvo dolazi u poziciju gdje je predmet u vidnom polju kamere(slika 5.9.), te se vrši prepoznavanje pozicije i orijentacije predmeta. Kada je predmet pronađen, te je određena njegova pozicija i orijentacija u baznom koordinatnom sustavu, predmet je uklonjen s milimetarskog papira kako bi se mogla označiti putanja vrha alat na papiru.



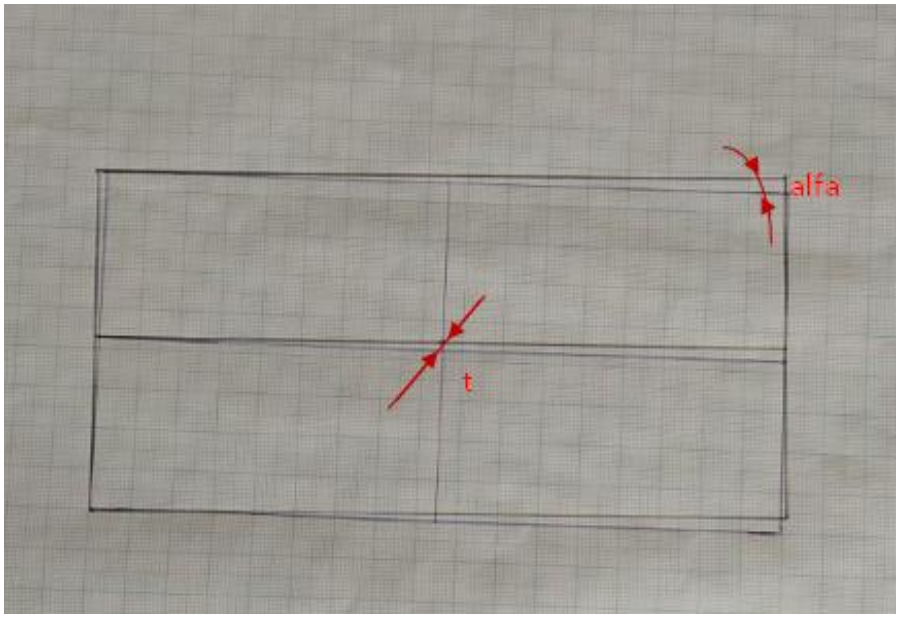
Slika 5.9. Pozicija za prepoznavanje objekta

Vrh alata manipulatora se dovodi do vrha detektiranog predmeta na način da laser postavljen kao vrh alata bude okomit na milimetarski papir, te se na milimetarskom papiru označuje gdje laser baca zrake(slika 5.10.).



Slika 5.10. Bilježenje greške sustava

Nakon označavanja greške sustava na jednom vrhu vrh alata se preko brida predmeta kreće prema drugom vrhu, te se ponovno označava greška mjerenja. Postupak se ponavlja za sva 4 vrha, te je kao produkt dobivena pozicija i orijentacija predmeta detektiranog od strane sustava označena na milimetarskom papiru zajedno s tlocrtom premeta koji se prepoznaje (slika 5.11.).



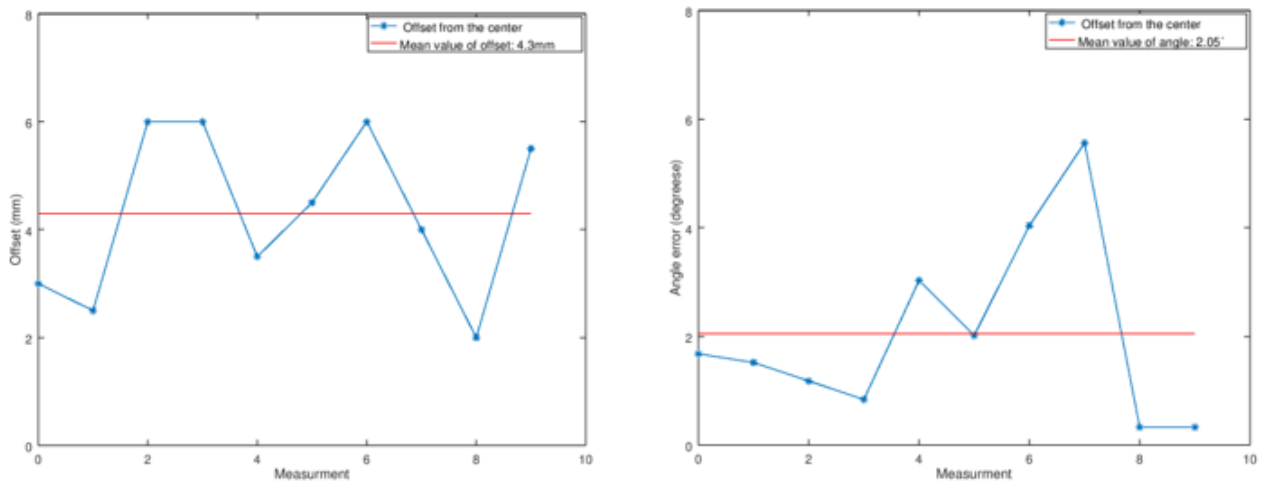
Slika 5.11. Označene pogreške sustava

Kao dva mjerila pogreške sustava mjerila se translacijska pogreška i rotacijska pogreška. Translacijska pogreška označena na slici 5.11. sa „t“ označava udaljenost centra detektiranog predmeta od stvarnog centra predmeta i mjerena je u milimetrima, a rotacijska pogreška označena sa „alfa“ je kut koji zatvaraju stvarni brid i detektirani brid predmeta. Mjerenje je ponovljeno 10 puta za različite pozicije i orijentacije predmeta te su rezultati zapisani u tablicu:

Mjerenje	t[mm]	alfa[°]
1.	3.1	1.7
2.	2.5	1.6
3.	6.15	1.3
4.	6.15	0.7
5.	3.6	3
6.	4.6	2
7.	6.15	4
8.	4.1	5.5
9.	2.4	0.35
10.	5.7	0.35

Tablica 5.1. Greške u translaciji i rotaciji detektiranog predmeta

Prosječna pogreška u translaciji je 4.3 milimetra, a u rotaciji je 2 °. Mjerni podaci su prikazani na grafu 5.1. Na lijevoj strani je prikazana greška u translaciji, a na desnoj greška u rotaciji.



Graf 5.1. Greške u translaciji i rotaciji detektiranog predmeta

6. ZAKLJUČAK

Robotski manipulatori imaju široku primjenu u industrije te se sve više koriste u dinamičkim okolinama gdje moraju manipulirati s objektima koji se ne nalaze uvijek na istim pozicijama. Kako bi ovo bilo moguće potrebno je u sustav upravljanja manipulatorom uvesti dodatni senzor koji će detektirati poziciju objekta. U ovom radu informacije o poziciji objekta od interesa se dobivaju iz 3D slike te se na temelju tih informacija provodi planiranje putanje vrha alata manipulatora po objektu. U okviru rada napravljen je program za kalibraciju sustava robot-kamera i program za planiranje putanje vrha alata u C++ programskom jeziku unutar ROS okvira. Kretanje vrha alata po objektu je proces koji zahtjeva visoku preciznost odnosno preciznost s maksimalnim odstupanjem ispod 1 milimetar. Eksperimentalnim dijelom ovog rada zaključeno je da 3D kamera nije prikladan samostalni senzor za detekciju pozicije objekta ukoliko je potrebna obrada površine objekta. Prosječna pogreška u translaciji detektiranog objekta je 4.3 milimetara u baznom koordinatnom sustavu robotskog manipulatora, a prosječna pogreška u rotaciji objekta je 2 stupnja. Razvijeni sustav je prikladniji za zadatke poput dohvaćanja objekta hvataljkom jer ova radnja tolerira manja odstupanja u detekciji pozicije objekta u baznom koordinatnom sustavu manipulatora. Poboljšanje rada sustava je moguće dodavanjem senzora sile na vrh alata. Pomoću kamere bi se odredila pozicija predmeta, a regulacijom sile moguće je pozicionirati alat s preciznošću koja je potrebna za obradu površine.

LITERATURA

[1] Robert Cupec, Osnove robotike – Osnove inteligentnih robotskih sustava

[2] http://www.chrobotics.com/wp-content/uploads/2012/11/Inertial-Frame.png posjećeno 18.9.2019

[3] Robert Cupec, Osnove robotike - predavanje 6. Planiranje trajektorije

[4] https://en.wikipedia.org/wiki/Robot_Operating_System , posjećeno 18.9.2019

[5] https://upload.wikimedia.org/wikipedia/commons/thumb/b/bb/Ros_logo.svg/1280px-Ros_logo.svg.png , posjećeno 18.9.2019

[6] https://www.uco.es/investiga/grupos/ava/node/26 , posjećeno 18.9.2019

[7] https://www.gnu.org/software/octave/, posjećeno 18.9.2019

[8] http://wiki.ros.org/tabletop_object_detector , posjećeno 18.9.2019

[9] https://github.com/wg-perception/object_recognition_msgs , posjećeno 18.9.2019

[10] ABB IRB 2400 data sheet

[11] https://schunk.com/hu_en/gripping-systems/product/16954-0370402-pgn-100-1-as/, posjećeno 18.9.2019

SAŽETAK

U ovom radu je obrađena problematika planiranja putanje vrha alata robotskog manipulatora po površini objekta od interesa pomoću informacija dobivenih 3D kamerom. U svrhu izrade ovog rada izrađen je program unutar ROS(*Robot Operating System*) okvira koji planira putanju vrha alata industrijskog robota. Unutar ROS okvira također je izrađena procedura za kalibraciju sustava robot-kamera koja omogućuje računanje transformacijskih matrica iz koordinatnog sustava kamere u koordinatni sustav robotskog manipulatora. Planiranje putanje vrha alata i procedura za kalibraciju sustava robot-kamera su izvedeni u C++ programskom, a računanje transformacijskih matrica iz koordinatnog sustava kamere u koordinatni sustav robota je izvedeno u MATLAB programskom jeziku. Točnost izrađenog sustava eksperimentalno je ispitana na industrijskom robotu ABB IRB 2400L .

Ključne riječi: robotski manipulator, ROS(*Robot Operating System*), kalibracija sustava robot-kamera, C++, MATLAB

ABSTRACT

In this thesis path planning along the surface of an object of interest for a robotic manipulator end effector is considered. The object pose is determined using a 3D camera. A software for planning the path of an industrial robot end effector is developed within the ROS framework. Furthermore, a hand-eye calibration procedure, which computes the transformation matrix from the camera coordinate system to the robot coordinate system was implemented. The path planning and hand-eye calibration procedures were created in C++ programming language and the calculation of the transformation matrix from the camera coordinate system to the robot coordinate system was made in MATLAB programming language. The accuracy of the developed system was experimentally tested on the industrial robot ABB IRB 2400L .

Keywords: robotic manipulator, ROS(*Robot Operating System*), Hand-eye calibration, C++, MATLAB

ŽIVOTOPIS

Adrian Čičić rođen je 29.11.1995. godine u Vinkovcima. Od 2002. do 2010. pohađao je OŠ „Antun Gustav Matoš“ u Vinkovcima. Nakon završene OŠ upisao je srednju tehničku školu „Ruđera Boškovića“ u Vinkovcima, smjer mehatronika. Maturirao je 2014. godine, te je zbog izvrsnog uspjeha u srednjoj školi ostvario pravo na izravan upis na Elektrotehnički fakultet u Osijeku, koji je upisao iste godine. Upisao je preddiplomski studij računarstva. Preddiplomski studij je završio 2017. godine te je upisao diplomski studij računarstva smjer procesno računarstvo. 2018. godine dobio je rektorovu nagradu za izvrstan seminarski rad iz predmeta Osnove robotike.

_____

Adrian Čičić

PRILOG A- KORIŠTENJE RAZVIJENOG SUSTAVA

Kako bi koristili razvijeni sustav potrebno je na osobno računalo instalirati sljedeće:

- Ubuntnu 16.04.- http://releases.ubuntu.com/16.04/
- ROS kinetic - http://wiki.ros.org/kinetic/Installation/Ubuntu
- ROS omot (engl. *wrapper*) za Orbbec Astru - http://wiki.ros.org/astra_camera
- ORK (Object Recognition Kitchen) - https://wg-perception.github.io/object_recognition_core/install.html
- ROS MoveIT! - https://moveit.ros.org/install/
- Bazu podataka - https://wg-perception.github.io/object_recognition_core/infrastructure/couch.html#object-recognition-core-db
- GNU Octave - https://www.gnu.org/software/octave/

Nakon što su instalirani operacijski sustav i svi potrebni programi kalibracija sustava robot-kamera se pokreće :

1. Povezivanje osobnog računala s upravljačem manipulatora

$roslaunch abb_irb2400_moveit_config moveit_planning_execution.launch sim:=false

robot_ip:=172.16.106.185.

2. Pokretanje ROS paketa kamere

$roslaunch astra_launch astra.launch

3. Pokretanje programa za spremanje slika

$rosrun image_view image_saver image:=camera/rgb/image_raw _save_all_image:=false

_filename_format:=%d.%s __name:=image_saver

4. Pokretanje kalibracije
   $roslaunch moveit Calibration.launch

Pokretanje programa za planiranje putanje:

1. Povezivanje osobnog računala s upravljačem manipulatora

$roslaunch abb_irb2400_moveit_config moveit_planning_execution.launch sim:=false

robot_ip:=172.16.106.185.

2. Pokretanje ROS paketa kamere

$roslaunch astra_launch astra.launch

3. Pokretanje programa za prepoznavanje objekta:

$rosrun object_recognition_core detection -c src/tabletop/conf/detection.object.ros.ork

4. Pokretanje programa za planiranje putanje vrha alata

$roslaunch moveit Demonstration.launch