Relativno vođenje robota koristeći 3D vizijski sustav

Mihalinec, Dominik

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Dominik Mihalinec

Zagreb, 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Prof. dr. sc. Bojan Jerbić, dipl. ing.

Student:

Dominik Mihalinec

Zagreb, 2015.

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

Zahvaljujem se svojem mentoru prof. dr. sc. Bojanu Jerbiću i asistentu mag. ing. Filipu Šuligoju na pruženoj stručnoj pomoći, ustupljenoj literaturi i korisnim savjetima tijekom izrade rada, te asistentu dr. sc. Marku Katiću na stručnoj pomoći u Laboratoriju za precizna mjerenja dužina (LFSB).

Posebno se zahvaljujem majci i ocu što su mi bili velika podrška tijekom studiranja i omogućili mi da ovaj studij uspješno privedem kraju.

Zahvaljujem se kolegama i prijateljima na pruženoj podršci.

Dominik Mihalinec

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Središnje povjerenstvo za završne i diplomske ispite

Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

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

DIPLOMSKI ZADATAK

Student: **Dominik Mihalinec** Mat. br.: 0035173463

Naslov rada na hrvatskom jeziku: **Relativno vođenje robota koristeći 3D vizijski sustava**

Naslov rada na engleskom jeziku: **Relative guidance of a robot using 3D vision system**

Opis zadatka:

Nesavršenosti mehaničkih elemenata robota u odnosu na njegov nominalni model uzrokuju pogreške prilikom pozicioniranja robota u zadanu točku prostora. Za potrebe ispravljanja pogreške pozicioniranja robota moguće je koristiti vizijski sustav za praćenje te procjenom relativnih odstupanja ispraviti greške pozicioniranja. U okviru diplomskog rada potrebno je izraditi upravljački program koji će omogućiti relativno vođenje alata robotske ruke na temelju informacija izmjerenih pomoću medicinskog optičkog sustava (Polaris Vicra) za praćenje markera. Nadalje, potrebno izračunati greške obaju sustava (mean, max, STD, RMS) na temelju podataka pohranjenih kroz automatski proces mjerenja pozicija i orijentacija.

Programska podrška treba omogućiti sljedeće funkcionalnosti:

- preuzimanje informacija položaja i orijentacije 2 markera iz vizijskog sustava,
- transformaciju podatka i izračun matrice transformacija koristeći dodatnu C++ knjižnicu za linearnu algebru,
- upućivanje alata na kraju robotske ruke u određenu poziciju i orijentaciju unutar koordinatnog sustava jednog od markera na temelju ručnog unosa podataka,
- razmjenu podataka s robotom koristeći razvijenu "korisnik-poslužilac" programsku podršku,
- automatsko mjerenje pomaka robota i spremanje pozicija i orijentacija izmjerenih na vizijskom sustavu i na robotu.

Zadatak zadan:	Rok predaje rada:	Predviđeni datum obrane:
7. svibnja 2015.	9. srpnja 2015.	15., 16. i 17. srpnja 2015.

Zadatak zadao: Predsjednik Povjerenstva:

Prof. dr. sc. Bojan Jerbić Prof. dr. sc. Franjo Cajner

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

SAŽETAK

U ovom diplomskom radu obrađena je tehnika vođenja robotske ruke Universal Robot 5 (UR5) pomoću medicinskog optičkog sustava za praćenje markera Polaris Vicra i na taj način smanjena je njegova pogreška pozicioniranja.

Nakon uvodnog poglavlja o već postojećim robotskim sustavima u medicini, slijede poglavlja koja sadrže: princip rada optičkog sustava te kratak opis robotske ruke i njegovog kinematičkog modela. Postoje i potpoglavlja u kojima je objašnjena matematička terminologija i veličine koje ti sustavi koriste.

Kao etalon za potrebe mjerenja korištena je kalibracijska ploča. Na početku praktičnog dijela diplomskog rada izvršena su mjerenja kalibracijske ploče na optičkom koordinatnom stroju za mjerenje (CMM) kako bi definirali međusobne udaljenosti rupa na ploči. **Kasnije su svi drugi rezultati mjerenja uspoređivani sa rezultatima dobivenima na CMM stroju.** Sljedeći korak bio je ručno mjerenje kalibracijske ploče sa optičkim sustavom u svrhu izračunavanja njegove pogreške mjerenja. Izračunate su pogreške mjerenja za:

- položaj samo jednog markera u prostoru;
- dva razmaknuta markera u prostoru tj. mjerenje položaja jednog u odnosu na drugi.

Ispitano je također utječe li orijentacija markera na točnost optičkog sustava.

U svrhu ispitivanja točnosti pozicioniranja robota, sonda sa markerom pričvršćena je u robotovoj prihvatnici. **Ručnim vođenjem robota**, vrh sonde pozicioniran je u jednakoj orijentaciji u svaku rupu na kalibracijskoj ploči i očitavane su paralelno vrijednsti sa robota i sa Polaris-a (obzirom na referentni marker).

U okviru diplomskog zadatka izrađen je upravljački program koji omogućuje relativno vođenje alata robotske ruke na temelju informacija izmjerenih pomoću medicinskog optičkog sustava. Program je napisan u programskom jeziku C++. Kao IDE koristio se Visual Studio i dodatna C++ knjižnica *Eigen*. Programska podrška omogućuje sve funkcionalnosti koje su zadane u diplomskom zadatku. Cijela programska podrška, uključujući komunikaciju između robota i PC-a, matematički algoritam, razmjenu i tok informacija, objašnjena je u petom poglavlju. U prilogu na kraju diplomskog rada priložene su sve korištene Exel tablice i programski kod koji je korišten.

Ključne riječi: optički sustav za praćenje, Polaris Vicra, vizijski sustav, Universal Robot 5, vođenje robota, pogreška pozicioniranja.

SUMMARY

This master's thesis describes how to guide a robotic arm ″Universal Robot (UR5)″ by using the medical OTS (Optical Tracking System) Polaris Vicra, in order to reduce Positioning Errors.

The introductory chapter, which is about already existing robotic systems in medicine, is followed by chapters on the operating principle of optical system and a short description of the robotic arm with its related kinematic model. There are also subsections that describe mathematical terminology and mathematical structures used by the OTS and the UR5.

The calibration board was used as etalon for measurements. At the beginning of the practical part of this master's thesis, the calibration board was measured on the optical coordinate measuring machine, so that the distances between holes can be determined. The acquired distances were used as referent values. **Later, all other measurement results were compared to the results obtained by the CMM.** Next step was to manually measure the calibration board with the optical system in order to calculate OTS measurement errors. The errors in measurement were calculated for:

- the position and orientation of only one marker;
- OTS returns the data of the marker probe in the second marker's coordinate system.

Marker probe was attached to the robot's end effector. **By manually guiding the robot**, the tip of the marker probe was centred in each hole on the calibration board in the same orientation and data was collected from the robot and the Polaris for each hole. The acquired data from the robot and from the Polaris were compared to the data from the CMM.

Within this master's thesis, a controlling program has been made that provides a relative guidance of the robotic arm tool by using the data measured by the OTS. The Program is written in the C++ programming language. The software support provides all functionality specified by a given task. Comprehensive description of the entire software support, including the communication protocol between the robot and the PC, the mathematical algorithm, the exchange of information and information flow is provided in the fifth chapter. Excel tables and used programming codes are provided in the attachment at the end of this master's thesis.

Key words: optical tracking system, Polaris Vicra, vision system, Universal Robot 5, robotic guidance, positioning errors.

1. UVOD

Skraćenice CAS (eng. Computer - assisted surgery) ili CIS (eng. Computer - integrated surgery) predstavljaju kirurški koncept i skup metoda koje koriste računalne tehnologije u svrhu planiranja operacijskih zahvata, kirurške navigacije, bolje vizualizacije i asistencije kirurgu za vrijeme operacije u smislu vođenja i usmjeravanja. Stereotaksija je tehnika koja se odnosi na preciznu lokalizaciju određenog područja na temelju stereotaksijskih okvira i instrumenata. Prednost te tehnike je to što male promjene mogu biti lokalizirane i uklonjene iz područja koja su prethodno bila kirurški nedostupna, uz minimalnu traumu susjednog tkiva. Nakon što se stereotaksijski uređaj stavi na glavu bolesnika, područje u mozgu koje će se operirati, određuje se pomoću X, Y, i Z koordinata vlastitog koordinatnog sustava. Postoji i neuronavigacijska tehnika, ona koristi optički navigacijski sustav koji omogućava precizno praćenje tro-dimenzionalnog položaja i orijentacije kirurškog instrumenata, u odnosu na intrakranijalnu (lubanja) ili intraspinalnu (kralježnica) patološku promjenu, korištenjem računala koje se spaja sa snimkama CT-a ili MR-a. Neuronavigacija se koristi softverom za obradu slike i orijentacijskim točkama na glavi ili kralježnici koje se nakon toga uspoređuju s prostornom projekcijom patološke promjene, s ciljem povezivanja koordinatnih sustava [1].

U ovom diplomskom radu bit će ispitana preciznost stereo-vizijskog sustava za praćenje Polaris Vicra i obzorom na to, preciznost pozicioniranja i orijentacija kirurškog alata pomoću niskobuđetne industrijske robotske ruke (Universal robot UR5). Ideja je da robotska ruka poput alata služi kirurgu da kvalitetnije i jednostavnije obavi operaciju na pacijentu.

CAS je područje iz kojeg se razvila robotski potpomognuta kirurgija tzv. RAS (eng. Robotically - assisted surgery). Posljednjih dvadesetak godina postepeno se razvijaju robotski sustavi primjenjivi u kirurgiji. U ovom odjeljku su navedeni poznatiji primjeri, koji se koriste u neurokirurgiji i kirurgiji općenito. Prvi kirurški robot u komercijalnoj upotrebi bio je ROBODOC, 1992. godine [2]. Razvoj ROBODOC projekta trajao je 11 godina i koristi se u ortopedskim operacijama zglobova i ugradnje umjetnog kuka. ROBODOC je zapravo SCARA tip robotske ruke sa pet stupnja slobode gibanja, sa dodatnim alatima i sigurnosnim karakteristikama[3]. Robotska ruka Rosa ima šest stupnjeva slobode gibanja i koristi se u neurokirurgiji za minimalno invazivni pristup lokacijama u mozgu, što uključuje i umetanje

elektroda ili sondi. Zahvati se obavljaju bez stereotaksijskih okvira, sa povećanom točnošću i smanjenim vremenom operiranja. Sustav Rosa je u kliničkoj uporabi u Europi, Sjevernoj Americi, Kanadi i Aziji [4]. NeuroMate [5] je robotski sistem sa pet stupnjeva slobode gibanja koji se također koristi u neurokirurgiji za umetanje eletrkoda, dubinsku moždanu stimulaciju (DBS), dubinsko snimanje moždanih signala (SEEG), biopsiju mozga i uklanjanje tumora. Može se koristiti sa stereotaksijskim okvirom ili bez njega. Neuro-robotski sustav PathFinder je pokretni sustav sa vrlo stabilnom bazom, koji se vrlo lako može pomicati iz jedne u drugu operacijsku salu. Kako bi locirao područje na kojem se vrši operacija, PathFinder koristi optički sustav integriran u distalnoj ruci i reflektirajuće markere pričvršćene na pacijentovoj glavi [6]. Motor Assisted Robotic Stereotaxy (MARS) sistem je posebno osmišljen stereotaksijski robot koji je vrlo sličan ručno podesivom stereotaksijskom okviru. Sastoji se od pet stupnjeva slobode gibanja, tri translacije i dvije rotacije. Postoji još dodatna translacijska os po kojoj sonda ulazi u pacijenta. Glavna prednost ovog sustava je njegova kompaktnost, tj. velićina i jednostavno postavljanje u operacijskoj sali. Lokalizacija kirurškog instrumenta se izvodi na principu statičnog magnetskog polja. MARS robot je lagan i ne zauzima mnogo mjesta u operacijskoj sali, no zbog njegove deformabilne konstrukcije njegova preciznost može biti smanjena [4]. Postoje i mikro-stereotaksijski okviri. Ti okviri su izravno pričvršćeni na lubanju pacijenta i imaju zadovoljavajuću točnost vođenja instrumenta zbog usko povezane mehaničke konstrukcije. Okviri su izvedeni u obliku Stewart-Gough platforme [7]. Glavni nedostatak mikro-stereotaksijskih okvira je to što imaju vrlo ograničen radni volumen, što ih ograničava samo na specifične operativne zahvate.

Robotski sustavi u medicini postaju sve učestalija praksa u klinikama širom svijeta. Stoga, od velike je važnosti da oni budu pouzdani i da im cijena bude prihvatljiva. Razvoj i relativno mala proizvodnja takvih sustava mogla bi biti dugotrajna i preskupa. S ciljem zaobilaženja tih problema, tim ljudi sa FSB-a u suradnji sa KBD (Klinička Bolnica Dubrava) i HIIM (Hrvatski Institut za Istraživanje Mozga) razvili su neurokirurški robotski sustav RONNA koji koristi dvije robotske ruke komercijanle namjene [4]. Prva robotska ruka ima šest stupnja slobode gibanja (KUKA KR6). Služi kao alternativa uobičajenoj stereotaksijskoj metodi zbog svoje čvrstoće i krutosti. Koristi se za precizno navođenje kirurških instrumenata prema točno određenoj lokaciji. Druga robotska ruka (KUKA LWR 4+) sa sedam stupnjeva slobode gibanja, ima svojstvo „popuštanja“. Opremljena je sa senzorima momenta, koji omogućuju mekano interaktivno ponašanje i prilagođavaju moment reaktivnih motora putem upravljanja impedancijom. Promjena u momentu motora praćena je visokom frekvencijom (1-

3 kHz) i interpretira se pripadnim kontrolerom. Ruka je programirana tako da spriječi mogućnost ozljeđivanja čovjeka. RONNA sustav ima poseban sistem upravljanja baziran na evolucijskim algoritmima razvijen da koordinira kretnje obje kobotske ruke, tako da spriječava neželjene kolizije [8].

Slika 1. RONNA, umetanje katetera [9]

1.1. Robotski sustavi u medicini

Kako bi se shvatile potencijalne vrijednosti robotike u kirurgiji, važno je razumijeti ključne razlike između čovjeka i robota (Tablica 1. Razlike između čovjeka i robota [8]). Robot koristi velik broj detaljnih, kvantitativnih informacija na temelju kojih obavlja precizne i ponovljive pokrete, ali ima vrlo ograničenu sposobnost donošenja odluka i kvalitativnu sposobnost procjenjivanja. S druge strane, ljudi su spretniji i imaju dobru koordinaciju očiju i ruku i najvažnije, „mekan“ osjet dodira.

Sadašnja uloga robota u kirurgiji je da asistira kirurgu u operaciji u smislu da proširi ili pojača ljudske sposobnosti. Robotski sustavi omogućuju obavljanje zahvata na vrlo maloj skali (mikro-kirurgija), omogućuju pristup lokaciji kroz vrlo ograničen prostor (minimalno

invazivan zahvat), imaju vrlo precizne pokrete koji mogu biti ponovljivi. Autori članka [8] navode prednosti robotskih kirurških sustava u odnosu na CAS:

- Veća točnost i programabilnost kompliciranе putanje u 3D prostoru.
- Ponavljanje identičnih pokreta u određenom vremenskom periodu.
- Sposobnost držanja alata na precizno definiranoj lokaciji tijekom duljeg vremena, bez podrhtavanja i umora.
- Pomicanje instrumenta je strogo ograničeno na specifičnu putanju ili lokaciju, čak i pod utjecajem vanjske sile, što spriječava ozljedu vitalnih organa.
- Brzo reagiranje i prilagodba na informacije sa senzora ili korisničke komande.

Tablica 1. Razlike između čovjeka i robota [8]

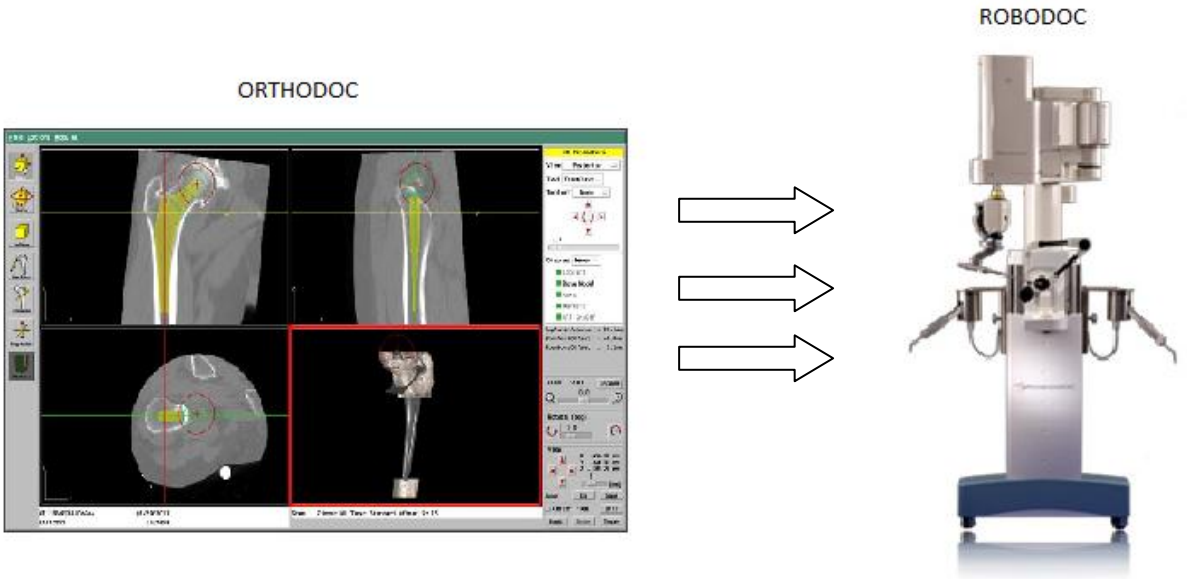
ČOVJEK	ROBOT
Prednosti:	
Dobra koordinacija očiju i ruku.	Dobra geometrijska točnost.
Spretnost (na ograničenoj mjernoj skali).	Stabilnost, ponovljivost, bez umora.
Fleksibilnost i prilagodljivost.	Potencijalna primjena u mikro i nano kirurgiji
Korištenje kvalitativnih informacija.	Prima velik broj različitih informacija.
Vrlo dobro procjenjivanje.	Koristi različite senzore u upravljanju.
Ograničenja:	
Smanjena spretnost u neprirodnim položajima	Smanjena spretnost i koordinacija.
Sklonost umoru i podrhtavanju ruke.	Nemogućnost donošenja samostalnih odluka.
Mogućnost pogreške.	Ograničeni su na relativno jednostavne zadatke.
Sterilni uvjeti nisu uvijek idealni.	Potreba za većim radnim prostorom, cijena.

Točnost kirurških robota može se odnostiti na više stvari, obično se dijele u tri različite kategorije [8]:

- Unutarnja točnost robota (npr. prosječna pogreška određene komponente, mehanička popustljivost, trenje, labavost...).
- Točnost obrade slika (rezolucija, šum, oblik i materijal markera, numeričke pogreške…).
- Točnost aplikacije u realnim uvjetima (uvjeti u operacijskoj sali, nelinearnosti, promjenjiva okolina, ispituje se kliničkim ispitavanjem).

ROBODOC

Robotski sustav ROBODOC koristi se kod oblikovanja i zamjene kuka, te kod operacije koljena. ROBODOC zajedno sa Orthodoc (radna stanica za planiranje), koristi CT slike pacijenta prije operacije kako bi pomoću tih podataka, u određenom softveru modelirao virtualni 3D model kosti ili zgloba. Pomoću 3D modela, kirurg može neposredno prije operacije precizno definirati specifične točke na kosti, veličinu i poziciju proteze, anatomiju zgloba i gustoću kosti. ROBODOC obrađuje kost alatom, bez izravne kontrole kirurga tijekom izvođenja. Stoga točnost aplikacije u takvim (realnim) uvjetima je jedan od najvažnijih faktora. Unutarnja točnost robota je 0.5 mm, dok prosječna točnost u realnim uvjetima iznosi 1.2 mm, a varira u rasponu od 1.0 mm do 3.5 mm [10].



Slika 2. Robotski sustav ROBODOC [10]

NeuroMate

Već spomenuti NeuroMate je stereotaksijski neurokirurški sustav sa pet stupnjeva slobode gibanja. Koristi se najčešće kod biopsije i uklanjanja tumora. Korištenjem slika pacijenta prije operacije, omogućuje vizualizaciju mjesta operiranja u realnom vremenu, kako bi kirurg precizno lokalizirao tumor.

Unutarnja točnost je definirana na 0.75 mm sa ponovljivošću od 0.15 mm, dok je srednja vrijednost točnosti u realnim uvjetima veća i iznosi 0.95 mm, sa standardnom deviacijom od 0.44 mm [10].

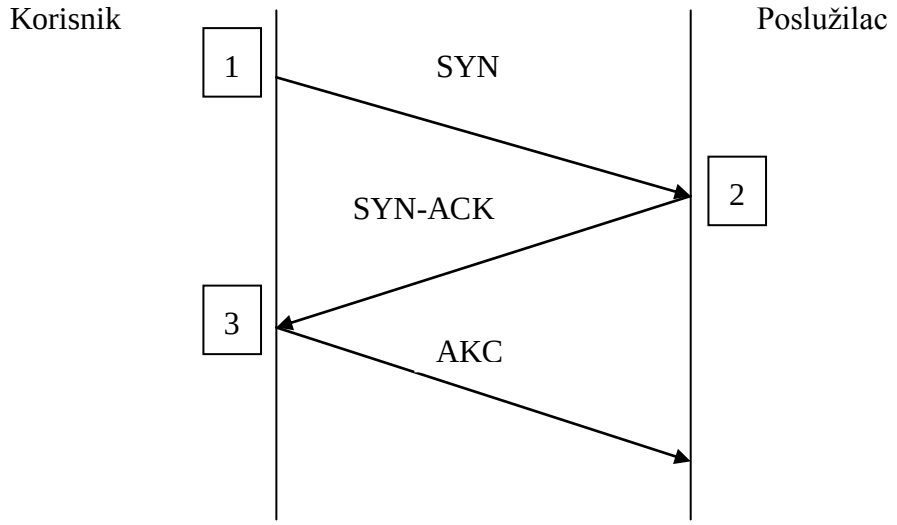
Slika 3. NeuroMate[10]

1.2. Korištena oprema

Glavne komponente robotskog sustava u ovom diplomskom radu su: Universal Robot UR5 – robotska ruka sa šest stupnjeva slobode gibanja, medicinski optički sustav za praćenje OTS (Optical Tracking Systems) Polaris Vicra sa dva markera od kojih svaki ima četiri reflektirajuće kuglice na sebi i računalo (PC). Od opreme za mjerenje korištena je koordinatna mjerna mašina CMM u Nacionalnom laboratoriju za duljinu na FSB-u i kalibracijska ploča. Kalibracijska ploča je dimenzija 250x250 mm i ima 8x8 rupa pozicionirane tako da su međusobno okomite. Rupe u kalibracijskoj ploči napravljene su tako da centriraju konusni vrh sonde kada ju umetnemo. Za potrebe aplikacije korištena je čelična čahura sa rupom promjera D=3.25 mm.

1.3. Komunikacija - „korisnik-poslužitelj“

Tcp (eng. Transmission Control Protocol) jedan je od osnovnih protokola unutar IP grupe protokola. Korištenjem protokola TCP aplikacija na nekom od hostova umreženog u računalnu mrežu kreira virtualnu konekciju prema drugom hostu, te putem ostvarene konekcije zatim prenosi podatke. U ovom diplomskom radu TCP konekcija korištena je na lokalnoj razini. TCP garantira pouzdanu isporuku podataka u kontroliranom redosljedu od pošiljatelja prema primatelju. TCP protokol u svom sastavu sadrži opise portova, odnosno brojčane vrijednosti temeljem kojih računalo po prihvatu podataka zna koju uslužnu programsku potporu mora aktivirati te na koji način razmjenjivati podatke. Prednost je i to što TCP ima kontrolu isporuke i kontrolu greške (traži ponavljanje izgubljenog ili neispravnog paketa podataka) i zadužen je za početnu uspostavu komunikacije između računala, dok je npr. UDP (User Datagram Protocol) znatno jednostavniji i nema kontrolu razmjene podataka [11]. TCP za uspostavljanje komunikacije najčešće koristi tzv. protokol trostrukog rukovanja (eng. Three way Handshake), slika 4.



Slika 4. Three way Handshake dijagram

Korak 1: „Korisnik“ želi uspostaviti komunikaciju sa „poslužiteljem“ ili serverom.“ Korisnik“ šalje sinkronizirajuću poruku SYN (segment sa SYN slijednim brojem), kojom obavijesti drugu stranu da želi s njim komunicirati. Slijedni broj je broj od kojeg „korisnik“ počinje

označavati slijed svojih segmenata tj. broj od kojeg počinje brojati segmente koje šalje, a koriste se kako bi se sačuvao redoslijed podataka.

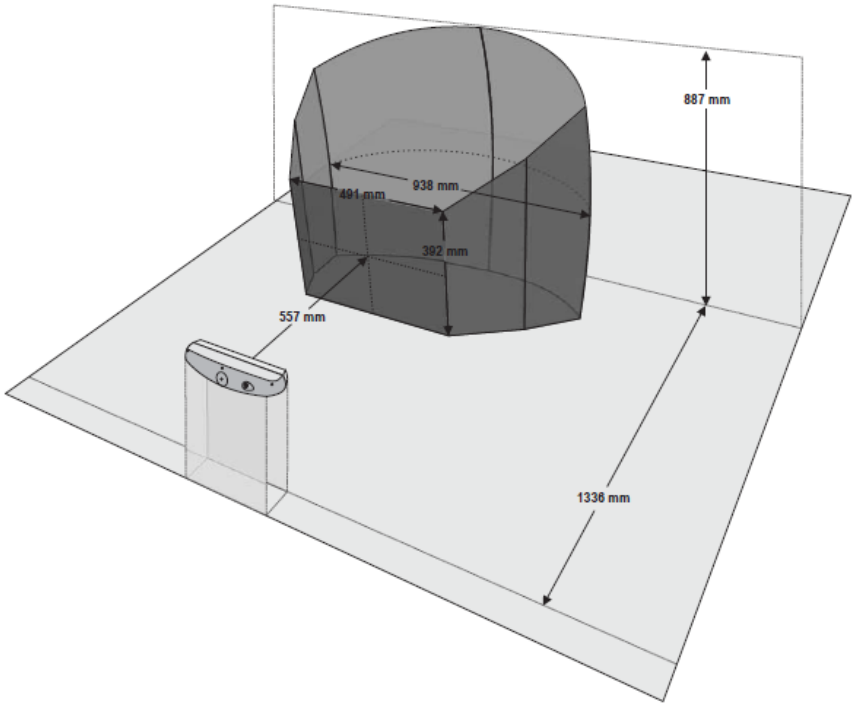
Korak 2: „Poslužitelj“ odgovara sa segmentom koji sadrži potvrdu prijema i svoj slijedni broj (ACK bit i SYN bit). Na taj način „poslužitelj“ potvrđuje prijem i kaže „korisniku“ od kojeg će broja on početi brojati svoje segmente.

Korak 3: Na kraju „korisnik“ pošalje segment kojim potvrđuje prijem segmenta od „poslužitelja“ i šalje svoje podatke.

TCP upotrebljava određen raspon portova kojima razdjeljuje programe na strani pošiljatelja i primatelja. Port omogućuje da dvije istorodne aplikacije ili servisi mogu interpretirati podatke koji su pristigli u računalo, a IP adresa definira tko je te podatke poslao i tko ih treba primiti. Svaka TCP konekcija ima dodijeljenu 16-bitnu oznaku na obje strane aplikacije (slanje, primanje). Portovi su u osnovi podijeljeni u 3 kategorije: poznati portovi, registrirani portovi i dinamički portovi. Par Port-IP naziva se Socket, spojne točke u komunikaciji.[11]

2. Polaris Vicra

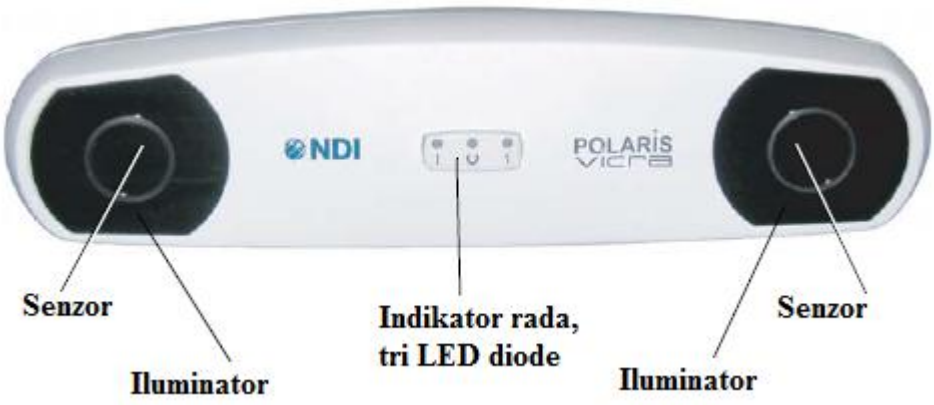
Polaris Vicra je optički mjerni sistem, tvrtke Northern Digital Inc., koji vidi retro-reflektirajuće sferne kuglice. Te kuglice se mogu koristiti pojedinačno, tada sustav mjeri samo poziciju pojedine kuglice. Kuglice mogu biti pričvršćene na specifičan alat u 3D prostoru, tada sustav može mjeriti poziciju i orijentaciju tog tijela (mora biti pričvršćeno minimalno tri kuglice). Koristeći se tim podacima, mogu se dobiti pozicije i orijentacije alata u specifičnom mjernom volumenu koji je prikazan na sljedećoj slici.



Slika 5. Trodimenzionalni prikaz mjernog volumena Polaris Vicra sistema[12]

Takav optički sustav za praćenje koristi se u raznim područjima uključujući medicinu, zubnu industriju i na području razovja i istraživanja.

Najvažnija komponenta Polaris Vicre sustava je senzor pozicije. Senzor pozicije emitira infra-crvenu (IR) svjetlost kroz tzv. iluminatore ili osvjetljivače, slično flash-u na uobičajenim kamerama. IR svjetlost obasjava mjernu okolinu i odbija se od sfernih markera natrag do senzora. Senzor pozicije tada odredi poziciju markera i izračuna poziciju i orijentaciju alata na koji markeri pričvršćeni. Senzor te podatke, zajedno sa informacijama o stanju sustava šalje na računalo kako bi se podaci pohranili, prikazali ili dalje obrađivali [12].



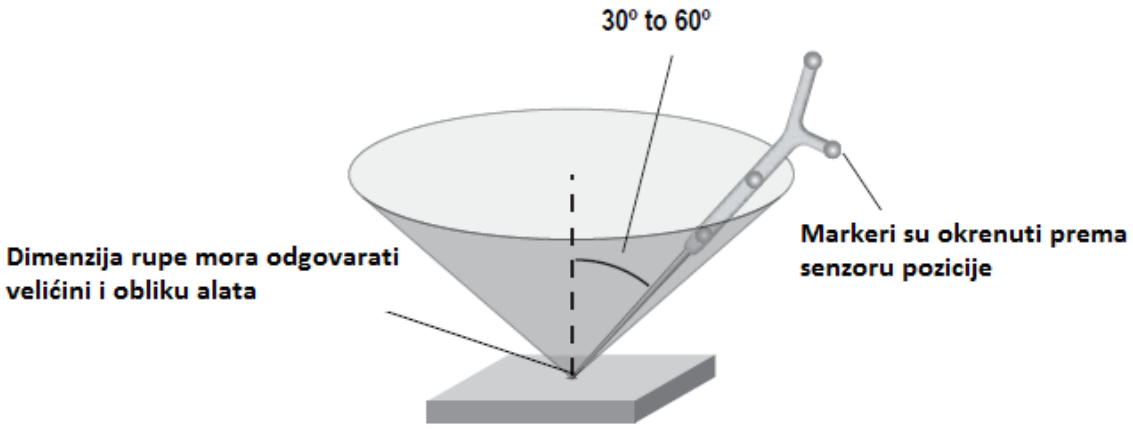
Slika 6. Polaris Vicra, pogled sprijeda [12]

Iluminatori zapravo predstavljaju dvostruki red dioda (IREDs) koji emitiraju IR svjetlost prema markerima. Senzori sadrže leću i CCD čipove za detektiranje vraćene svjetlosti.

Slika 7. Sonda sa sfernim markerima [8]

Objekt na slici ima čvrstu strukturu tako da je relativani odnos između markera nepromjenjiv. Geometriju markera, potrebno je definirati na način da se unese „tool definition file“ u korištenu aplikaciju. Ukupno se 15 takvih file-ova može unijeti u aplikaciju. Sustav je u mogućnosti istovremeno pratiti šest različitih pasivnih alata i jedan aktivni alat (sam odašilje IR svjetlost) uz ograničenje: ukupan broj markera ne smije biti veći od 32 [12].

Tracking Tools aplikacija ima mogućnost pivotiranja alata. To znači da aplikacija izračunava poziciju vrha špice sonde umjesto pozicije koordinatnog sustava cijelog alata. Pivotiranje je prikazano na sljedećoj slici, a izvodi se na način da se vrh sonde vrti oko rupe promjera oko 1mm, u trajanju od 20 sekundi. Kut nagiba u odnosu na vertikalu mora cijelo vrijeme biti između 30 i 60 stupnjeva [12].



Slika 8. Tehnika pivotiranja alata [12]

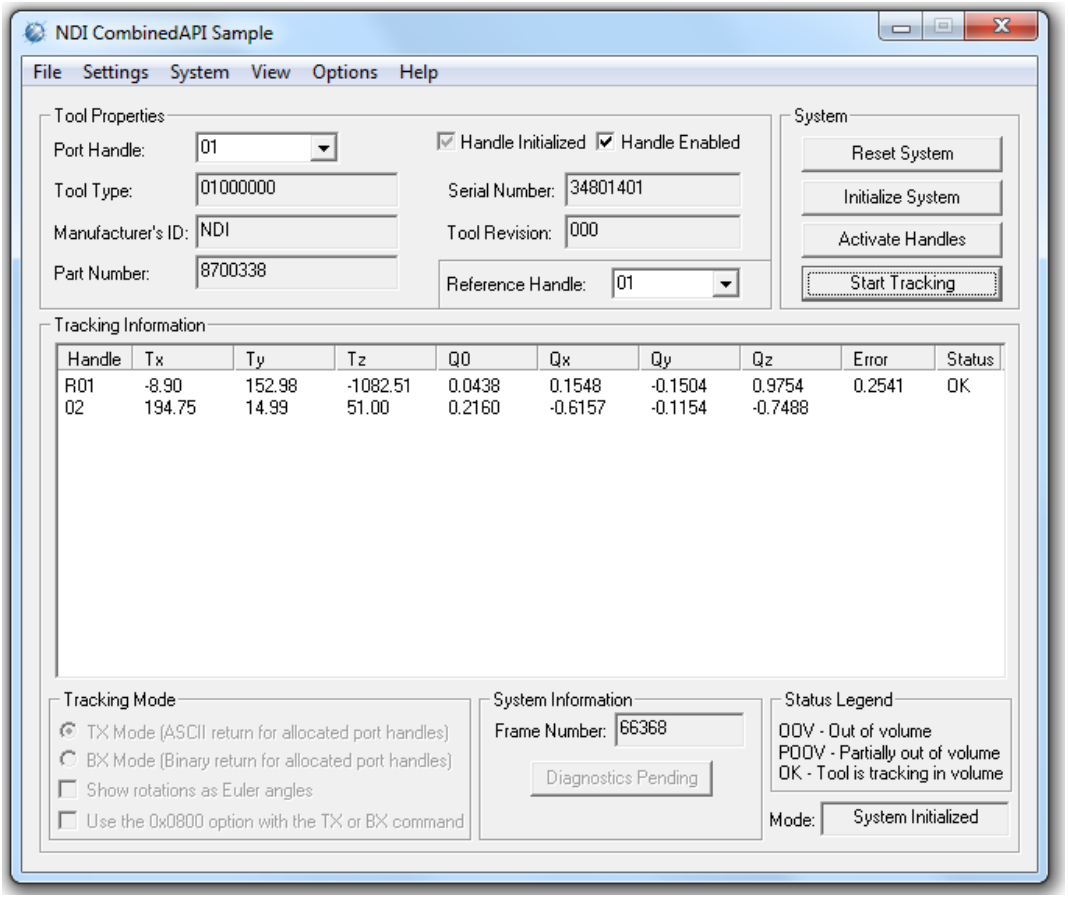
U Tablici 2. prikazani su dobiveni rezulati pivotiranja koji su korišteni u daljnjem radu:

Tablica 2. Rezultati pivotiranja alata

X: -19.93	Y: -1.23	Z: -157.10	RMS Pogreška: 0.29
Broj uzetih uzoraka: 425			Maks. Pogreška: 0.61
			Min. Kut: 61.58

2.1. Polaris API

Polaris API je aplikacija napisana u C++ programskom jeziku. Pomoću nje dobivamo podatke o položaju i orijentaciji dvaju alata u odnosu na globalni koordinatni sustav koji se nalazi u samom Polarisu. Isto tako, može se podesiti da Polaris daje podatke za jedan alat u odnosu na lokalni koordinatni sustav koji se onda nalazi u drugom alatu. Što znači da imamo relativne podatke o jednom markeru u odnosu na drugi. Ta forma će se najviše koristiti u ovom diplomskom radu.

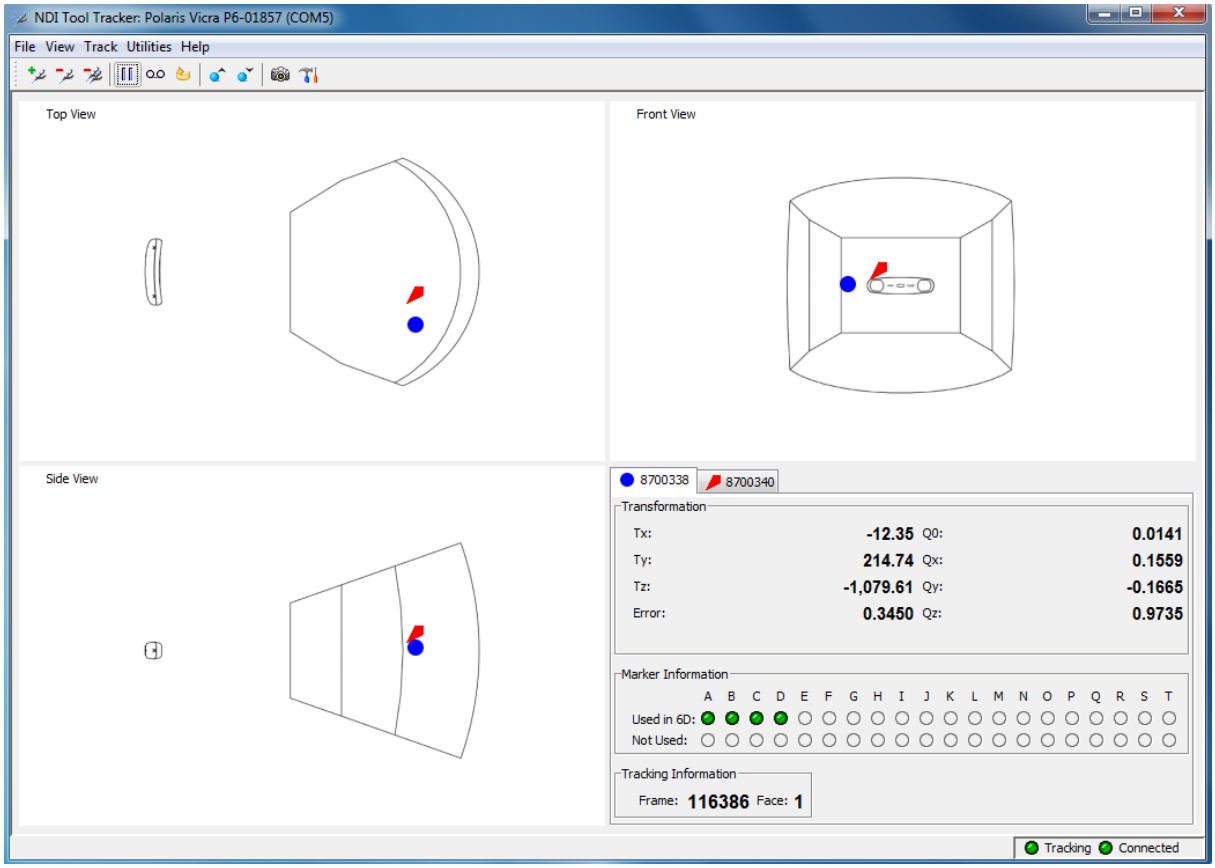


Slika 9. Polaris API

Na slici 9. nalazi se GUI aplikacija. Orijentacija alata može biti prikazana u obliku Eulerovih kutova ili u kvaternionima. U ovom diplomskom radu koristit će se zapis u kvaternionima. Više o kvaternionima nalazi se u poglavlju 2.3.

2.2. NDI Tool Tracker: Polaris Vicra

NDI Tool Tracker je program kojim se vizualno mogu pratiti markeri u radnom prostoru (volumenu) Polaris Vicre. Na početku korištenja potrebno je učitati u program definiranu geometriju markera (tool definition file) koji se koristi u aplikaciji. Pomoću ovog programa izvršava se potrebno pivotiranje alata.



Slika 10. NDI Tool Tracker: Polaris Vicra

Na slici 10. prikazan je program za praćenje na kojem su vidljiva dva markera iz tri različita pogleda unutar radnog volumena. Vidljivi su još i podaci o trenutnim položajima i orijentacijama markera u odnosu na Polarisov koordinatni sustav, a isto tako je moguće dobiti podatke o relativnom položaju i orijentaciji za jedan marker u odnosu na poziciju drugog markera. U donjem dijelu dobiva se informacija o vidljivosti pojedine reflekritajuće kuglice na markerima.

2.3. Kvaternioni

U ovom poglavlju dana je teorijska podloga za lakše razumijevanje matematičkih veličina kvaterniona. Kvaternion predstavlja matematičku veličinu koja se koristi za računanje i opisivanje prostornih transformacija. Za matematičke proračune sa kvaternionima u Visual Studiu korištena je C++ biblioteka Eigen (poglavlje 4.3.), pored toga korišten je još i programski paket MatLab.

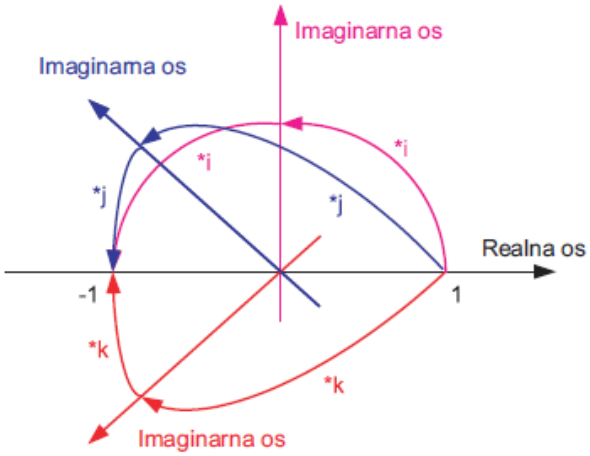
Kvaternion je broj koji ima četiri dimenzije, jednu realnu i tri imaginarne. Iz tog razloga, čovjeku je teško na neki fizički način interpretirati četverodimenzionalni prostor. Svaka imaginarna dimenzija ima svoju vrijednost za $\sqrt{-1}$ što može biti i, j ili k [13]. Veličina q predstavlja kvaternion:

$$q = a + ib + jc + kd \tag{1}$$

Postoje pravila koja određuju i opisuju imaginarne dimenzije:

$$\begin{aligned} i^2 = j^2 = k^2 = -1 \\ i \cdot j = -j \cdot i = k \\ j \cdot k = -k \cdot j = i \\ k \cdot i = -i \cdot k = j \end{aligned} \tag{2}$$

Djeluje zbunjujuće što imamo tri različite vrijednosti za $\sqrt{-1}$, ali razlog tome je što se mi nalazimo u četverodimenzionalnom prostoru što znači da postoje barem tri različita puta koja će nas odvesti iz točke 1 do točke -1. Na sljedećoj slici grafički su predočena ova tri puta.



Slika 11. Tri različita puta od točke 1 do točke -1 [13]

Prava svrha kvaterniona različita je od svrhe kompleksnih brojeva. Npr. pomoću komplesnog broja vektorska se veličina može prikazati u dvodimenzionalnom prostoru dok pomoću kvaterniona prikazujemo rotaciju u trodimenzionalnom prostoru. Dakle kvaternion je matematička veličina koja predstavlja rotaciju [13]. Možda bi najbolji način za njegovo predstavljanje bio sljedeći zapis:

$$Q = \cos\left(\frac{\alpha}{2}\right) + i\left[x \cdot \sin\left(\frac{\alpha}{2}\right)\right] + j\left[y \cdot \sin\left(\frac{\alpha}{2}\right)\right] + k\left[z \cdot \sin\left(\frac{\alpha}{2}\right)\right] \tag{3}$$

gdje je:

α - kut rotacije

x, y, z - komponente jediničnog vektora koje predstavljaju osi rotacije

Različite literature koriste različite zapise kvaterniona. U ovom radu se najviše koristio vektorski zapis:

$$Q = \begin{bmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\alpha}{2}\right) \\ x \cdot \sin\left(\frac{\alpha}{2}\right) \\ y \cdot \sin\left(\frac{\alpha}{2}\right) \\ z \cdot \sin\left(\frac{\alpha}{2}\right) \end{bmatrix} \tag{4}$$

Za razliku od Eulerovih kutova jedan kvaternion zbog svoje četverodimenzionalnosti može predstavljati i osnovnu i složenu rotaciju [13]. Kvaternion se poistovjećuje sa matricom rotacije (3x3). Vrlo bitna prednost kvaterniona je u tome što kvaternioni na egzaktan način izbjegavaju tzv. blokadu kardana (eng. Gimbal lock).

Gimbal lock je problem poznat kao blokada jednog od stupnja slobode kardanskog prostora uslijed međusobnog poravnavanja rotacijskih osi. Što znači da sustav gubi jednu od osi slobode. Matematičko objašnjenje problema je sljedeće: kada se jedna os poklopi sa drugom pod kutom od 90° ili $\pm\frac{\pi}{2}$, u Eulerovom prostoru taj se položaj definira kao kvocijent kosinusa kuta, koji u tom slučaju iznosi 0. Dakle dolazi do dijeljenja sa nulom, te bi rezultat trebao biti beskonačan (odnosno, dijeljenje s nulom nije definirano). A to fizikalno znači gubitak jedne dimenzije [14].

Još jedna prednost kvaterniona je to što se izbjegava veliki proračun i skraćuje vrijeme računanja. Algebra s kvaternionima je puno jednostavnija nego algebra s Eulerovim kutovima. Ovo se prvenstveno odnosi na množenje matrica.[13]

- **Normiranje kvaterniona**

Prije prebacivanja kvaterniona u matricu, treba se izvršiti normiranje kvaterniona ukoliko on nije normiran. Neka je $Q = q_0 + i \cdot q_x + j \cdot q_y + k \cdot q_z$. Normiranje kvaterniona Q vrši se tako da se prvo izračuna *norma* (oznaka n), pa se svi članovi kvaterniona podijele sa tom normom [14]:

$$n = \sqrt{q_0^2 + q_x^2 + q_y^2 + q_z^2} \tag{5}$$

$$Q_n = \frac{q_0}{n} + i \cdot \frac{q_x}{n} + j \cdot \frac{q_y}{n} + k \cdot \frac{q_z}{n} \tag{6}$$

Norma se još naziva i veličina kvaterniona. Normirani kvaternion još se naziva i jediniični kvaternion. Za normirani kvaternion vrijedi sljedeće svojstvo [14]:

$$q_0^2 + q_x^2 + q_y^2 + q_z^2 = 1 \tag{7}$$

- **Prebacivanje kvaterniona u matricu rotacije**

Za potrebe daljnjeg računanja sa kvaternionima, oni se moraju prebaciti u matricu rotacije (DCM matricu). Ako je kvaternion zadan u obliku (4), onda se prebacivanje može vršiti prema sljedećim formulama [15] :

$$\boldsymbol{R_Q} = \begin{bmatrix} 1 - 2 \cdot q_y^2 - 2 \cdot q_z^2 & 2 \cdot q_x \cdot q_y - 2 \cdot q_z \cdot q_0 & 2 \cdot q_x \cdot q_z + 2 \cdot q_y \cdot q_0 \\ 2 \cdot q_x \cdot q_y + 2 \cdot q_z \cdot q_0 & 1 - 2 \cdot q_x^2 - 2 \cdot q_z^2 & 2 \cdot q_y \cdot q_z + 2 \cdot q_x \cdot q_0 \\ 2 \cdot q_x \cdot q_z + 2 \cdot q_y \cdot q_0 & 2 \cdot q_y \cdot q_z + 2 \cdot q_x \cdot q_0 & 1 - 2 \cdot q_x^2 - 2 \cdot q_y^2 \end{bmatrix} \tag{8}$$

3. Universal Robot- UR5

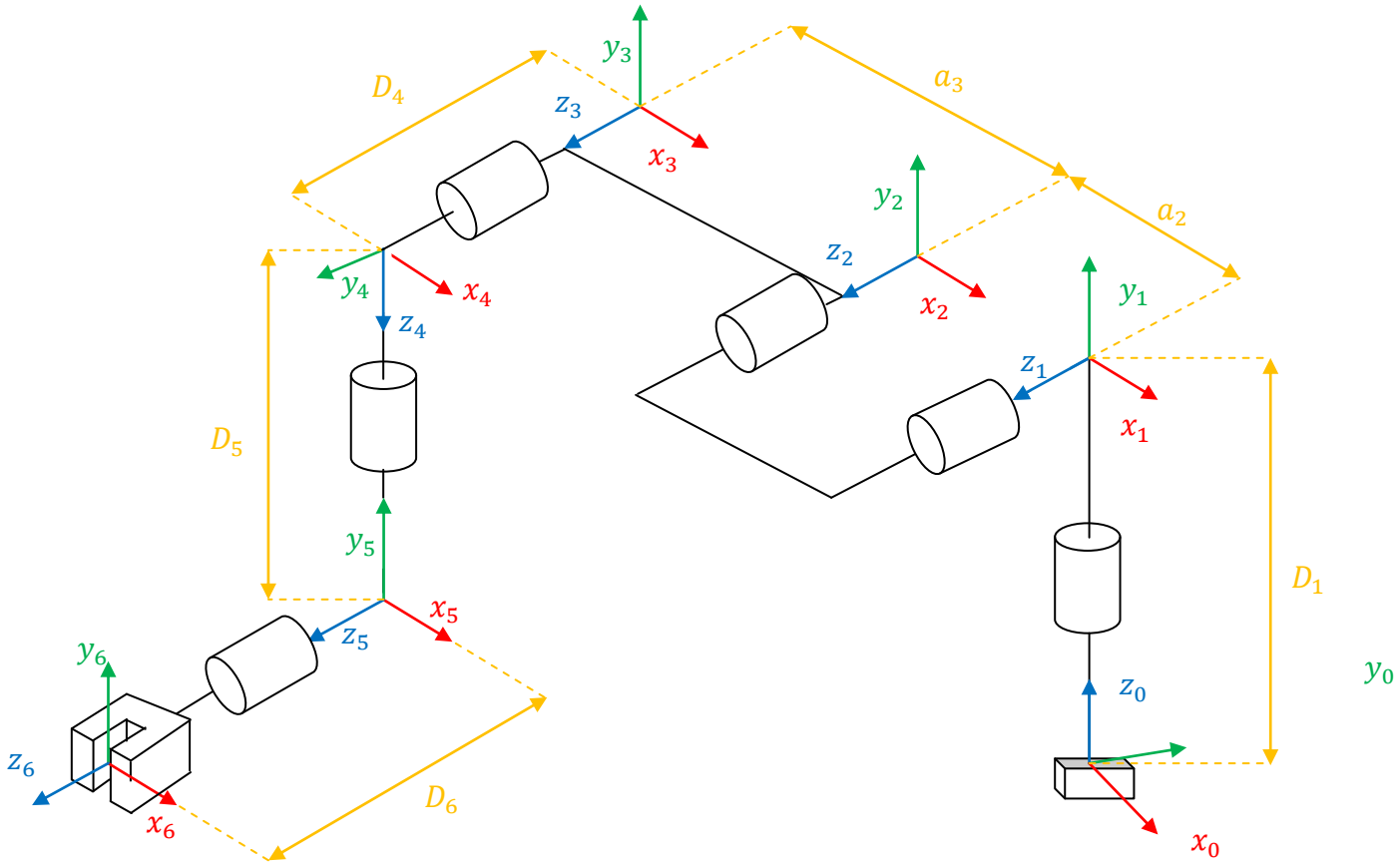
Slika 12. UR5 [16]

UR5 je robotska ruka sa šest stupnjeva slobode gibanja opće namjene i moguće ju je koristi za automatizaciju procesa u raznim područjima. UR5 ima Linux OS platformu i moguće ga je programirati direktno preko tableta. Robot je opremljen sa digitalnim i analognim ulazima i izlazima i ima mogućnost komunikacije sa vanjskim uređajem putem Ethernet kabla [16]. Putem Ethernet kabla i korištenjem TCP komunikacijskog protokola, u ovom diplomskom radu napravljen je kompleksniji sustav upravljanja robotom preko PC-a. Razmjena podataka izvršava se korištenjem „korisnik-poslužilac“ (eng. server-client) programske podrške.

Kinematički model robota izveden je u Denavit-Hartenberg (DH) notaciji. Transformacija iz „base“ robotovog koordinatnog sustava u koordinatni sustav prihvatnice alata **„**tool**“** izračunat je množenjem matrica. Greške pozicioniranja javljaju se zbog mehaničkih nesavršenosti u pojedninim segmentima robotske ruke, tj. zbog odstupanja od nominalnih parametara. Greška pozicioniranja mogla bi se smanjiti re-kalibracijom i optimizacijom MDH parametara [4]. Međutim u ovom radu cilj je istražiti točnost robota sa njegovim originalnim kinematičkim modelom i tu točnost povećati koristeći optički sustav za praćenje Polaris Vicra. DH parametri robota su prikazani u Tablici 3.

Tablica 3. DH parametri za UR5[4]

Zglobovi na robotu	Parametri			
	θ[rad]	a[m]	D[m]	α[rad]
J1	0	0	0.089159	π/2
J2	0	-0.42500	0	0
J3	0	-0.39225	0	0
J4	0	0	0.10915	π/2
J5	0	0	0.09465	- π/2
J6	0	0	0.08230	0



Slika 13. Kinematički model UR5 robota

3.1. Angle-Axis notacija

Za računanje orijentacije robot koristi podatke zapisane u obliku trodimenzionalnog vektora, tzv. vektor rotacije:

$$\boldsymbol{V_{rot}} = [R_x \quad R_y \quad R_z\,] \tag{9}$$

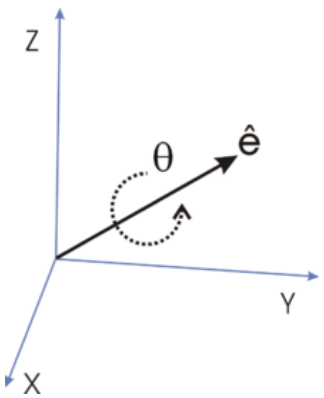
R_x, R_y i R_z predstavljaju kutove rotacije oko određene osi, u radijanima. Ti podaci nam kasnije trebaju za ispravno kalibriranje alata na robotu. Kako bi se došlo do potrebne matrice rotacije s kojom dalje možemo računati, potrebno je objasniti Angle-Axis zapis. To je način interpretiranja rotacije koja je definirana kombinacijom jediničnog vektora $\hat{e}$ i kuta rotacije θ:

$$\boldsymbol{V_A} = [e_x \quad e_y \quad e_z \quad \theta\,] \tag{10}$$

$$\theta = \sqrt{{R_x}^2 + {R_y}^2 + {R_z}^2} \tag{11}$$

$$e_x = \frac{R_x}{\theta};\ e_y = \frac{R_y}{\theta};\ e_z = \frac{R_z}{\theta}; \tag{12}$$

Axis-angle notacija je ekvivalentna vektoru rotacije koji se ponekad naziva i Eulerov vektor. Takav zapis jednostavno se može vizualizirati i prikazan je na sljedećoj slici:



Slika 14. Angle-Axis inerpretacija rotacije [17]

Nakon što se definira rotacija u Angle-Axis zapisu, jednostavno se može pomoću sljedećih formula izračunati matrica rotacije $\boldsymbol{R}$ (DCM matrica) [15]:

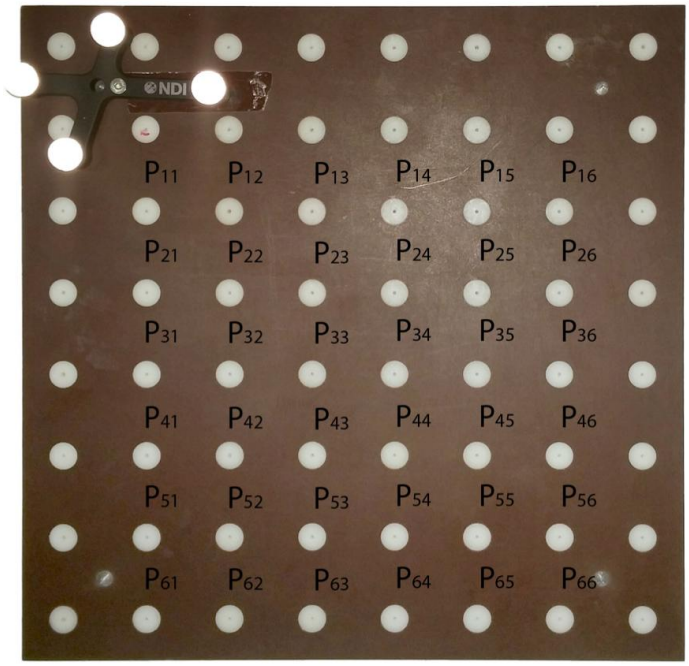
$$\boldsymbol{R} = \begin{bmatrix} C * \boldsymbol{e_x^2} + \boldsymbol{\cos}\,\theta & C * \boldsymbol{e_x} * \boldsymbol{e_y} - \boldsymbol{e_z} * \boldsymbol{\sin}\,\theta & C * \boldsymbol{e_x} * \boldsymbol{e_z} + \boldsymbol{e_y} * \boldsymbol{\sin}\,\theta \\ C * \boldsymbol{e_x} * \boldsymbol{e_y} + \boldsymbol{e_z} * \boldsymbol{\sin}\,\theta & C * \boldsymbol{e_y^2} + \boldsymbol{\cos}\,\theta & C * \boldsymbol{e_y} * \boldsymbol{e_z} - \boldsymbol{e_x} * \boldsymbol{\sin}\,\theta \\ C * \boldsymbol{e_x} * \boldsymbol{e_z} - \boldsymbol{e_y} * \boldsymbol{\sin}\,\theta & C * \boldsymbol{e_y} * \boldsymbol{e_z} + \boldsymbol{e_x} * \boldsymbol{\sin}\,\theta & C * \boldsymbol{e_z^2} + \boldsymbol{\cos}\,\theta \end{bmatrix} \tag{13}$$

$$C = 1 - \boldsymbol{\cos}\,\theta \tag{14}$$

U svrhu računanja navedenih transformacija i prebacivanja iz jednog zapisa u drugi, korišten je programski alat MatLab. Korišteni kod i vrijednosti navedene su u poglavlju 5.1.1.

4. Proces mjerenja točnosti Polaris Vicra i točnosti pozicioniranja robota UR5

Kako bismo izmjerili točnost vizijskog sustava Polaris Vicra, potrebna nam je neka vrsta etalona koja je izmjerena na sustavu koji je za barem jedan red točniji od Polarisa. Kao etalon korištena je kalibracijska ploča, prethodno izmjerena na optičkom koordinatnom stroju za mjerenje (CMM).



Slika 15. Kalibracijska ploča sa označenim mjernim točkama i pričvršćenim referentnim markerima [4]

U procesu mjerenja, za svaku točku uzeti su podaci iz koordinatnog sustava mjernog uređaja, pa su potom transformirani u koordinatni sustav kalibracijske ploče. Koordinatni sustav kalibracijske ploče je formiran koristeći točke P_{11}, P_{16} i P_{61}. Tako dobivena transformacija opisuje poziciju i orijentaciju kalibracijske ploče u bilo kojem drugom koordinatnom sustavu **X**. Matrica rotacije $\boldsymbol{{}^{X}_{C}R}$ dobije se iz tri normalizirana vektora koja se dobiju iz navedenih točaka P_{11}, P_{16} i P_{61}. A pozicijski vektor $\boldsymbol{{}^{X}_{C}P}$ definiran je položajem (x, y i z) točke P_{11} [4] :

$$ {}^{X}_{C}\boldsymbol{P} = \begin{bmatrix} P_{11_x} \\ P_{11_y} \\ P_{11_z} \end{bmatrix} \tag{15} $$

$$ \vec{\boldsymbol{x}} = \|\boldsymbol{P_{61}} - \boldsymbol{P_{11}}\| \tag{16} $$

$$ \vec{\boldsymbol{z}} = \left\|\vec{\boldsymbol{x}} \times \|\boldsymbol{P_{61}} - \boldsymbol{P_{11}}\|\right\| \tag{17} $$

$$ \vec{\boldsymbol{y}} = \|\vec{\boldsymbol{z}} \times \vec{\boldsymbol{x}}\| \tag{18} $$

$$ {}^{X}_{C}\boldsymbol{R} = \begin{bmatrix} \overrightarrow{\boldsymbol{x_x}} & \overrightarrow{\boldsymbol{y_x}} & \overrightarrow{\boldsymbol{z_x}} \\ \overrightarrow{\boldsymbol{x_y}} & \overrightarrow{\boldsymbol{y_y}} & \overrightarrow{\boldsymbol{z_y}} \\ \overrightarrow{\boldsymbol{x_z}} & \overrightarrow{\boldsymbol{y_z}} & \overrightarrow{\boldsymbol{z_z}} \end{bmatrix} \tag{19} $$

Po tri podatka je uzeto za svaku točku (x, y i z os). Množenjem tog vektora sa homogenom matricom transformacije $ {}^{X}_{C}\boldsymbol{T} $ dobije se pozicija točke u koordinatnom sustavu ploče.

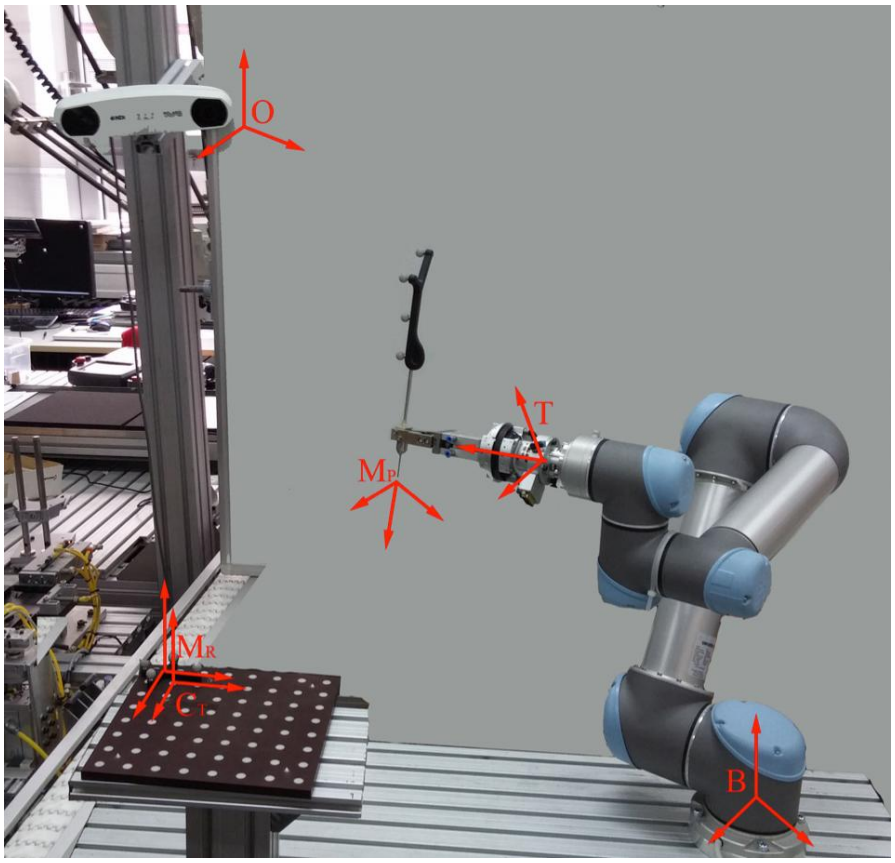
Matrica transformacije opisuje poziciju i orijentaciju jednog koordinatnog sustava u odnosu na drugi koordinatni sustav i sastoji se od matrice rotacije $\boldsymbol{R}$, dimenzija 3x3 i 3D pozicijskog vektora vektora $\boldsymbol{P}$ [4]:

$$ {}^{B}_{C}\boldsymbol{T} = \begin{bmatrix} {}^{B}_{C}\boldsymbol{R} & {}^{B}_{C}\boldsymbol{P} \\ 0 & 1 \end{bmatrix} \tag{20} $$

Sljedeći koordinatni sustavi korišteni su za navigiranje robotske ruke i mjerenja:

- koordinatni sustav kalibracijske ploče **C** definiran je u jednoj od vanjskih rupa za centriranje na kalibracijskoj ploči;
- robotov bazni koordinatni sustav **B** definiran je unutar robota;
- robotov koordinatni sustav alata **T** opisan je u **B** sa transformacijom $ {}^{B}_{T}\boldsymbol{T} $ i trenutnom pozicijom i orijentacijom koje ovise o pozicijama robotovih zglobova;
- OTS-ov koordinatni sustav **O** pozicioniran je sa strane na stalku i namješten tako da OTS nesmetano prati markere;

- koordinatni sustav sonde sa markerima $\boldsymbol{M}_P$, pomaknut za „tool tip offset“, što zapravo određuje vrh sonde, fiksiran je na robotovoj prihvatnici i može biti opisan transformacijom ${}^{T}_{\boldsymbol{M}_p}\boldsymbol{T}$;
- Koordinatni sustav $\boldsymbol{M}_{\boldsymbol{R}}$ referentnog markera pričvršćen je na kalibracijskoj ploči;



Slika 16. Sustav mjerenja sa ucrtanim koordinatnim sustavima [4]

Pozicijska pogreška računala se kao apsolutna razlika u mjerenju duž svake (x,y,z) osi i kao Euklidska udaljenost između točaka. Euklidska udaljenost računa se po sljedećoj formuli [4]:

$$E_{ij} = \sqrt{\left(P_{ijx} - P'_{ijx}\right)^2 + \left(P_{ijy} - P'_{ijy}\right)^2 + \left(P_{ijz} - P'_{ijz}\right)^2} \tag{21}$$

gdje P_{ij} označava udaljenost u Kartezijskom koordinatnom sustavu jedne te iste točke izmjerene sa dva različita uređaja.

Za sva mjerenja izračunato je: srednja pogreška, maksimalna pogreška, srednja kvadratna pogreška (RMS), standardna devijacija, interval pouzdanosti 95% (CI 95%).

Za skup od *n* točaka standardna devijacija STD računa se prema formuli [4]:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{22}$$

STD je statistički pojam koji označava mjeru raspršenosti podataka u skupu. Računa se kao prosječno odstupanje od prosjeka i to u apsolutnom iznosu.

Interval pouzdanosti:

$$IC = 1.96 \cdot \frac{\sigma}{\sqrt{n}} \tag{23}$$

Za skup od *n* točaka RMS pogreška računa se prema formuli [4]:

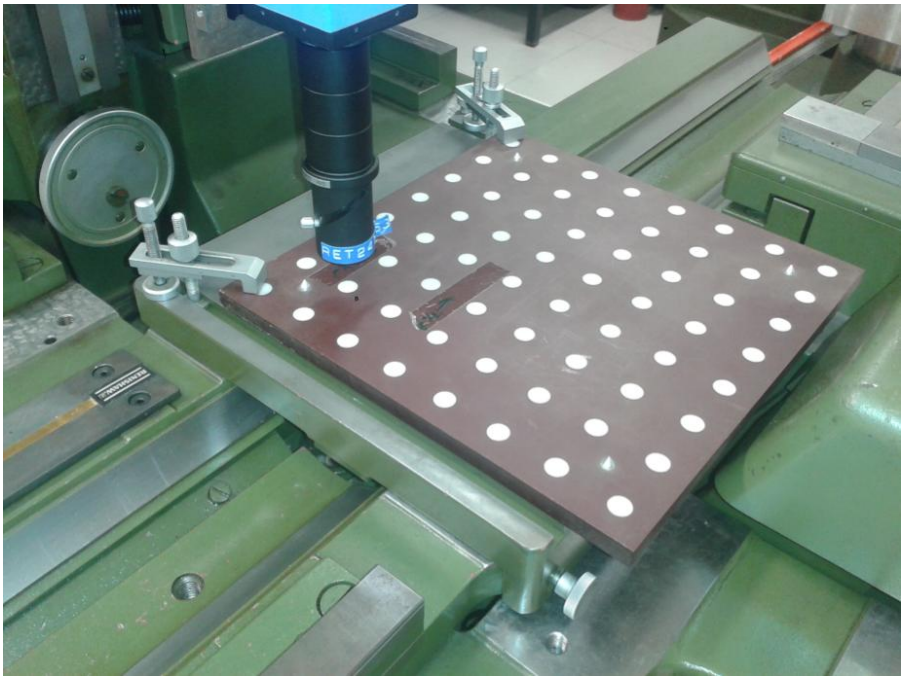
$$x_{rms} = \sqrt{\frac{x_1^2 + x_2^2 + x_3^2 + \cdots + x_n^2}{n}} \tag{24}$$

Vrijednosti dobivene mjerenjem ploče na optičkom CMM stroju biti će uspoređene sa:

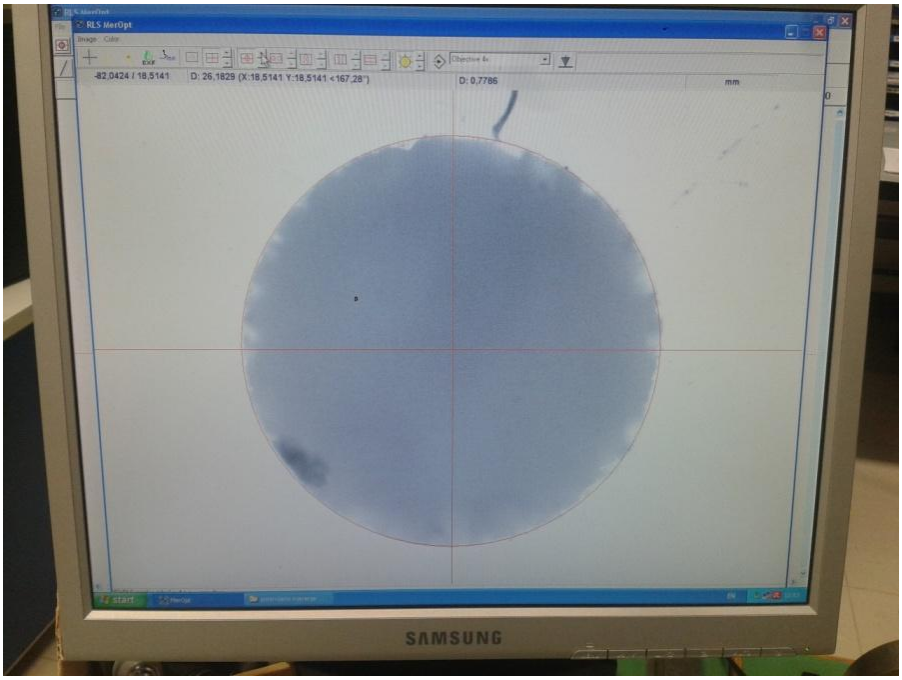
- vrijednostima dobivenim optičkim sustavom za praćenje Polaris,
- vrijednostima dobivenim iz robotovog koordinatnog sustava.

4.1. Mjerenje na optičkom CMM stroju

Mjerna nesigurnost optičkog CMM stroja iznosi: $0.2 + 4 * L$ izraženo u [µm], gdje je L izražen u metrima [4]. Mjerenja su izvršena u Laboratoriju za precizna mjerenja dužina na FSB-u.



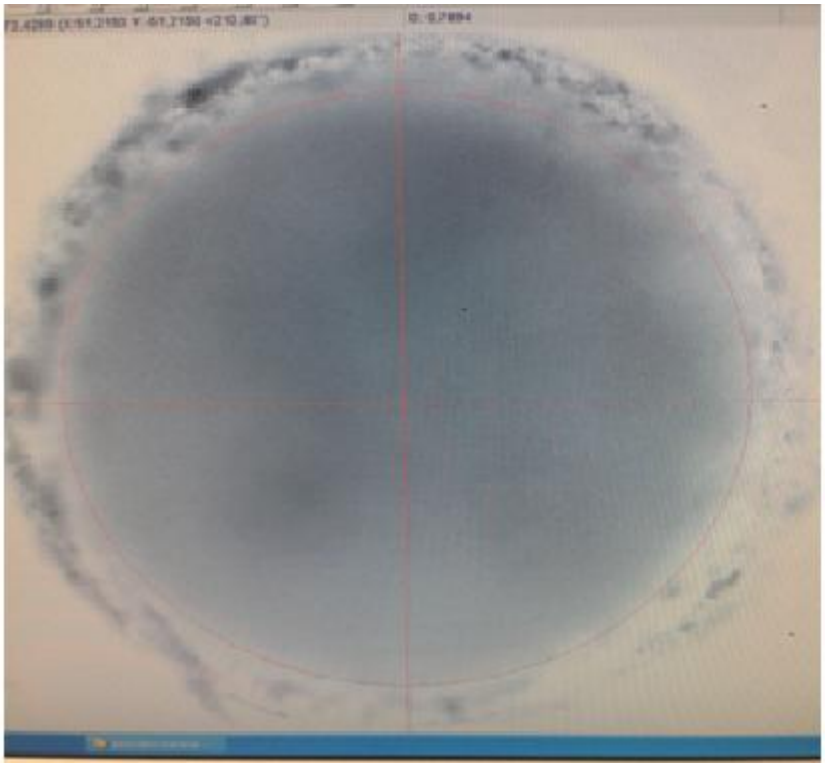
Slika 17. Mjerenje ploče na CMM stroju



Slika 18. Precizno određivanje središta rupe na ploči

Za mjerenje je odabrano unutarnjih 6x6 točaka čija dijagonalna udaljenost iznosi 212.1 mm. Na slici 17. prikazana je kalibracijska ploča pričvršćena na postolju CMM stroja. Postupak mjerenja je bio sljedeći: prvo se u programu odredilo ishodište koje se u ovom slučaju nalazi u točki P_{11}, zatim je bilo potrebno spremiti koordinate od svih preostalih 35 točaka. Kako bi se dobile udaljenosti točaka od ishodišta, podatke je trebalo ubaciti u programski alat Catia i očitati udaljenosti od definiranog ishodišta za svaku točku. Mjerenja na stroju i cijeli postupak bili su ponovljeni još dva puta. Potom je u Excelu izračunata prosječna udaljenost od ishodišta do svake točke posebno. Na taj način su dobivene udaljenosti od središta svake točke od ishodišta u X-Y ravnini.

S obzirom da se središte svake od 36 rupa određuje vizualno na računalu, to središte nije moguće sasvim idealno odrediti. Na slici 19. može se vidjeti kako rub svake rupe nije savršeno oštar, pa je teško savršeno odrediti središte. Ovdje se govori o pogrešci na mikrometarskoj razini. Dakle, na određenoj razini ovdje je prisutna i pogreška čovjeka koji izvodi ta mjerenja.



Slika 19. Precizno određivanje središta rupe na kalibracijskoj ploči

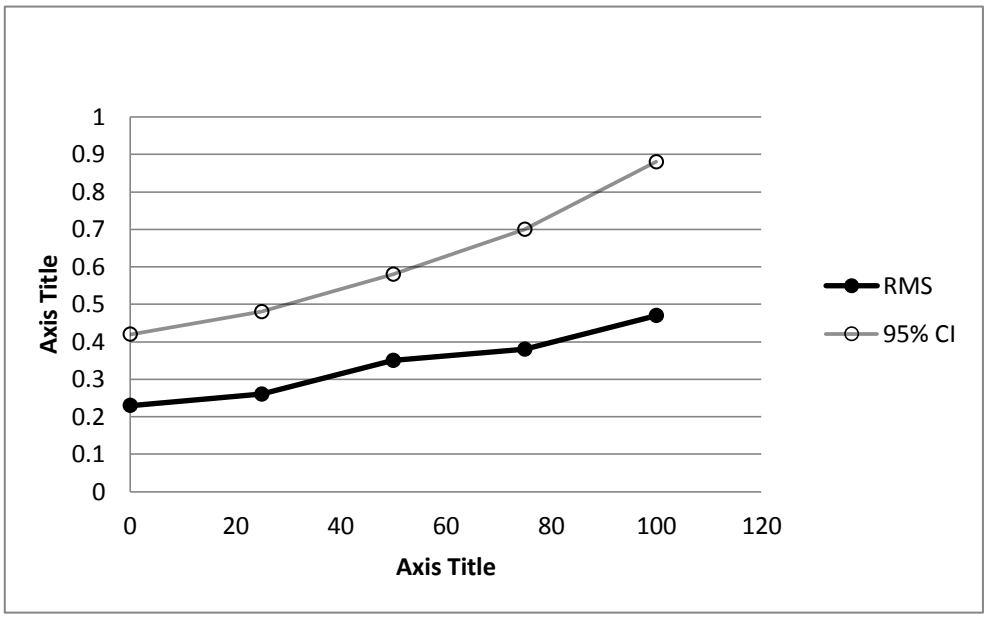
Tablica 4. Srednje vrijednosti udaljenosti točaka od ishodišta- 3 mjerenja na CMM stroju.

		x	y
1. red	T(1,1)	0	0
	T(1,2)	29.99423333	-0.015266667
	T(1,3)	60.00193333	0.030633333
	T(1,4)	89.9713	0.011533333
	T(1,5)	119.9476	0.031733333
	T(1,6)	149.9465	0.006766667
2. red	T(2,1)	-0.027166667	29.94443333
	T(2,2)	29.9585	29.95023333
	T(2,3)	59.9712	29.95036667
	T(2,4)	89.97736667	29.9545
	T(2,5)	119.9740667	29.97153333
	T(2,6)	149.9426333	29.9699
3. red	T(3,1)	-0.01985	59.88243333
	T(3,2)	29.98473333	59.88273333
	T(3,3)	59.98575	59.88748333
	T(3,4)	89.95721667	59.89708333
	T(3,5)	119.9893	59.89971667
	T(3,6)	149.9907333	59.89536667
4. red	T(4,1)	-0.027833333	89.82195
	T(4,2)	29.98585	89.819
	T(4,3)	59.97415	89.82355
	T(4,4)	89.98286667	89.82366667
	T(4,5)	120.0001167	89.8242
	T(4,6)	149.99545	89.83501667
5. red	T(5,1)	0.01405	119.7426333
	T(5,2)	29.99603333	119.7411
	T(5,3)	60.0052	119.8628
	T(5,4)	90.02076667	119.7524
	T(5,5)	120.0174	119.7536667
	T(5,6)	150.0157	119.7531333
6. red	T(6,1)	0.021066667	149.7151667
	T(6,2)	30.01953333	149.6718667
	T(6,3)	60.02153333	149.7015667
	T(6,4)	90.01893333	149.7015667
	T(6,5)	120.0226333	149.6964
	T(6,6)	150.0195667	149.6801333

Tablica 4 prikazuje izračunate srednje udaljenosti središta pojedinih točaka od ishodišta. Kako bi dobili što preciznije rezultate, ukupno su izvršena tri mjerenja ploče. Vidljivo je da se središta rupa ne poklapaju idealno sa osima definiranog koordinatnog sustava. Te vrijednosti su kasnije korištene u svhru izračunavanja pogrešaka optičkog sustava.

4.2. Ručno mjerenje kalibracijske ploče sa vizijskim sustavom Polaris Vicra

Optički sustav za praćenje Polaris Vicra ima od proizvođača deklariranu RMS pogrešku od 0.25 mm sa intervalom pouzdanosti (CI) od 95%. Deklarirana preciznost vrijedi kod praćenja samo jedne sferne kuglice. Mnogo bitnija točnost je ona kod koje optički sustav prati tri ili više sfernih kuglica, pa je pomoću njih moguće odrediti poziciju i orijentaciju markera [4]. U članku [18] zaključeno je da Polaris Vicra ima veću točnost kada prati tijelo sa četiri sferna markera, nego kad prati samo jedan sferni marker. U tom slučaju, izračunata pogreška ima vrijednost RMS Error=0.231 mm. Također su izvedena mjerenja sa pomaknutim vrhom alata i zaključak je bio da se RMS pogreška linearno povećava sa povećanjem udaljenosti vrha alata. Za to mjerenje koristila se jedna definicija alata, tj. tijelo je imalo četiri markera pozicionirana u jednakom međusobnom odnosu, a ishodište je bilo pomaknuto na 25, 50, 75 i 100 mm od originalnog mjesta u smjeru z-osi tijela. To je prikazano na sljedećoj slici.



Slika 20. Smanjivanje točnosti uslijed povećanja udaljenosti vrha alata.[18]

U ovom radu napravljena su mjerenja sa Polaris Vicrom na način da dobivamo podatke o poziciji i orijentaciji vrha sonde obzirom na Polarisov koordinatni sustav koji se nalazi u njemu i obzirom na referentni koordinatni sustav koji se nalazi u drugom tijelu sa četiri sferna markera koje je pričvršćeno na kalibracijskoj ploči.

U kirurškim aplikacijama korišteni alat uvijek je pomaknut od središta tijela markera i njegovih sfernih kuglica. Nastojeći dobiti što kvalitetnije rezultate u ovom radu se za mjerenje koristi sonda sa četiri sferna markera čiji je vrh alata pomaknut za oko 160 mm od originalnog ishodišta sfernih markera, slika 7. Mjerenja su provedena ručnim pomicanjem sonde po kalibracijskoj ploči, slika 21.

Slika 21. Ručno pomicanje sonde po kalibracijskoj ploči

Napravljena su dva uzastopna mjerenja pozicije za svaku od 36 točaka. Kako bi se dobili točniji rezultati, za svaku točku uzeto je pet vrijednosti sa Polaris Vicre i izračunala se srednja vrijednost za svaku točku. Potom su se u Excelu ti podaci usporedili sa onima izmjerenima na stroju (Tablica 4.) kako bi se izračunale potrebne pogreške i odstupanja. U procesu mjerenja sonda je ručno pomicana pomoću improviziranog držača i vrh sonde je svaki put centriran u rupi.

Prvo mjerenje napravljeno je na način da su markeri sonde cijelo vrijeme bili okrenuti prema senzoru pozicije na Polaris Vicri. Dok je kod drugog mjerenja orijentacija sonde u svakoj točki bila nasumično promijenjena u odnosu na orijentaciju u prethodnoj točki.

Tablica 5. Točnost pozicioniranja I. [4]

Vrsta pogreške	Pogreške pozicioniranja			
	x	y	z	Euklidska
Srednja vrijednost	0.292	0.121	0.038	0.407
Maks. vrijednost	0.568	0.557	0.094	0.797
STD	0.182	0.292	0.250	0.206
CI 95%[±]	0.059	0.095	0.082	0.067
RMS	0.344	0.270	0.048	0.439

Tablica 6. Točnost pozicioniranja II. [4]

Vrsta pogreške	Pogreške pozicioniranja			
	x	y	z	Euklidska
Srednja vrijednost	0.144	0.311	0.091	0.378
Maks. vrijednost	0.406	0.817	0.159	0.905
STD	0.107	0.225	0.039	0.217
CI 95%[±]	0.035	0.074	0.013	0.071
RMS	0.179	0.384	0.099	0.436

Na temelju podataka iz tablica 5 i 6 može se vidjeti da je točnost pozicioniranja sonde sa četiri sferna markera i sa pomaknutim ishodištem od 160 mm vrlo slična za oba mjerenja: RMS pogreška za prvo mjerenje iznosi 0.439 mm, a za drugo 0.436 mm. Standardne devijacije su 0.206 i 0.217 mm. Slični rezultati kod oba mjerenja pokazuju da promjena orijentacije sonde sa markerima ne utječe na pogrešku pozicioniranja niti ju povećava.

Za potrebe matematičkih transformacija i računanja, te zbog jednostavnosti korištena je Eigen C++ biblioteka koja je opisana u sljedećem podpoglavlju.

4.3. Eigen C++ biblioteka

Kako bi se podaci iz Polarisovog koordinatnog sustava transformirali, koštena je C++ open source Eigen „template“ biblioteka. Eigen biblioteka koristi se za linearnu algebru; računanje sa matricama, vektorima i povezanim algoritmima [19]. Biblioteka se jednostavno koristi, na način da ju je nakon preuzimanja na računalo samo potrebno uključiti u header glavnog programa. Za razvoj upravljačkog programa u ovom diplomskom radu koristio se Visual C++ IDE, pa je uz uključivanje u „header“ programa još bilo potrebno includati biblioteku unutar izbornika Propertise.

```
#include <iostream>
#include <Eigen/Dense>

using Eigen::MatrixXd;

int main()
{
  MatrixXd m(2,2);
  m(0,0) = 3;
  m(1,0) = 2.5;
  m(0,1) = -1;
  m(1,1) = m(1,0) + m(0,1);
  std::cout << m << std::endl;
}
```

Output:

```
  3  -1
2.5 1.5
```

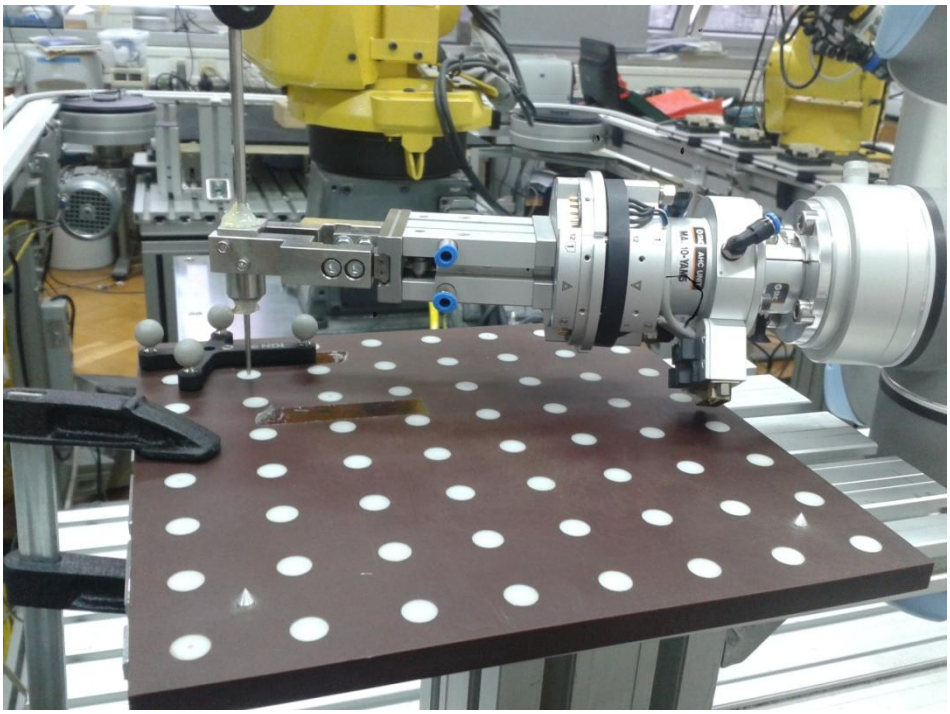
Slika 22. Jednostavan primjer korištenja Eigen biblioteke [19]

4.4. Mjerenje kalibracijske ploče, OTS - robot

Postupak ovog mjerenja sastoji se od paralelnog prikupljanja podataka sa OTS-a i sa robota. Sonda sa četiri reflektirajuća markera fiksirana je u robotovoj prihvatnici (Slika 16.). Ploča je držačima fiksirana za podlogu kako se ne bi pomaknula. Kako bi se prikupili podaci, ručnim vođenjem robota bilo je potrebno centrirati i pozicionirati vrh sonde u pojedinu rupu na kalibracijskoj ploči (Slika 23.).

Alternativna metoda vođenja robota pomoću vizijskog sustava je vođenje u odnosu na relativni (lokalni) koordinatni sustav. Takav način vođenja robota opisan je u ovom poglavlju. Ta se metoda inače koristi u slučaju kada objekt našeg interesa nije statičan, tj. postoji mogućnost da se pomakne. Npr. kod operacije u neurokirurgiji koristi se takva tehnika, na glavu pacijenta fiksira se referentni marker u slučaju da se glava pomakne. Dakle, kod ove tehnike, optički sustav istovremeno prati dva odvojena markera, od kojih je jedan pričvršćen na alat kojeg robot koristi, a drugi na objekt našeg interesa tj. u ovom slučaju na kalibracijsku ploču. Robot je za potrebe ovog mjerenja vođen obzirom na referentni marker lokaliziran u radnom prostoru, u prostoru mjerenja.

Optički sustav prati istovremeno oba markera i vraća na računalo poziciju i orijentaciju sonde markera u koordinatnom sustavu drugog markera $\boldsymbol{M_R}$, koji je fiksiran na rub kalibracijske ploče (Slika 16.). Sonda koju koristi robot prethodno je kalibrirana tako da dobivamo poziciju pomaknutu za „*Tool tip offset*“ od središta originalne geometrije markera, tj. dobivamo poziciju vrha sonde koja se nalazi u rupi.



Slika 23. Ručno pozicioniranje vrha sonde u rupe na kalibracijskoj ploči

Ciljevi ovog mjerenja su sljedeći:

- **procijeniti pogrešku vizijskog sustava za praćenje Polaris Vicra kada koristi dva markera na različitim udaljenostima;**
- **dati usporedbu sa pogreškama prijašnjih rezultata kada se koristio samo jedan marker za praćenje;**
- **izračunati unutarnju pogrešku pozicioniranja robota UR-5.**

Tablica 7. Vrijednosti pogrešaka [4]

Sustav	**Vrsta pogreške**	**Pozicijske pogreške**			
		x	**y**	**z**	**Euklidska**
Polaris mjeri samo jedan marker (sondu)	**Sred. vrijednost**	0.180	0.050	0.063	0.260
	Maks. vrijednost	0.757	0.330	0.396	0.758
	STD	0.152	0.116	0.100	0.141
	CI 95%[±]	0.029	0.022	0.019	0.027
	RMS	0.241	0.129	0.118	0.298
Polaris sa relativnim referenciranjem	**Sred. vrijednost**	0.214	0.078	0.101	0.348
	Maks. vrijednost	0.677	1.198	0.576	1.249
	STD	0.175	0.198	0.140	0.188
	CI 95%[±]	0.033	0.037	0.026	0.035
	RMS	0.280	0.220	0.176	0.398
Robot UR5	**Sred. vrijednost**	0.047	0.478	0.022	0.595
	Maks. vrijednost	0.310	1.172	0.928	1.268
	STD	0.097	0.329	0.310	0.302
	CI 95%[±]	0.018	0.062	0.059	0.057
	RMS	0.116	0.581	0.311	0.669

Mjerenja su izvršena ukupno tri puta, pa je ukupan broj uzoraka bio 108. Pogreške pozicioniranja u Tablici 7. izračunate su kao razlika između koordinata točaka izmjerenih na CMM stroju i:

- koordinata izmjerenih na robotu;
- koordinata izmjerenih Polarisom pomoću jednog markera;
- koordinata izmjerenih Polarisom pomoću dva markera.

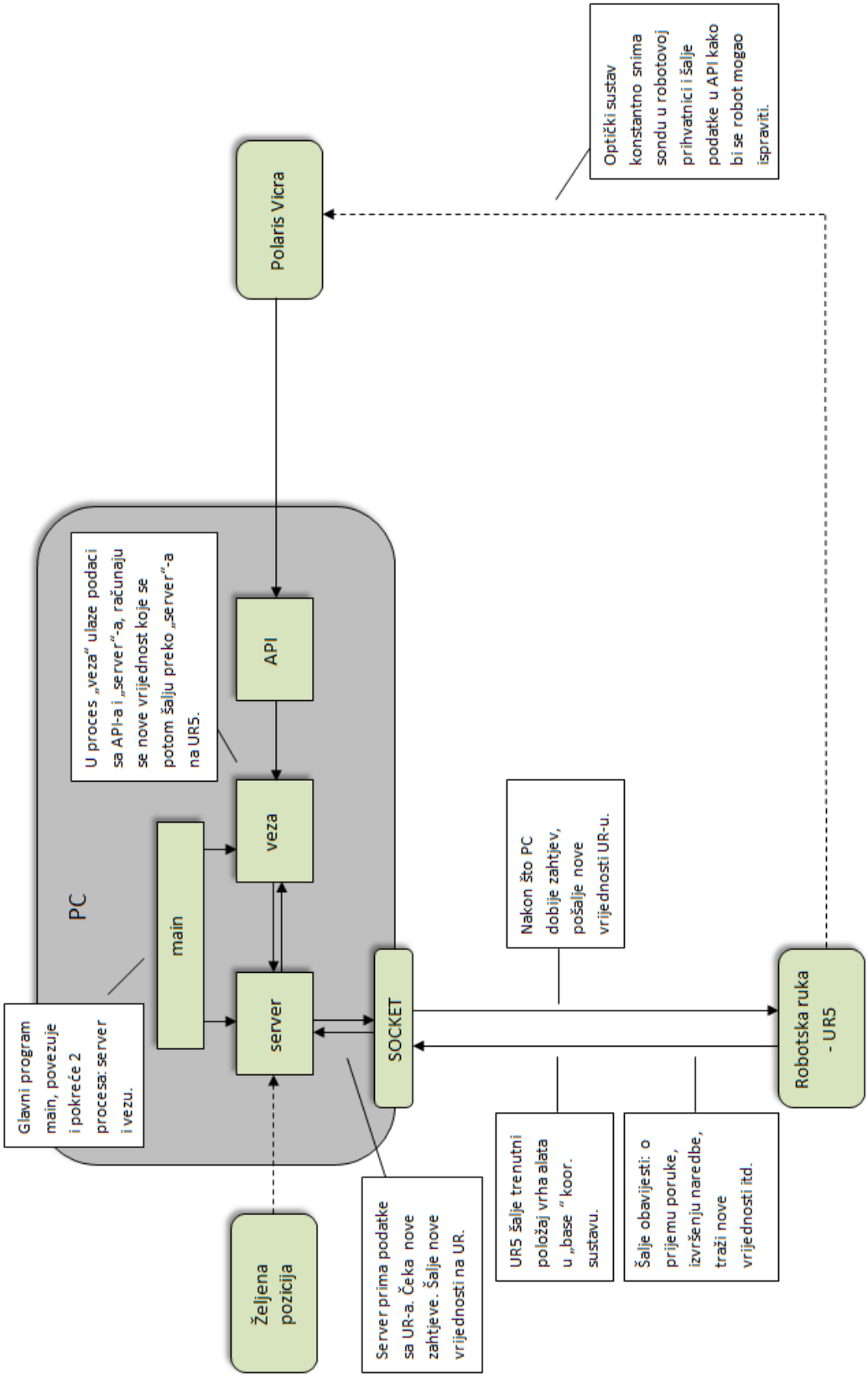
Mjerenja su pokazala sljedeće: ako vodimo robota sa optičkim sustavom za praćenje Polaris Vicra koristeći sondu tj. samo jedan marker, srednja RMS pogreška pozicioniranja se smanjuje sa 0.669 mm na 0.298 mm. U postocima to iznosi 55.4%. Kada se koristi relativno referenciranje i dva markera, srednja RMS pozicijska pogreška iznosi 0.398 mm, što je za 40.5% manje od robotove pogreške [4]. Važno je napomenuti da su se orijentacija alata i pozicija kalibracijske ploče tijekom mjerenja promijenile tri puta. Svaki skup od 36 točaka izmjeren je sa jednakom orijentacijom alata. Podaci koji se odnose na robota u Tablici 7., predstavljaju robotovu unutarnju točnost izmjerenu na području promjera $\mathrm{D} = 212.1\ \mathrm{mm}$, sa jednakom orijentacijom alata. Veće udaljenosti i različite orijentacije alata uzrokovale bi i veće pogreške kod robota[4]. Uz pomoć veće točnosti pozicioniranja optičkog sustava za praćenje, moguće je poboljšati robotovu točnost pozicioniranja. To će se dokazati u nastavku ovog diplomskog rada.

5. Aplikacija za vođenje alata robotske ruke

Nesavršenosti mehaničkih elemenata robota u odnosu na njegov nominalni model uzrokuju pogreške prilikom pozicioniranja robota u zadanu točku u prostoru. U sklopu ovog poglavlja razvijen je i prikazan upravljački program za potrebe vođenja robota i ispravljanja (smanjivanja) pogreške pozicioniranja robota uz pomoć vizijskog sustava za praćenje. Pomoću medicinskog optičkog sustava za praćenje Polaris Vicra procjenjuju se relativna odstupanja i na taj način se ispravljaju greške pozicioniranja.

Razmjena podataka između PC-a i robota koristi razvijenu „korisnik-poslužilac“ programsku podršku. Računalo i robot povezani su Ethernet kablom i koriste TCP protokol za komunikaciju, koja je opisana u poglavlju 1.3.. PC ima ulogu servera ili poslužioca, a robot je u ovom slučaju korisnik ili klijent.

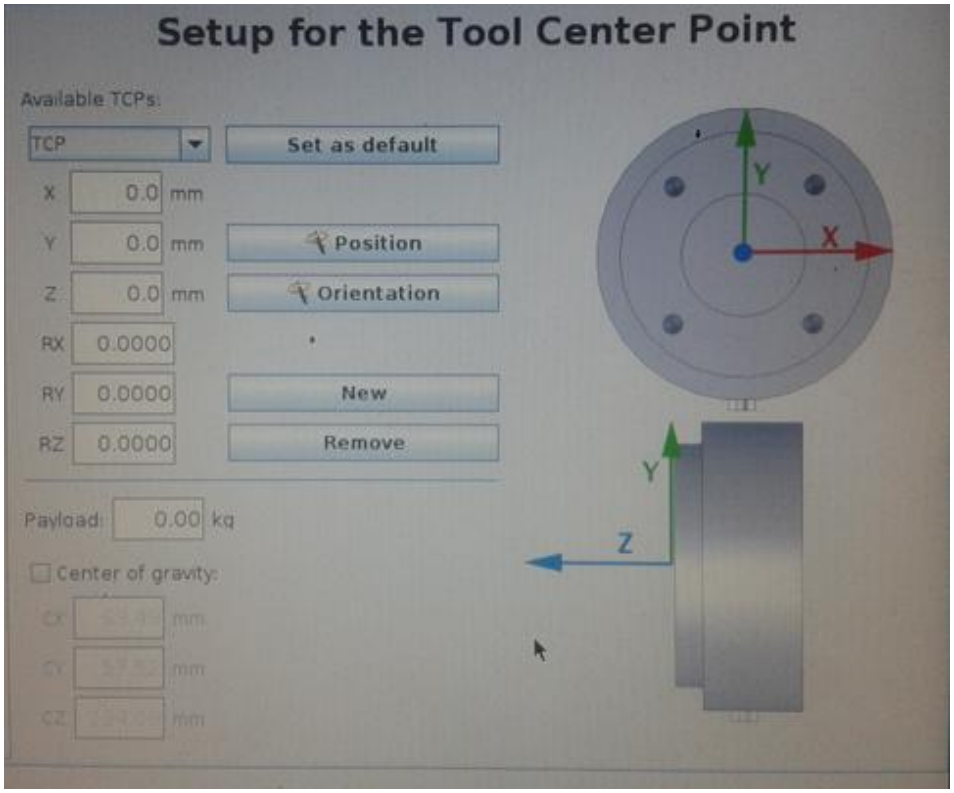
Slika 24. Markeri sa reflektirajućim sfernim kuglama



Slika 25. Dijagram toka informacija vođenja robota

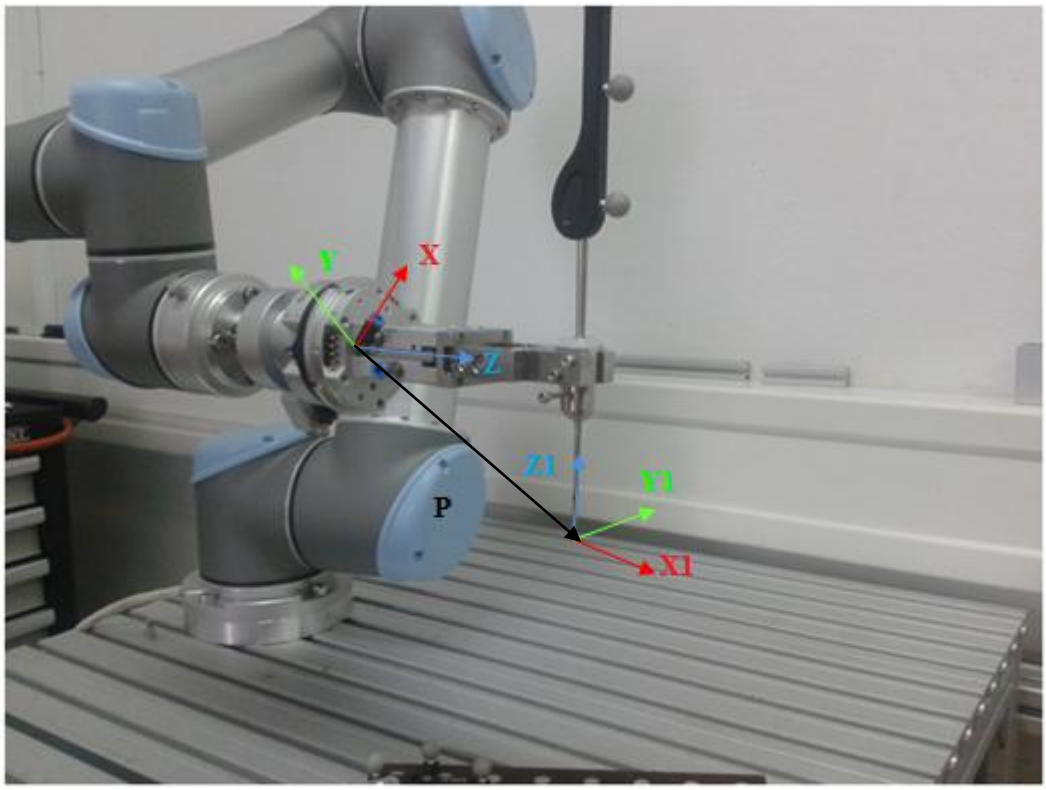
5.1. Kalibracija alata robota

Kako bi povezali položaj sonde na robotu i onaj koji pokazuje Polaris, bilo je potrebno napraviti kalibraciju alata robota. TCP (Tool Center Point) je karakteristična točka na vrhu alata kojim robot manipulira.



Slika 26. Unos parametara za kalibraciju alata.

Na slici 26. prikazana je pozicija koordinatnog sustava alata bez kalibracije i kakva mu je orijentacija. Kalibracijom se taj koordinatni sustav translatira za vektor ***P*** u vrh sonde i rotira na način da se *Z* os poklapa sa uzdužnom osi sonde (Slika 27.).



Slika 27. Kalibracija koordinatnog sustava alata

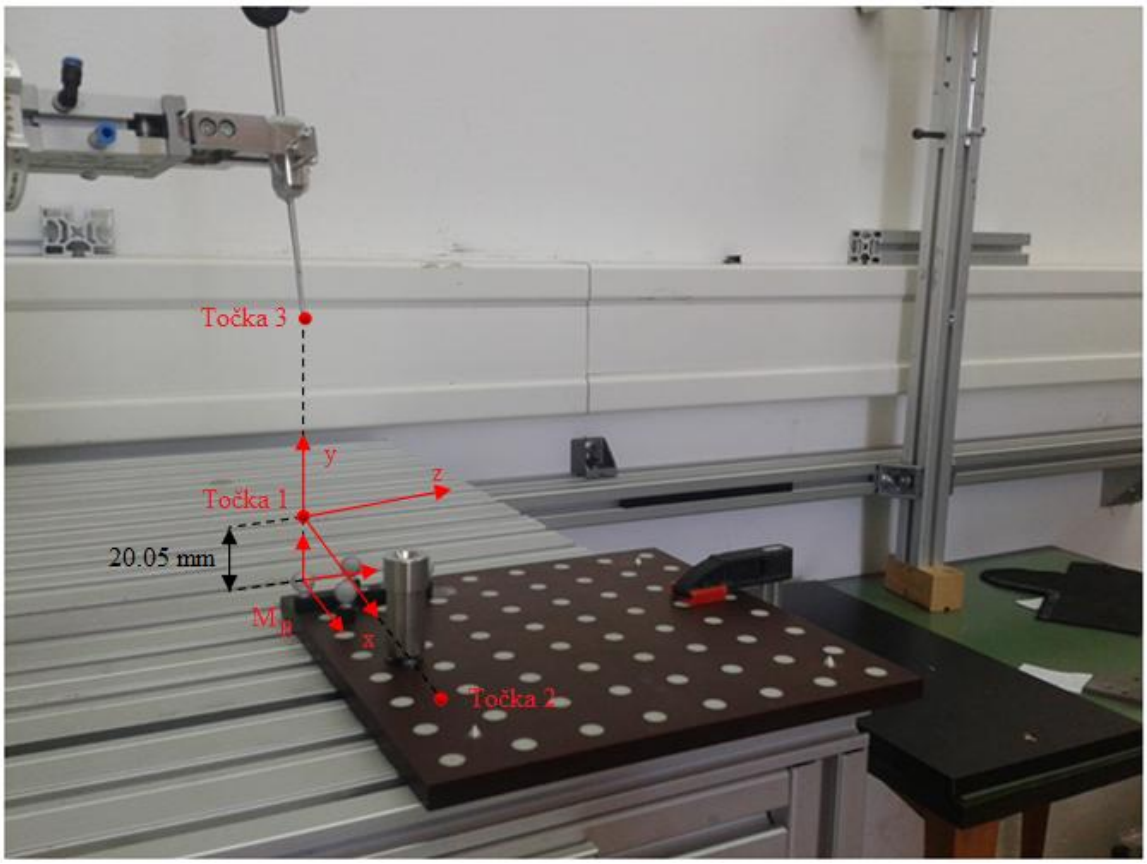
5.1.1. Podešavanje orijentacije

Za ispravno određivanje orijentacije alata, nužno je sljedeće: **orijentacija koordinatnog sustava referentnog markera M_R i orijentacija novog koordinatnog sustava robota „Plane“ moraju biti iste**. Dakle, prvi korak je namještavanje novog koordinatnog sustava „*Plane*“ u robotu (Slika 28.).

Novi koordinatni sustav robota definira se sa tri točke. Prva točka predstavlja ishodište koordinatnog sustava i pozicionirana je 20 mm iznad ishodišta referentnog markera. Druga točka predstavlja smjer Y osi, a treća točka je potrebna za generiranje X-Y ravnine. Os Z okomita je na X-Y ravninu. Robot je doveden u te tri točke ručno, pomoću preciznog navođenja robota. Kako bi te tri karakteristične pozicije bile što preciznije, koristio se NDI Tool Tracker program pomoću kojeg su dobiveni relativni podaci za sondu u robotovoj prihvatnici.

Tablica 8. Definiranje novog koodrinatnog sustava robota.

	Točka_1	Točka_2	Točka_3
X	20.05	20.03	120.08
Y	0.04	176.1	-0.01
Z	-0.02	0.01	0.05



Slika 28. Definiranje novog koordinatnog sustava *Plane* na robotu

Nužno je da robot interpretira orijentaciju sonde na način kaku ju interpretira Polaris. Dakle, u ovom podpoglavlju izračunate su vrijednosti RX, RY i RZ (kutevi izraženi u radijanima). Na taj način postiže se da je u svakom trenutku orijentacija očitana na robotu jednaka orijentaciji sonde očitanoj na Polarisu. Za računanje RX, RY i RZ kuteva, koristio se programski alat MatLab.

Kako bi odredili RX, RY i RZ kuteve, potrebno je očitati orijentaciju robota u *Plane* koordinatnom sustavu kada je TCP alata bez zadanih orijentacija. Ti podaci se prije računanja prebace u rotacijsku (DCM) matricu ***R_dcm***, formule (10), (11), (12), (13) i (14). Nakon toga, očita se (relativna) orijentacija markera sa Polarisa koja je prikazana u kvaternionima i ona se također prebaci u rotacijsku matricu ***P_dcm***, formula (8) . Pomoću sljedeće formule izračuna se matrica transformacija koja je zapravo veza između kuteva koje pokazuje Polaris i kuteva koji se očitavaju na robotu:

$$\boldsymbol{T_k} = \boldsymbol{R\_dcm^{-1}} * \boldsymbol{P\_dcm} \tag{25}$$

Dobivena $\boldsymbol{T_k}$ matrica je dimenzija 3x3 i potrebno ju je prebaciti u „Axis-angle“ notaciju, objašnjenu u poglavlju 3.1.. Takav zapis definira kombinacija jediničnog vektora $\hat{e}$ i kut rotacije θ:

$$\boldsymbol{V_A} = [e_x \quad e_y \quad e_z \quad \theta\,] \tag{26}$$

a za kalibraciju alata je u ovom slučaju potreban trodimenzionalan vektor koji dobijemo množenjem prve tri vrijednosti $\boldsymbol{V_A}$ sa kutom θ:

$$R_x = e_x \cdot \theta; \qquad R_y = e_y \cdot \theta; \qquad R_z = e_z \cdot \theta \tag{27}$$

Za računanje tih vrijednosti koristio se programski paket MatLab. Ispod teksta priložen je kod i originalne vrijednosti koje su korištene:

```
                    %podaci sa Polarisa

q0=0.2586;
q1=-0.6257;
q2=-0.0936;
q3=-0.7300;
Q=[q0 q1 q2 q3];            %Definiranje kvaterniona Q.
polaris_dcm=quat2dcm(Q);    %Pretvaranje Q u matricu rotacije.
polaris_dcm=polaris_dcm.';  %Transponiranje matrice.

        %angle-axis zapis, podaci sa robota u „plane“

e1=0.1674;
e2=0.3450;
e3=-3.3621;
theta=sqrt(e1*e1+e2*e2+e3*e3); %Računanje kuta theta.
r=[e1 e2 e3 theta];            %Definiranje Eulerovog vektora.
robot_dcm=vrrotvec2mat(r);     %Pretvaranje vektora u mat.rot.
T=robot_dcm^-1*polaris_dcm     %Računanje mat. rotacije T.
R1=vrrotmat2vec(T);            %Eulerov vektor.
r1=r1*r1(4)                   %Računanje vrijednosti RX,RY,RZ.
```

Dobivene vrijednosti za orijentaciju alata su jednake:

- RX=0.5208 rad
- RY=-1.5180 rad
- RZ=0.6468 rad

5.1.2. Podešavanje pozicije alata

Kod kalibracije pozicije alata koordinatni sustav se iz originalnog položaja prikazanog na slici 26, pomakne za X, Y i Z vrijednost (Slika 27.). Vrijednosti X, Y i Z određuju se pomoću alata za kalibraciju, na način da se vrh alata robotske ruke dovede u vrh alata za kalibraciju iz četiri različita kuta, tj. četiri različite orijentacije (Slika 29.). Svaku od četiri točke robot spremi, potom automatski izračuna X, Y i Z pomake i upiše ih u gornji prozor. Na taj način se koordinatni sustav sa originalnog položaja na vrhu robotske ruke pomaknuo u vrh sonde na prihvatnici.

Slika 29. Kalibracija alata na robotu.

Dobivene su sljedeće vrijednosti za poziciju:

- X=63.49 mm
- Y=57.52 mm
- Z=234.09 mm

5.2. Upravljački program

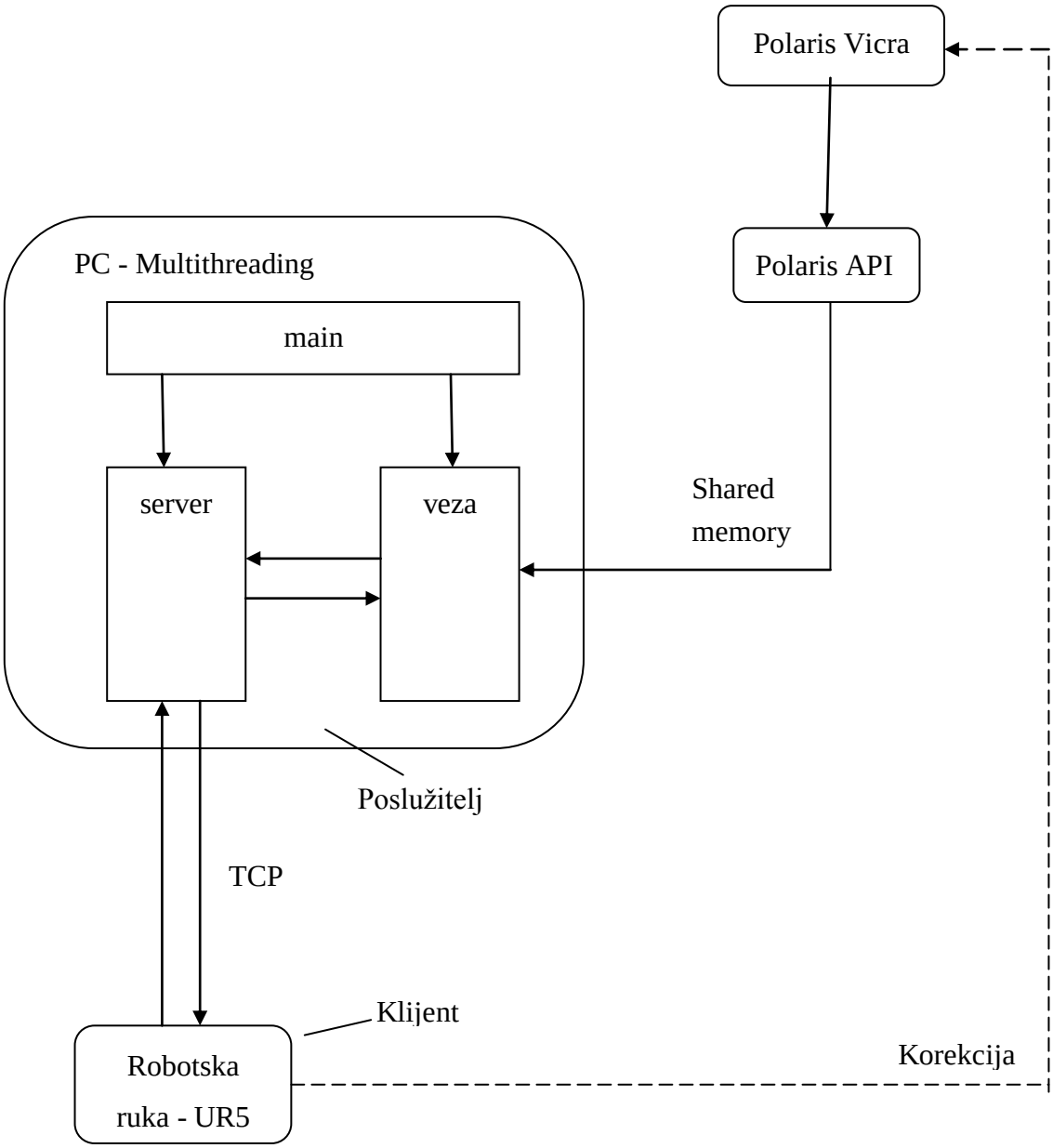
U ovom poglavlju opisan je princip rada upravljačkog programa za potrebe upravljanja robota i smanjivanje njegove pogreške pozicioniranja. Upravljački program razvijen je u C++ programskom jeziku koristeći razvojni alat Microsoft Visual Studio i prethodno opisanu Eigen C++ knjižnicu za linearnu algebru.

Kako bi cijeli upravljački program radio, Polaris API aplikacija cijelo vrijeme mora biti uključena tj. prikupljati podatke. Izvorni kod Polaris API aplikacije prilagođen je na način da prikuplja informacije o poziciji i orijentaciji oba markera i šalje ih u glavni dio programa pomoću tzv. „Shared memory-a“. „Shared memory“ ili dijeljenje memorije je najbrži inter-procesorski komunikacijski mehanizam. Programski kod bit će u cijelosti priložen u prilogu na kraju diplomskog rada. Kako bi dijeljenje memorije između dva procesa radilo, moraju biti napravljena dva osnovna koraka:

1. Zahtjeva se od operativnog sistema segment memorije koji se slobodno može dijeliti između procesa. Korisnik može stvoriti, uništiti ili otvoriti tu memoriju koristeći „shared“ memorijski objekt, koji je u ovom slučaju definiran kao vektor. To je objekt koji predstavlja memoriju koja može biti mapirana istovremeno u adresni prostor više od 1 procesa [20].

2. Pridružiti tu memoriju ili dio nje adresnom prostoru prvog procesa, u ovom slučaju je to Polaris API. Operativni sustav taj raspon memorijskih adresa označi kao specifični raspon. Promjene u tom rasponu automatski su vidljive u drugom procesu koji je također mapirao isti memorijski objekt [20].

Glavni dio programa ima ulogu poslužitelja ili servera. On prima podatke sa Polarisa, obrađuje ih i pretvara u format prepoznatiljiv za robota. Matematički algoritam bit će kasnije objašnjen. Potom tako obrađene podatke šalje robotu prema definiranom protokolu. Glavni dio programa izveden je pomoću „Multithreading“-a ili višedretvenosti. To je sposobnost računala da pokreće dva ili više programa istovremeno, tj. sposobnost programa da pokreće istovremeno dva procesa ili dva odvojena dijela programa.

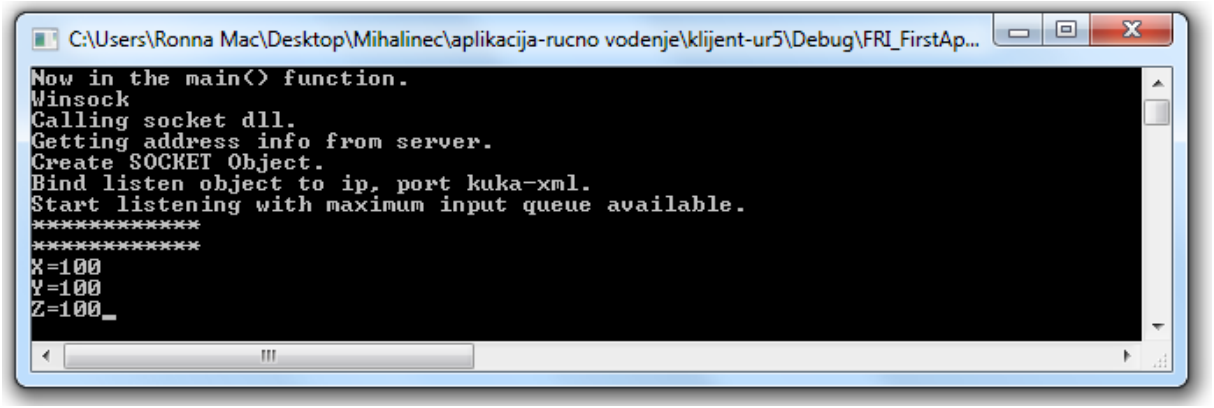


Slika 30. Dijagram toka informacija

Na slici 30. prikazani su dijagram toka informacija i arhitektura upravljačkog programa. Transformirani podaci sa OTS-a koriste se za vođenje robota u njegovom koordinatnom sustavu (Base).

5.2.1. Ručni unos koordinata

Upravljački program prije pokretanja robota traži od korisnika da upiše koordinate u koje želi poslati robota (Slika 31). U ovom primjeru želimo robota poslati u točku čije su koordinate: T(100,100,100).



Slika 31. Upis koordinata.

Tada se pritisne *Play* tipka na tabletu robota kako bi se pokrenuo program na robotu .tj kako bi se uspostavila „korisnik-poslužilac“ komunikacija. Niže je priložen kod programa koji se izvršava na robotu.

Svaki put kada poruka stigne sa robota na *server*, server ulazi u petlju i šalje podatke robotu sve dok robot ne pošalje poruku „prim“. Tada *server* izlazi iz petlje i prestaje slati. Poruka „prim“ zapravo služi kako prekidač neželjenog slanja.

```
Program

  BeforeStart
    Receive_Data1:=[0,0,0,0,0,0]
    Rec2:=[0,0,0,0,0,0,0]
    target_1:=p[0,0,0,0,0,0]
    target_2:=p[0,0,0,0,0,0]

  Robot Program
    socket_open("192.168.123.200",200)  % Početak komunikacije.
    var_3:=get_target_tcp_pose()     %UR očitava položaj vrha alata.
    Wait: 1.0
    socket_send_string(var_3)    %Šalje položaj vrha alata serveru.
    socket_send_string("prim")    %Prekid slanja.
    Wait: 1.0
    socket_send_string("tran")        %Transformacija.
```

Deklariranje varijabli prije pokretanja programa.

```
    Wait: 1.0
    socket_send_string("nove")        %Zahtjeva nove vrijednosti.
    Receive_Data1=socket_read_ascii_float(6)%Prima nove vrijednosti
    If Receive_Data1[1] and Receive_Data1[2] and Receive_Data1[3]≠0
      socket_send_string("prim")     %Prekid slanja.
    Else
      Halt
    Script: spremi1_b.script       %U skripti se primljeni podaci
iz Receive_Data1 spremaju u varijablu target_1.
    MoveL                          %Robot kreće u target_1.
      target_1
    socket_send_string("save")  %Računanje pogreške pozicioniranja.
    Wait: 1.0
    socket_send_string("prim")          %Prekid slanja.
    var_4:=get_target_tcp_pose()%UR očitava novi položaj vrha alata.
    Wait: 1.0
    socket_send_string(var_4)           %Šalje novi položaj.
    socket_send_string("prim")          %Prekid slanja.
    Wait: 1.0
    socket_send_string("tran")     %Računanje nove transformacije.
    Wait: 1.0
    socket_send_string("nove")      %Zahtjeva nove vrijednosti.
    Rec2=socket_read_ascii_float(6) $Prima nove vrijednosti.
    If (Rec2[1]≠Receive_Data1[1] and Rec2[2]≠Receive_Data1[2]) and
(Rec2[1] and Rec2[2] and Rec2[3]≠0)
      socket_send_string("prim")    %Prekid slanja.
    Else
      Halt
    Script: spremi2_b.script      %Spremanje podataka iz varijable
Rec2 u varijablu target_2.
    MoveL                        %Ispravljanje pozicije.
      target_2
    socket_send_string("sav1")   %Računanje nove pogreške
popzicioniranja.
    Wait: 1.0
    socket_send_string("prim")    %Prekid slanja.
    Halt
```

Nakon što se uspostavi komunikacija, optički sustav opaža dva reflektirajuća markera i određuje njihovu trenutnu poziciju i orijentaciju. Podatke za marker koji je pričvršćen za kalibracijsku ploču optički sustav određuje u vlastitom koordinatnom sustavu **O**, a podatke za sondu u robotovoj prihvatnici određuje u relativnom koordinatnom sustavu $\boldsymbol{M_R}$ koji se nalazi u marketu na ploči. Sonda je inicijalno u nekoj proizvoljnoj točci unutar radnog volumena optičkog sustava. Ti se podaci iz aplikacije API u obliku vektora šalju na server koji je na PC-u:

Marker na ploči: x=190.179993, y=34.41, z=-927.659973, Q0=0.0268, Qx=0.1916, Qy=-0.0534, Qz=0.9796;

Sonda: x=208.880005, y=126.00, z=40.689999, Q0=0.0986, Qx=-0.5685, Qy=0.0047, Qz=-0.8167;

Istovremeno, robot u obliku niza znakova šalje trenutnu poziciju i orijentaciju vrha sonde. Niz znakova se u programu automatski pretvara u niz vrijednosti tipa Float, kako bi se dalje moglo računati.

var_1= p[-0.429462, -0.0451054, 0.0757729, 0.467062, 0.0202697, -2.21141]

Prve 3 vrijednosti izražene su u metrima i predstavljaju poziciju, a zadnje tri predstavljaju orijentaciju u radijanima. Pomoću Eigen C++ knjižnice, podprogram „veza“ uzima podatke za sondu sa aplikacije API i podatke od robota *var_1* i pretvara ih u rotacijske (DCM) matrice ${}^{B}_{M1}\boldsymbol{P}$, ${}^{M2}_{M1}\boldsymbol{P}$ koje su proširene vektorima položaja:

$$ {}^{M2}_{M1}\boldsymbol{P} = \begin{bmatrix} -0.334128 & 0.155719 & 0.929575 & 208.88 \\ -0.166408 & -0.98051 & 0.104438 & 126 \\ 0.927721 & -0.119793 & 0.353529 & 40.69 \\ 0 & 0 & 0 & 1 \end{bmatrix}; $$

$$ {}^{B}_{M1}\boldsymbol{P} = \begin{bmatrix} -0.566281 & 0.757919 & -0.32386 & -429.462 \\ -0.751855 & -0.636012 & -0.173791 & -45.1054 \\ -0.337699 & 0.145081 & 0.930006 & 75.7729 \\ 0 & 0 & 0 & 1 \end{bmatrix}; $$

U programski kod također su unesene i vrijednosti za pomak koordinatnog sustava sonde tj. Tool Tip Offest:

$$ \boldsymbol{TTO} = \begin{bmatrix} 1 & 0 & 0 & -19.1055 \\ 0 & 1 & 0 & -1.4255 \\ 0 & 0 & 1 & -157.6049 \\ 0 & 0 & 0 & 1 \end{bmatrix}; $$

Potrebna matrica transformacije računa se po formuli:

$$\boldsymbol{T}={}_{\boldsymbol{M1}}^{\boldsymbol{B}}\boldsymbol{P}*\left({}_{\boldsymbol{M1}}^{\boldsymbol{M2}}\boldsymbol{P}*\boldsymbol{TTO}\right)^{-1} \tag{28}$$

Dobije se matrica transformacije $\boldsymbol{T}$ iz Polarisovog koordinatnog sustava u robotov koordinatni sustav:

$$\boldsymbol{T}=\begin{bmatrix} 0.00618098 & -0.682737 & -0.730638 & -375.779 \\ -0.00937553 & 0.730581 & -0.682762 & -150.08 \\ 0.999937 & 0.0110703 & -0.00188542 & 5.91631 \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

Sljedeći dio programa računa koordinate potrebne robotu da bi došao u zadanu točku T(100,100,100). Željena orijentacija je unaprijed definirana u ovom djelu programa i glasi :

Q0=0.1310, Qx=-0.7038, Qy=0.1221, Qz=-0.6871;

Zadanu orijentaciju potrebno je transformirati u matricu rotacije $\boldsymbol{R}$. Sonda je u takvoj orijentaciji uvijek okomito okrenuta u odnosu na kalibracijsku ploču. Program se može prilagoditi tako da korisnik upisuje i željenu orijentaciju u obliku kvaterniona ili vektora rotacije, u ovom primjeru unaša se samo željena pozicija. Program dalje računa matricu rotacije gornjih vrijednosti proširenu sa unešenim vrijednostima, tj. vektorom pomaka:

$$\boldsymbol{V}=[100 \quad 100 \quad 100 \quad 0],$$

Matrica rotacije $\boldsymbol{R}$ iznosi:

$$\boldsymbol{R}=\begin{bmatrix} 0.0254931 & 0.00815621 & 0.999642 & 100 \\ -0.352061 & -0.93583 & 0.0166139 & 100 \\ 0.93563 & -0.352358 & -0.0209859 & 100 \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

Sada je potrebno pomnožiti matricu $\boldsymbol{R}$ sa matricom transformacije $\boldsymbol{T}$:

$$\boldsymbol{R_1} = \boldsymbol{T} * \boldsymbol{R} \tag{29}$$

Konačno se dobije matrica $\boldsymbol{R_1}$ iz koje treba izvaditi podatke za robota:

$$\boldsymbol{R_1} = \begin{bmatrix} -0.443085 & 0.896422 & 0.0101689 & -516.499 \\ -0.896261 & -0.443199 & 0.017094 & -146.236 \\ 0.01983 & -0.00153987 & 0.999802 & 106.829 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. stupac matrice $\boldsymbol{R_1}$ predstavlja X,Y,Z koordinate za robotov koordinatni sustav „Base“:

Željena pozicija u „Base“ od robota iznosi: X=-516.499, Y=-146.236, Z= 106.829;

Željena orijentacija dobije se na način da se iz 3x3 podmatrice $\boldsymbol{R_1}$ prvo izračuna tzv. Eulerov vektor: $\boldsymbol{V_{rot}} = [e_x \quad e_y \quad e_z \quad \theta\ ]$, pomoću sljedećih formula:

$$\theta = \text{acos}\left(\frac{m(1,1) + m(2,2) + m(3,3) - 1}{2}\right) \tag{30}$$

$$e_x = \frac{m(3,2) - m(2,3)}{\sqrt{(m(3,2) - m(2,3))^2 + (m(1,3) - m(3,1))^2 + (m(2,1) - m(1,2))^2}} \tag{31}$$

$$e_y = \frac{m(1,3) - m(3,1)}{\sqrt{(m(3,2) - m(2,3))^2 + (m(1,3) - m(3,1))^2 + (m(2,1) - m(1,2))^2}} \tag{32}$$

$$e_z = \frac{m(2,1) - m(1,2)}{\sqrt{(m(3,2) - m(2,3))^2 + (m(1,3) - m(3,1))^2 + (m(2,1) - m(1,2))^2}} \tag{33}$$

Gdje $\boldsymbol{m}$ predstavlja 3x3 podmatricu od $\boldsymbol{R_1}$:

$$\boldsymbol{m} = \begin{bmatrix} -0.443085 & 0.896422 & 0.0101689 \\ -0.896261 & -0.443199 & 0.017094 \\ 0.01983 & -0.00153987 & 0.999802 \end{bmatrix};$$

Na kraju vektor rotacije dobijemo na sljedeći način:

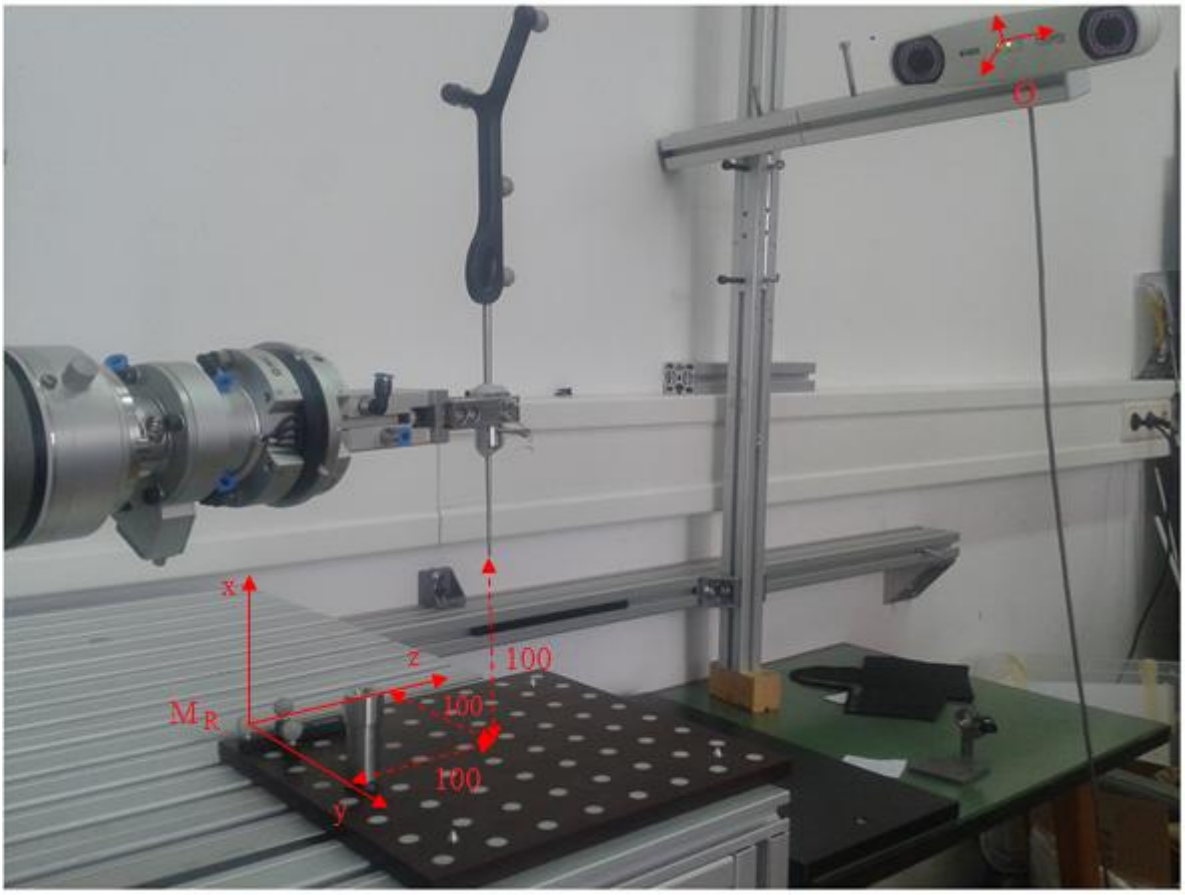
$$e_1 = e_x * \theta \tag{34}$$

$$e_2 = e_y * \theta \tag{35}$$

$$e_3 = e_z * \theta \tag{36}$$

i on je jednak: $e_1 = -0.0210993$, $e_2 = -0.0109393$, $e_3 = -2.02987$.

U ovom trenutku „server“ strana programa šalje robotu izračunate vrijednosti. Robot ih prima i kreće u zadanu točku, slika 32 .



Slika 32. Vođenje robota ručnim unosom koordinata

U trenutku kada robot dođe u zadanu točku, potrebno je izračunati pogrešku pozicioniranja robota. U tu svrhu aplikacija API šalje trenutnu poziciju i orijentaciju sonde i podprogram “veza“ izračuna kolika je pogreška pozicioniranja. Pogreška se računa u obliku Euklidske udaljenosti (formula 21). Nakon toga program automatski pohranjuje izračunate podatke u zasebnu datoteku. Izračunata pogreška pozicioniranja iznosi:

$$|x - x'| = 0.398087\ mm;$$

$$|y - y'| = 0.780655\ mm\ ;$$

$$|z - z'| = 0.865128\ mm;$$

$$E_1 = \sqrt{|x - x'|^2 + |y - y'|^2 + |z - z'|^2} = 1.2314\ mm$$

Zadatak je smanjiti izračunatu robotovu pogrešku pozicioniranja pomoću vizijskog sustava za praćenje. Nakon što je program završio pohranjivanje izračunate pogreške, ponavlja cijeli postupak računanja. Dakle, uzima nove trenutne vrijednosti za sondu, jedne sa robota *var*_2 i jedne sa API-a kako bi dobio novu matricu transformacije $\boldsymbol{T_1}$, koja je vrlo slična prijašnjoj:

$$\boldsymbol{T_1} = \begin{bmatrix} 0.00835486 & -0.682206 & -0.731112 & -376.117 \\ -0.0104653 & 0.731038 & -0.682256 & -151.332 \\ 0.99991 & 0.0133515 & -0.00103196 & 6.01096 \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

Matrica $\boldsymbol{R}$ koja predstavlja poziciju i orijentaciju zadane točke ostala je ista:

$$\boldsymbol{R} = \begin{bmatrix} 0.0254931 & 0.00815621 & 0.999642 & 100 \\ -0.352061 & -0.93583 & 0.0166139 & 100 \\ 0.93563 & -0.352358 & -0.0209859 & 100 \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

Nakon što se pomnože matrice prema formuli 17, robot dobije nove vrijednosti kako bi se ispravio. U tom trenutku se pomakne i smanji ukupnu pogrešku pozicioniranja. Pri završetku gibanja program ponovno uzima trenutne podatke sa API-a i uspoređuje ih sa zadanima kako bi izračunao pogrešku pozicioniranja koja je sada smanjena. Ona sada iznosi:

$$|x - x'| = 0.031662\ mm;$$
$$|y - y'| = 0.0568695\ mm\ ;$$
$$|z - z'| = 0.115089\ mm;$$

$$E_2 = \sqrt{|x - x'|^2 + |y - y'|^2 + |z - z'|^2} = 0.132209\ mm$$

Prikazan je primjer za samo jednu točku. Kako bi se izvukli neki konkretniji zaključci, ispitana je točnost pozicioniranja za 36 nasumično odabranih točaka unesenih u upravljački program. Rezultati su prikazani u tablici 9.

Tablica 9.Odstupanje ispravljenog položaja od zadanog.

Vrsta pogreške	Pogreške pozicioniranja nakon ispravljanja			
	x	y	z	Euklidska
Srednja vrijednost	0.066	0.068	0.096	0.154
Maks. vrijednost	0.189	0.244	0.305	0.343
STD	0.053	0.068	0.073	0.028
CI 95%[±]	0.017	0.022	0.024	0.028
RMS	0.084	0.094	0.119	0.174

Mjerile su se udaljenosti od ispravljenih položaja do inicijalno zadanog položaja. U tablici 9. vidljivo je da *RMS* pogreška robota nakon ispravljanja optičkim sustavom iznosi 0.174 mm. Međutim, ti podaci ne mogu biti pokazatelji točnosti pozicioniranja robota iz razloga što sam optički sustav ima pogrešku pozicioniranja od 0.398 mm (Tablica 7.). Već kod postupka računanja transformacije Polaris šalje podatke za sondu u koju mora biti uključena određena pogreška. **Pogreška pozicioniranja robota nikako ne može biti manja od pogreške optičkog sustava.**

5.2.2. Unaprijed definirani položaji

Prethodno objašnjena aplikacija sa ručnim unosom podataka iz prošlog poglavlja, ovdje je modificirana. Za razliku od prošle aplikacije, u ovoj aplikaciji unaprijed su zadane sve pozicije i orijentacije u koje robot mora doći. Ideja je bila napraviti aplikaciju koja će omogućiti da robot može ponavljati jednake pokrete bez obzira ako se objekt našeg interesa slučajno pomakne.

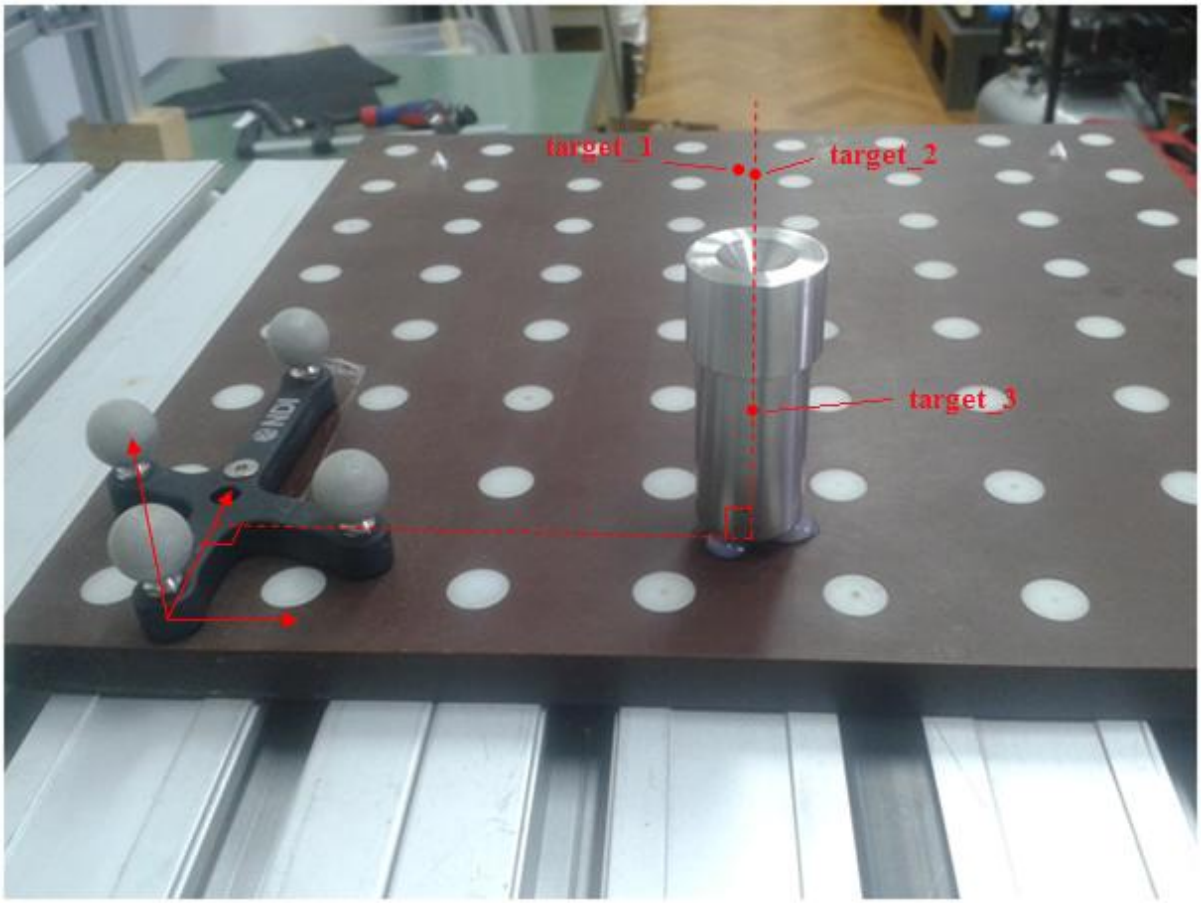
Kada se objekt pomakne, robot se pomoću optičkog sustava automatski navodi i dolazi u nove pozicije koje su uvijek u jednakom odnosu obzirom na referentni marker $\mathbf{M}_R$. **Robotov zadatak je sljedeći: kada se kalibracijska ploča pomakne, robot se mora pozicionirati iznad čahure, uvesti sondu 2 cm u čahuru u određenoj poziciji i izvadi ju van.** Čahura je pričvršćena za kalibracijsku ploču, slika 33.

Slika 33. Čahura pričvršćena za kalibracijsku ploču.

Upravljački program inicijalno ima zadana dva položaja u koje robot mora doći. Prvi položaj nalazi se iznad čahure, a drugi 2 cm unutar nje. Promjer rupe na čahuri je veličine 3.25 mm, a debljina sonde 2.5 mm. Aplikacija koristi identičan matematički algoritam opisan u prošlom poglavlju u svrhu korigiranja pozicijske pogreške. U sklopu diplomskog rada, snimljen je kratak video gdje se prikazuje automatsko pozicioniranje alata robotske ruke prema objašnjenom protokolu:

Opis programa:

- **Robot kreće prema inicijalno zadanom prvom položaju, ali zbog pogreške pozicioniranja dolazi u položaj nazvan *target_1*;**
- **Optički sustav ispravi položaj *target_1*, robot se pomakne i zapamti ispravljeni položaj kao *target_2* ;**
- **Robot uvodi sondu unutar čahure u inicijalno zadan drugi položaj koji je označen kao *target_3;***
- **Robot čeka dvije sekunde;**
- **Robot se vraća u korigiran položaj *target_2*.**



Slika 34. Unaprijed definirane pozicije.

Važno je napomenuti da vrijednosti inicijalno zadane prve pozicije i vrijednosti pozicije *target_2* (koja je ispravljena) nisu identične zbog pogreške optičkog sustava. U poziciji *target*_2 robot još uvijek ima određenu pogrešku pozicioniranja koja je u ovom slučaju smanjena. Dokaz da je pogreška pozicioniranja smanjena, bit će prikazan u sljedećem poglavlju, a to je vidljivo i iz snimljenog videa. Niže je priložen programski kod koji se izvršava na robotu:

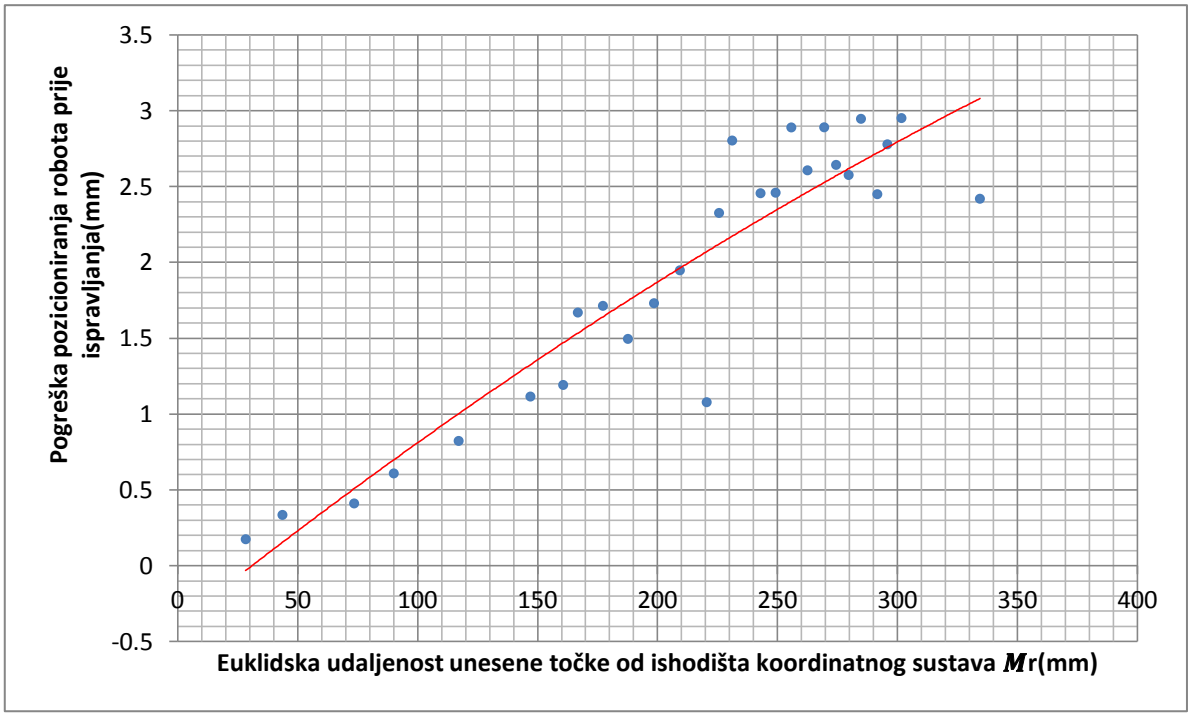
```
Program
  BeforeStart
    Receive_Data1:=[0,0,0,0,0,0,0]
    Receive_Data2:=[0,0,0,0,0,0,0]
    Receive_Data3:=[0,0,0,0,0,0,0]
    target_1:=p[0,0,0,0,0,0]
    target_2:=p[0,0,0,0,0,0]
    target_3:=p[0,0,0,0,0,0]
  Robot Program
    socket_open("192.168.123.200", 200)
    var_1:=get_target_tcp_pose()
    Wait: 1.0
    socket_send_string(var_1)
```

```
      socket_send_string("prim")
      Wait: 1.0
      socket_send_string("tran")
      Wait: 1.0
      socket_send_string("nove")
      Receive_Data1=socket_read_ascii_float(6)
      If Receive_Data1[1] and Receive_Data1[2]≠0
        socket_send_string("prim")
      Else
        Halt
      Script: spremi.script
      MoveL
        target_1
      socket_send_string("save")
      Wait: 1.0
      socket_send_string("prim")
      var_2:=get_target_tcp_pose()
      Wait: 1.0
      socket_send_string(var_2)
      socket_send_string("prim")
      Wait: 1.0
      socket_send_string("tran")
      Wait: 1.0
      socket_send_string("nove")
      Receive_Data2=socket_read_ascii_float(6)
      If Receive_Data2[1]≠Receive_Data1[1]
        socket_send_string("prim")
      Else
        Halt
      Script: spremi1.script
      MoveL
        target_2
      socket_send_string("sav1")
      Wait: 1.0
      socket_send_string("prim")
      var_3:=get_target_tcp_pose()
      Wait: 1.0
      socket_send_string(var_3)
      socket_send_string("prim")
      Wait: 1.0
      socket_send_string("tra1")
      Wait: 1.0
      socket_send_string("nove")
      Receive_Data3=socket_read_ascii_float(6)
      If (Receive_Data3[0] and Receive_Data3[1]≠0) and
(Receive_Data3[1]≠Receive_Data2[1])
        socket_send_string("prim")
      Else
        Halt
      Script: spremi2.script
      MoveL
        target_3
      socket_send_string("sav2")
      Wait: 1.0
      socket_send_string("prim")
      Wait: 3.0
      MoveJ
```

```
  target_2
socket_send_string("kraj")
socket_send_string("prim")
Halt
```

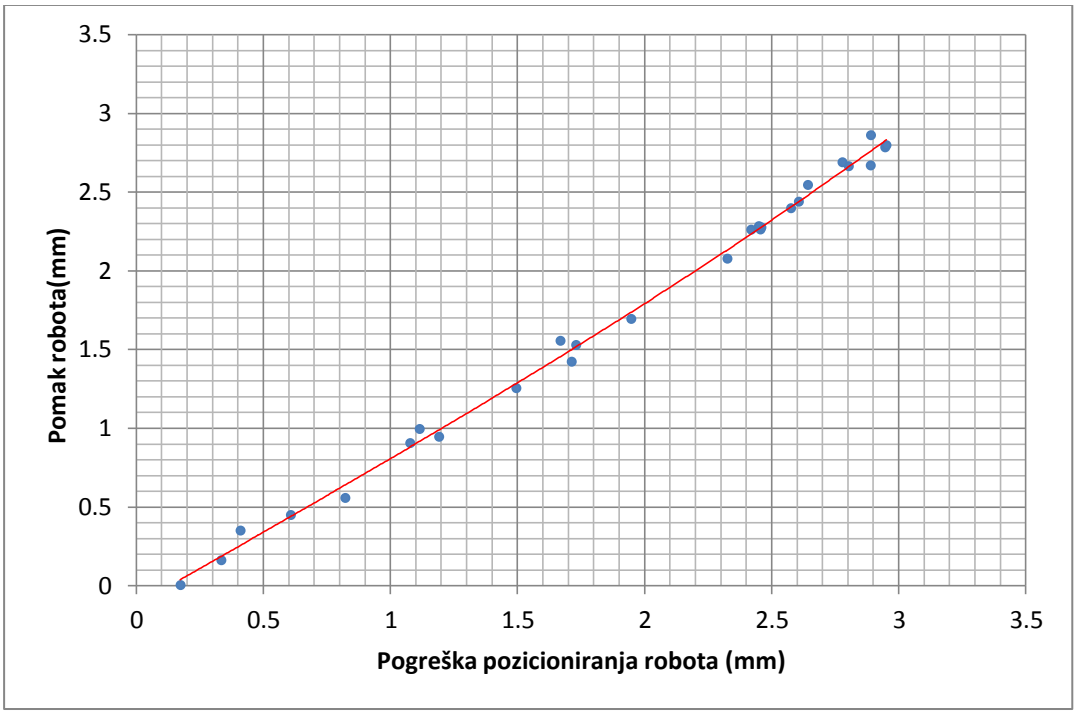
5.3. Analiza rezultata vođenja robota

Koristeći funkciju izrađenog upravljačkog programa da korisnik ručno upisuje koordinate u koje želi poslati robota, robot je poslan u 30 nasumično odabranih točaka. Vođenje robota u zadane točke izvršeno je pomoću optičkog sustava Polaris, obzirom na referentni koordinatni sustav $\boldsymbol{M_R}$ koji se nalazi u markeru na ploči. Orijentacija sonde cijelo je vrijeme bila ista. Robot se u svaku točku slao iz iste početne točke u koordinati T(20,0,0). Pogreška pozicioniranja robota izračunata je kao euklidska udaljenost od robotove trenutne pozicije do idealno točne pozicije.



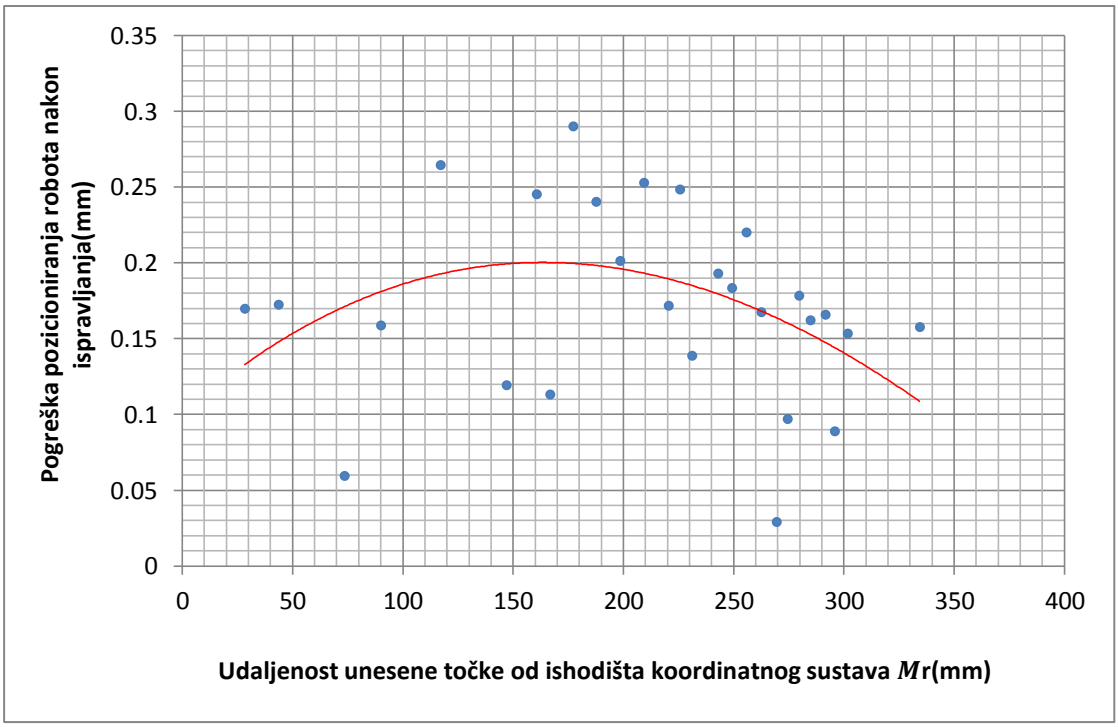
Slika 35. Pogreška pozicioniranja robota prije ispravljanja

Na slici 35. vidljivo je da pogreška pozicionranja robota raste sa većom udaljenošću zadane točke od ishodišta koordinatnog sustava. Nakon što se robot pozicionira u prostoru, optički sustav izračuna kolika mu je pogreška pozicioniranja i potom mu ispravi poziciju. Slika 36. prikazuje za koliko se robot pomakne u smjeru idealno točne pozicije u svrhu ispravljanja.



Slika 36. Prikaz pomaka robota u svrhu ispravljanja

Pomak kod ispravljanja, za svaku točku manji je od udaljenosti koja predstavlja pogrešku pozicioniranja prije ispravljanja za istu točku. Na slici 37. prikazano je da se pogreška pozicioniranja za svaku točku smanjila na ispod 0.3 mm, tj. da se robot vrlo dobro ispravio.



Slika 37. Smanjena pogreška pozicioniranja robota nakon ispravljanja

Crvenom je bojom u dijagramima označena trend linija koja je aproksimirana polinomom drugog reda. Dijagrami na slici 35. i 37. jasno prikazuju promjenu u trendu, prije i nakon ispravljanja pozicije robota.

Važno je napomenuti da su podaci koji se nalaze u dijagramima na slici 35., 36. i 37. izračunati pomoću optičkog sustava, dakle u te podatke uključena je pozicijska pogreška optičkog sustava na koju nikako ne možemo utjecati.

6. Zaključak

Zbog nesavršenosti svojih mehaničkih elemenata, robotska ruka griješi prilikom pozicioniranja. U ovom diplomskom radu izrađen je upravljački sustav koji omogućuje relativno vođenje alata robotske ruke na temelju informacija izmjerenih pomoću medicinskog optičkog sustava (PolarisVicra) za praćenje markera. Uz pomoć veće točnosti pozicioniranja optičkog sustava za praćenje, poboljšana je robotova točnost pozicioniranja. Kada se koristi relativno referenciranje i dva markera, robotova srednja RMS pozicijska pogreška smanjena je za 40.5% u ispitanom radnom prostoru. U udaljenim, rubnim područjima radnog prostora robota, pozicijska je pogreška veća. U tim dijelovima radnog prostora, srednju RMS pogrešku pozicioniranja robota moguće je smanjiti i za više od 40.5 %. Pogreška pozicioniranja kod robota ne može se smanjiti ispod točnosti optičkog sustava kojim je vođen. Mjerenjima je ustanovljeno da je točnost optičkog sustava podjednaka u cijelom njegovom radnom volumenu i orijentacija markera ne utječe bitno na nju.

Cijeli sustav upravljanja sastoji se od povezivanja tri komponente: optičkog sustava Polaris Vicra, robotske ruke UR5 i PC-a. Za izradu takvog upravljačkog sustava potrebno je naprednije poznavanje programskog jezika C++, poznavanje razmjene podataka na principu *„korisnik-poslužitelj“* kao i razumijevanje matematičkih algoritama za izračunavanje prostornih transformacija. Prednost tehnike vođenja pomoću optičkog sustava Polaris Vicra je i to što osvjetljenje nema utjecaj na vidljivost markera, zahvaljujući korištenju infra-crvenog spektra. Za vrijeme vođenja robota, oba markera moraju biti vidljiva optičkom sustavu, moglo bi se reći da je na neki način sustav time ograničen. Brzina razmjene podataka između robota i PC-a nije bila prioritet pa se za komunikaciju koristio TCP protokol. Kod aplikacija gdje se traži što brži prijenos podataka možda bi bilo bolje koristiti UDP komunikacijski protokol.

7. Literatura

[1] Kurtović B., Zdravstvena njega neurokirurških bolesnika, HKMS, 2013.

[2] Bargar W.L., Bauer A., Börner M., Primary and Revision Total Hip Replacement Using the Robodoc System, Clinical Orthopaedics and Related Research, 1998, 354: 82-91

[3] Rosen J., Hannaford B., Satava R.M., Surgical Robotics-System Applications and Visions, Springer, London 2011

[4] Šuligoj F., Jerbić B., Švaco M., Šekoranja B., Mihalinec D., Vidaković J., Medical applicability of a low-cost industrial arm guided with an optical tracking system, IROS 2015.

[5] Li Q.H., Zamorano L., Panday A., Perez R., Gong J., Diaz F., The Application Accuracy of the NeuroMate Robot-A Quantitative Comparison with Frameless and Frame-Based Surgical Localization System, Computer Aided Surgery, 7: 90-98

[6] Eljamel M.S., Validation of the PathFinder neurosurgical robot using a phantom, Int. J. Med. Robotics Comput. Assist. Surg. 2007, 3:372-377

[7] Kobler J.F., Kotlarski J., Lexow J.G., Majdani O., Ortmaier T., Design of Bone - Attached, Redundant and Reconfigurable Parallel Kinematic Device for Skull Surgery, ICRA 2014.

[8] Nathoo N., Cavusoglu M.C., Vogelbaum M.A., Barnett G.H., In Touch with Robotics: Neurosurgery for the Future, Neurosurgery 56:421-433, 2005

[9] http://ronna.fsb.hr/

[10] Haidegger T., Kovács L., Benyó B., Benyó Z., Spatial Accuracy of Surgical Robots, Biomedical Engineering Laboratory, Budapest 2009

[11] https://en.wikipedia.org/wiki/Transmission_Control_Protocol

[12] Northern Digital Inc., Polaris Vicra User Guide, Canada 2012

[13] https://www.fer.unizg.hr/_download/repository/Kvaternioni.pdf

[14] https://en.wikipedia.org/wiki/Gimbal_lock

[15] http://www.euclideanspace.com/maths/geometry/rotations/conversions/quaternionToMatrix/index.htm

[16] Universal Robots, Universal Robot User Manual, Version 3.0

[17] https://en.wikipedia.org/wiki/Axis%E2%80%93angle_representation

[18] Wiles A.D., Thompson D.G., Frantz D.D., Accuracy assessment and interpretation for optical tracking systems, SPIE Medical Imaging

[19] http://eigen.tuxfamily.org/index.php?title=Main_Page

[20] http://www.boost.org/

PRILOZI

I. Excel tablice

II. Programski kod

III. CD-R disc

I. EXCEL TABLICE

Tablica 10. Podaci sa Polarisa, potrebni za računanje pogrešaka navedenih u tablici 5

T(1,3)	60,2101	0,5698	0,0857
T(1,4)	90,0944	0,3655	0,0478
T(1,5)	120,0320	0,0939	0,0112
T(1,6)	150,0010	0,0000	0,0000
T(2,1)	0,2303	30,2563	0,0396
T(2,2)	30,1617	30,1334	0,0806
T(2,3)	60,3573	30,4385	0,0936
T(2,4)	90,3481	30,5099	0,0666
T(2,5)	120,1930	30,1531	0,0357
T(2,6)	150,0590	29,7940	-0,0100
T(3,1)	0,3005	60,2466	0,0261
T(3,2)	30,2249	59,9511	0,0418
T(3,3)	60,3936	60,1981	0,0658
T(3,4)	90,4224	60,1514	0,0922
T(3,5)	120,3670	60,2066	0,0484
T(3,6)	150,1170	59,5312	-0,0128
T(4,1)	0,3397	90,0714	0,0349
T(4,2)	30,2033	89,7103	0,0281
T(4,3)	60,2596	89,7901	0,0798
T(4,4)	90,3713	89,9198	0,0523
T(4,5)	120,3880	89,9571	0,0489
T(4,6)	150,2930	89,7041	0,0060
T(5,1)	0,5816	120,3000	0,0437
T(5,2)	30,5083	119,9280	0,0429
T(5,3)	60,4568	119,7290	0,0345
T(5,4)	90,4768	119,8120	0,0496
T(5,5)	120,3980	119,7350	0,0186
T(5,6)	150,2720	119,4470	0,0066
T(6,1)	0,5153	149,8300	0,0000
T(6,2)	30,5818	149,8420	0,0234
T(6,3)	60,4570	149,6890	0,0658
T(6,4)	90,3769	149,3930	0,0463
T(6,5)	120,4870	149,5110	0,0431
T(6,6)	150,3770	149,3260	0,0165

Tablica 11. Podaci sa Polarisa, potrebni za računanje pogrešaka navedenih u tablici 6

	X	Y	Z
T(1,1)	0	0,0000	0,0000
T(1,2)	29,8855	0,4438	0,0965
T(1,3)	60,1451	0,5284	0,1068
T(1,4)	89,8220	0,5599	0,1221
T(1,5)	119,9000	0,5654	0,0856
T(1,6)	149,6590	0,0000	0,0000
T(2,1)	-0,2128	30,2043	0,0826
T(2,2)	29,9914	30,0841	0,0913
T(2,3)	59,8931	30,2675	0,1236
T(2,4)	90,1707	30,5973	0,1342
T(2,5)	119,8370	30,3777	0,1299
T(2,6)	149,9270	30,0791	0,0595
T(3,1)	-0,0767	60,4256	0,0984
T(3,2)	30,0038	60,0631	0,1050
T(3,3)	59,8465	60,1320	0,1402
T(3,4)	90,0562	60,1960	0,1323
T(3,5)	119,8620	60,3963	0,1465
T(3,6)	150,0360	60,1146	0,0887
T(4,1)	-0,0348	90,1851	0,0975
T(4,2)	29,9977	89,7708	0,0677
T(4,3)	59,8212	89,8440	0,1587
T(4,4)	90,1489	90,0634	0,1177
T(4,5)	119,8050	89,7843	0,0990
T(4,6)	150,0930	90,0147	0,0749
T(5,1)	-0,1270	119,5660	0,0531
T(5,2)	30,0087	119,5820	0,0143
T(5,3)	59,6675	119,1730	0,0781
T(5,4)	90,1134	119,8840	0,1142
T(5,5)	119,7760	119,5630	0,0911
T(5,6)	149,7530	119,6250	0,1213
T(6,1)	-0,3845	148,9060	0,0000
T(6,2)	29,9529	149,8030	0,0718
T(6,3)	59,8203	149,5400	0,1161
T(6,4)	89,6872	149,1230	0,1036
T(6,5)	119,7600	149,2490	0,0859
T(6,6)	149,6890	148,8630	0,0809

Tablica 12. Tri mjerenja sa Polarisom

T	x1	y1	z1	x2	y2	z2	x3	y3	z3
1	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
2	30,0986	0,0384	0,1475	30,0807	0,1318	0,0656	30,0265	-0,1058	0,0877
3	60,0496	0,1945	0,0621	59,9195	-0,0300	0,1112	59,8688	-0,0345	0,0433
4	89,8655	-0,1415	0,0227	89,6336	-0,2077	0,0457	89,7810	-0,1007	0,0326
5	119,9570	0,0809	-0,0103	119,7700	-0,0022	0,1154	119,7500	-0,1367	0,0626
6	149,9700	0,0000	-0,0001	149,7640	0,0000	0,0000	149,9260	0,0000	0,0000
7	-0,0336	29,8399	0,1006	-0,0603	29,9199	0,0077	-0,0613	29,8550	0,0150
8	29,9802	29,9356	0,3672	29,8451	29,8926	0,3791	29,9511	30,0579	0,3522
9	60,0226	30,0644	0,0673	59,7825	29,9041	0,0577	59,8687	29,9480	0,0670
10	89,9213	30,0215	-0,0143	89,7179	29,8518	0,0287	89,8717	29,8335	0,0697
11	119,9180	29,9901	0,0028	119,7950	30,0645	0,0399	119,7790	29,8233	0,0734
12	149,9930	30,2149	-0,0836	149,7020	29,9527	0,0315	149,8170	29,8365	-0,0255
13	-0,0455	59,7536	0,1329	-0,1288	59,8643	0,0787	-0,1059	59,7622	0,1039
14	29,8732	59,7317	0,1672	29,7344	59,6992	0,0994	30,0168	59,9533	0,0025
15	59,8535	59,6777	0,1031	59,6941	59,7888	0,0376	59,9648	59,9203	0,0232
16	89,7169	59,7859	0,0923	89,5724	59,7556	0,0912	89,8009	59,8267	0,0914
17	119,7950	59,9128	0,0356	119,6890	59,9526	0,0888	119,6780	59,7073	0,0771
18	150,0550	60,1514	0,0419	149,8700	59,9803	0,0916	149,7380	59,7480	0,0922
19	-0,1073	89,7734	0,0739	-0,3056	89,6673	0,0425	-0,2723	89,6164	0,0633
20	29,8974	89,7921	0,0192	29,6393	89,6365	0,0938	29,8827	89,6949	0,0974
21	59,7821	89,8090	0,1451	59,6197	89,5963	0,1180	59,7448	89,6984	0,0829
22	89,9747	89,8366	0,0130	89,7502	89,6318	0,0224	89,7549	89,8062	0,0243
23	119,5780	89,5727	0,0038	119,6110	89,6591	-0,0330	119,9080	89,7462	0,1073
24	150,0850	89,9340	0,1029	149,6690	89,7566	0,0429	149,8310	89,9274	0,0791
25	-0,0053	119,9170	0,3952	-0,1118	119,7420	0,3872	-0,1201	119,6630	0,3959
26	29,9345	119,8710	0,2320	29,7434	119,7380	0,2897	29,8408	119,8610	0,2877
27	59,7335	119,5600	0,0736	59,6932	119,6740	0,0848	59,8507	119,5330	0,0368
28	89,7635	119,7910	0,0655	89,5843	119,5580	0,0490	89,7393	119,5930	0,0096
29	119,2600	119,7610	0,0106	119,5290	119,6750	-0,0450	119,6980	119,5690	-0,0076
30	149,7680	119,8580	-0,0408	149,6030	119,7210	0,0126	149,9290	119,9110	0,0851
31	-0,1496	149,6790	0,0000	-0,1780	149,7140	0,0000	-0,1224	149,6440	0,0000
32	29,7536	149,5770	0,0176	29,7956	149,6380	-0,0257	29,9226	149,6840	-0,0257
33	59,8782	149,8700	-0,0421	59,8489	149,7780	0,0008	59,7943	149,6820	-0,0212
34	89,7024	149,6500	-0,0257	89,6662	149,5850	-0,0087	89,7082	149,4610	-0,0220
35	119,4530	149,3840	-0,0850	119,3720	149,4190	-0,0759	119,7180	149,6950	-0,0517
36	149,5000	149,5960	0,0050	149,4910	149,5670	-0,0796	149,8550	149,7300	0,0393

Kod ovih mjerenja Polaris je pratio samo jedan marker. Podaci su iskorišteni za računanje podataka prikazanih u tablici 7.

Tablica 13. Tri mjerenja sa Polarisom, referentni

T	x1	y1	z1	x2	y2	z2	x3	y3	z3
1	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
2	30,0814	0,1201	0,1355	30,2236	-0,0747	-0,1390	30,0505	0,0128	0,0960
3	60,0205	0,2572	0,0786	60,0216	-0,1541	-0,0357	59,8656	0,0032	0,0802
4	89,8835	0,0023	0,0469	89,7947	-0,2556	0,0397	89,7277	-0,2751	0,0267
5	119,8680	0,1689	0,1413	119,7970	-0,0304	0,2461	119,7560	-0,1870	0,0632
6	149,8520	-0,0003	0,2983	149,8660	0,0000	0,0000	149,9480	0,0000	0,0000
7	-0,0911	29,8553	0,0670	-0,0175	29,6468	-0,1423	-0,0686	29,8303	0,0744
8	29,9611	29,9804	0,3561	29,9958	29,7818	0,2538	29,9565	30,0138	0,4632
9	59,9648	30,0483	0,0165	59,8441	29,8075	0,0391	59,7970	29,8212	0,1911
10	89,8154	30,0347	0,0073	89,8099	29,7264	-0,0150	89,8981	29,7533	0,0469
11	119,7840	29,9135	-0,0967	119,7820	29,9848	0,0659	119,7580	29,6940	0,0473
12	149,8590	30,2625	-0,1443	149,7090	30,6046	0,2966	149,8600	29,8648	0,2256
13	-0,0747	59,7357	0,0864	-0,0794	59,6596	-0,1230	-0,1873	59,6890	0,1751
14	29,7940	59,7073	0,1813	29,7696	59,5058	0,0548	30,0115	59,8906	0,0506
15	59,8130	59,7229	0,1124	59,7173	59,6131	0,0034	59,9356	59,8510	0,1660
16	89,5736	59,8182	0,0225	89,6490	59,7135	0,2192	89,7412	59,7760	0,0041
17	119,7230	60,0217	0,1248	119,7830	59,9268	0,1291	119,7690	59,6992	0,1429
18	149,8690	59,9006	-0,0450	149,7190	60,0866	0,1504	149,7720	59,5865	0,0353
19	-0,1646	89,7611	0,0876	-0,2711	89,5115	-0,0064	-0,2006	89,6081	0,1507
20	29,8986	89,6968	0,0032	29,6916	89,4479	0,0555	29,8762	89,6829	0,1485
21	59,6522	89,7026	0,1169	59,5512	89,5105	0,1656	59,7785	89,6654	0,1322
22	89,3505	89,8511	0,0559	89,6777	89,6425	0,1653	89,7264	89,7970	0,1140
23	119,4090	89,7175	0,1765	119,5030	89,7623	0,0248	119,8580	89,6938	0,1925
24	149,3550	89,9257	-0,0489	149,7200	89,8500	0,3656	149,7370	89,9475	0,3220
25	-0,0833	119,8430	0,4206	-0,1773	119,5640	0,2214	-0,0892	119,6380	0,5762
26	29,7917	119,8590	0,0562	29,6328	119,4930	0,5315	29,9232	119,7230	0,5093
27	59,5623	119,5680	0,0774	59,8164	119,5410	0,2086	59,8645	119,5010	0,0773
28	89,6052	119,8170	0,0662	89,6164	119,4180	0,2585	89,8822	119,4800	0,0627
29	119,6580	119,7750	-0,2227	119,3700	119,5670	0,1447	119,7830	119,6170	0,1359
30	149,8250	119,9060	-0,0400	149,5310	119,8890	-0,0435	149,8680	119,8650	0,1862
31	-0,1732	149,6970	0,0000	-0,3687	149,3840	0,0000	-0,1451	149,5480	0,0000
32	29,6913	149,5150	-0,0856	29,6916	149,5010	0,3146	29,9006	149,5720	0,0680
33	59,7124	149,8620	0,0505	59,7358	149,6420	0,1208	59,8020	149,5370	0,2259
34	89,6342	149,7330	-0,1263	89,5312	149,4260	0,1476	89,5779	149,2720	-0,0350
35	119,3570	149,4140	-0,0733	119,3460	149,4250	0,1739	119,7240	149,6460	0,1461
36	149,3610	149,5650	0,0169	149,3800	149,6490	-0,0954	149,8640	150,8780	0,3163

Kod ovih mjerenja Polaris je pratio oba markera. Podaci su mjereni obzirom na referentni marker na ploči. Podaci su iskorišteni za računanje podataka prikazanih u tablici 7.

Tablica 14. Tri mjerenja sa robotom

T	x1	y1	z1	x2	y2	z2	x3	y3	z3
1	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
2	30,0990	-0,1136	0,6605	29,9022	0,0263	-0,1300	30,0592	-0,4733	0,1873
3	59,9069	0,0781	0,1352	59,9567	-0,2567	0,2382	59,9020	-0,0346	-0,2467
4	89,9555	-0,4699	-0,0493	89,8427	-0,2043	-0,0416	89,8991	-0,0305	-0,1947
5	119,9280	-0,1745	-0,7938	119,8030	0,0730	0,3902	119,9380	0,1901	-0,0003
6	150,0430	0,0000	0,0000	149,9160	0,0000	0,0000	150,0500	0,0000	0,0000
7	0,0836	29,5797	0,3421	-0,0368	29,6686	-0,2687	-0,0028	29,5509	-0,4738
8	30,0547	29,7545	0,2462	29,9475	29,8169	0,3523	29,9077	29,9394	-0,3681
9	59,9796	29,8042	0,0790	59,8494	29,7359	0,1015	59,8737	29,6651	0,0110
10	89,9701	29,6981	-0,2402	89,9347	29,7397	-0,1382	90,1146	29,7612	-0,0153
11	119,9850	29,9151	-0,4257	119,8040	29,8878	-0,1975	119,9860	29,8260	0,0369
12	150,0160	30,0001	-0,8628	149,8910	29,7594	-0,1781	150,1700	29,8975	-0,1698
13	0,2430	59,3585	0,4064	0,0487	59,5117	-0,1960	0,0332	59,2949	0,0946
14	30,1304	59,2980	0,6880	29,8602	59,4159	0,1630	30,1520	59,5129	-0,3909
15	60,0911	59,3357	0,8505	59,8578	59,5894	-0,0658	60,0695	59,4815	-0,0795
16	89,9638	59,8090	0,2783	89,8694	59,4826	0,1135	90,0138	59,5878	-0,1566
17	119,9720	59,5514	0,1673	119,8660	59,7596	0,5554	119,9810	59,5611	0,2251
18	149,9870	59,5655	-0,2106	149,9010	59,6595	0,1449	150,0620	59,5981	0,9277
19	0,1803	89,0861	-0,0495	0,0181	89,3866	-0,2233	-0,0398	89,1192	0,0834
20	30,1576	89,0251	-0,5682	29,9852	89,2743	-0,0141	30,1853	89,1722	0,1666
21	59,9591	89,3401	0,4980	59,9392	89,2735	0,1358	59,8954	89,4329	-0,4296
22	90,2745	89,2419	-0,1118	90,0896	89,1033	0,1350	89,9663	89,5458	-0,1763
23	120,0190	89,3466	0,3220	119,9950	89,1451	-0,2349	120,1110	89,3277	0,3639
24	150,2920	89,2306	0,1320	150,0780	89,2776	-0,2237	149,9960	89,5585	0,0679
25	0,2786	118,9440	0,0458	0,1636	119,0660	-0,1416	0,1446	118,8020	-0,0084
26	30,1974	119,0290	-0,1310	30,0138	119,0540	0,4981	29,9722	119,0300	0,0554
27	60,1224	118,8290	-0,1011	60,0498	119,1330	0,1987	60,3153	118,9780	-0,2822
28	90,0691	119,0990	0,2042	90,1055	118,9600	0,3377	90,0324	119,0750	-0,1900
29	120,0820	119,0280	-0,1547	119,9090	118,9590	-0,2127	120,1530	119,1670	-0,0867
30	150,0620	119,0810	-0,7421	149,9400	118,9140	-0,1729	150,0320	119,2660	0,0359
31	0,2810	148,6860	0,0000	0,1482	148,7700	0,0000	0,2175	148,5430	0,0000
32	30,1936	148,7590	0,3031	30,1630	148,6430	-0,0708	30,2246	148,6380	0,0930
33	60,1651	148,7650	-0,4948	60,1241	148,8150	-0,1297	60,1658	148,8780	0,0973
34	90,1806	148,7450	-0,3431	90,1330	148,5890	-0,2712	90,0870	148,8970	-0,3508
35	120,1060	148,8080	-0,4601	120,0260	148,5730	-0,2815	120,0080	148,9660	-0,1484
36	150,0080	149,0240	-0,2554	150,0650	148,6310	-0,7110	150,1540	148,9850	0,0842

Podaci prikupljeni pri ručnom pomicanju robota. Podaci su iskorišteni za računanje podataka prikazanih u tablici 7.

II. Programski kod

Navedeni kod služi za transformaciju podataka iz Polarisovog koordinatnog sustava u koordinatni sustav kalibracijske ploče:

```
#include<iostream>
#include<cmath>
#include <Eigen/Dense>
#include <Eigen/Geometry>
using namespace std;
using namespace Eigen;


int main() {

	Vector3d X0(2.1214, 143.9914, 41.9806); //Odredivanje prvog
vektora, npr. vektora X0.
	X0.normalize();  //Normalizacija vektora
	cout << " X0 =" << endl << X0 << "\n" << endl;

	Vector3d Y0_pomocni(4.2124, -41.4826, 143.9118); //Odredivanje
pomocnog vektora, npr. pomocni vektor Y0_pomocni.
	Y0_pomocni.normalize();  //Normalizacija vektora
	cout << " Y0_pomocni =" << endl << Y0_pomocni << "\n" << endl;

	Vector3d Z0;
	Z0 = X0.cross(Y0_pomocni);   //Pomocu vektorskog produkta X0 i
Yo_pomocni dobivamo ispravni Z0 vektor.
	Z0.normalize();
	cout << "Z0 =" << endl << Z0 << "\n" << endl;

	Vector3d Y0;
	Y0 = Z0.cross(X0);  //Pomocu vektorskog produkta Z0 i X0 dobivamo
ispravni vektor Y0.
	cout << "Y =" << endl << Y0 << "\n" << endl;

	double OX = -37.4492, OY = -180.6838, OZ = -1109.4384; // Vektor
polozaja.

	Matrix4d T; //Matrica transformacije T, dimenzija 4x4.
	T << X0(0), Y0(0), Z0(0), OX,
	     X0(1), Y0(1), Z0(1), OY,
	     X0(2), Y0(2), Z0(2), OZ,
	     0, 0, 0, 1;
	cout << "T =" << endl << T << "\n" << endl;


	Vector4d P1, P(-37.4492, -180.6838, -1109.4384, 1);
	P1 = T.inverse()*P; //Racunanje koordinata tocke obzirom na novi
koordinatni sustav, pomaknut za vektor polozaja.
	cout << "P1 =" << endl << P1 << endl;
	system("PAUSE");
}
```

Naveden je dio koda koji računa prostorne transformacije iz Polarisovih koordinata u koordinate za vođenje robota:

```
void veza(void *P)
{
//Open the managed segment
  managed_shared_memory shm(open_only, "MySharedMemory");

//Find the vector using the c-string name
MyVector *myvector = shm.find<MyVector>("MyVector").first;
ofstream myfile;
myfile.open ("provjera.txt");
while(1){                  // start:

for (int i=0; i<14; i++){polaris_data_M2[i]=myvector->at(i); };
 Sleep(1000);
 if (polja=="tran"){
 cout<<"Transformacija koordinatnog sustava: "<<endl;
 printf("Podaci sa polarisa marker 2 (relativni), M2: Tx=%f, Ty=%f,
Tz=%f, Q0=%f, Qx=%f, Qy=%f, Qz=%f \n
",polaris_data_M2[0],polaris_data_M2[1],polaris_data_M2[2],polaris_data
_M2[3],polaris_data_M2[4],polaris_data_M2[5],polaris_data_M2[6] );

printf("Podaci sa polarisa za 1. marker: Tx=%f, Ty=%f, Tz=%f, Q0=%f,
Qx=%f, Qy=%f, Qz=%f \n
",polaris_data_M2[7],polaris_data_M2[8],polaris_data_M2[9],polaris_data
_M2[10],polaris_data_M2[11],polaris_data_M2[12],polaris_data_M2[13] );

q.w()=polaris_data_M2[3];
q.x()=polaris_data_M2[4];
q.y()=polaris_data_M2[5];
q.z()=polaris_data_M2[6];
polaris_dcm=q.normalized().toRotationMatrix();
Pm2 << polaris_dcm(0,0), polaris_dcm(0,1),
polaris_dcm(0,2),polaris_data_M2[0],

polaris_dcm(1,0), polaris_dcm(1,1),
polaris_dcm(1,2),polaris_data_M2[1],
polaris_dcm(2,0), polaris_dcm(2,1),
polaris_dcm(2,2),polaris_data_M2[2],
0, 0, 0,1;
cout<<"Pm2 \n"<<Pm2<<endl;
```

```
 offset<< 1, 0, 0, -19.105503,
          0, 1, 0, -1.4255,
          0, 0, 1, -157.6049,
          0, 0, 0, 1;
Pm2=Pm2*offset;
R<<polja_primanje[3],polja_primanje[4],polja_primanje[5];// e1,e2,e3
theta=sqrt(pow(polja_primanje[3],2)+pow(polja_primanje[4],2)+pow(polja_
primanje[5],2));
robot_dcm=AngleAxisf(theta, R.normalized());
Rm2<<robot_dcm(0,0),
robot_dcm(0,1),robot_dcm(0,2),polja_primanje[0]*1000,
robot_dcm(1,0),robot_dcm(1,1),robot_dcm(1,2),polja_primanje[1]*1000,
robot_dcm(2,0),robot_dcm(2,1),robot_dcm(2,2),polja_primanje[2]*1000,
0,0,0,1;
cout<<"Robot \n"<<Rm2<<endl;
T=Rm2*Pm2.inverse();
cout<<"Matrica transformacije T= \n"<<T<<endl;
cout<<"------------Racunanje novih vrijednosti--------------"<<endl;
q.w()=0.1310;
q.x()=-0.7038;
q.y()=0.1221;
q.z()=-0.6871;
cout<<"Q0="<<q.w()<<endl;
cout<<"Qx="<<q.x()<<endl;
cout<<"Qy="<<q.y()<<endl;
cout<<"Qz="<<q.z()<<endl;
polaris_dcm1=q.normalized().toRotationMatrix();
Pm11<<polaris_dcm1(0,0), polaris_dcm1(0,1), polaris_dcm1(0,2),x,
polaris_dcm1(1,0), polaris_dcm1(1,1), polaris_dcm1(1,2),y,
polaris_dcm1(2,0), polaris_dcm1(2,1), polaris_dcm1(2,2),z,
0, 0, 0,1;
cout<<"R=\n"<<Pm11<<endl;
P1=T*Pm11;
cout<<"Nova tocka \n"<<endl;
cout<<"Mat. nove tocke:\n"<<P1<<endl;
       // Podaci za robota (base)
```

```
polja_slanje[0]=P1(0,3)/1000;
polja_slanje[1]=P1(1,3)/1000;
polja_slanje[2]=P1(2,3)/1000;
P1_3x3=P1.block<3,3>(0,0);
angle1=acos((P1_3x3(0,0)+P1_3x3(1,1)+P1_3x3(2,2)-1)/2);
polja_slanje[3]=(P1_3x3(2,1)-P1_3x3(1,2))/sqrt( pow(P1_3x3(2,1)-
P1_3x3(1,2),2) + pow(P1_3x3(0,2)-P1_3x3(2,0),2) + pow(P1_3x3(1,0)-
P1_3x3(0,1),2) )*angle1; polja_slanje[4]=(P1_3x3(0,2)-
P1_3x3(2,0))/sqrt( pow(P1_3x3(2,1)-P1_3x3(1,2),2) + pow(P1_3x3(0,2)-
P1_3x3(2,0),2) + pow(P1_3x3(1,0)-P1_3x3(0,1),2) )*angle1;
polja_slanje[5]=(P1_3x3(1,0)-P1_3x3(0,1))/sqrt( pow(P1_3x3(2,1)-
P1_3x3(1,2),2) + pow(P1_3x3(0,2)-P1_3x3(2,0),2) + pow(P1_3x3(1,0)-
P1_3x3(0,1),2) )*angle1;
cout<<"Koordinate tocke 1:"<<endl;
cout<<"X= "<<polja_slanje[0]*1000<<endl;
cout<<"Y= "<<polja_slanje[1]*1000<<endl;
cout<<"Z= "<<polja_slanje[2]*1000<<endl;
cout<<"e1= "<<polja_slanje[3]<<endl;
cout<<"e2= "<<polja_slanje[4]<<endl;
cout<<"e3= "<<polja_slanje[5]<<endl;
}
if (polja=="save") {
for (int i=0; i<7; i++){polaris_data_M2[i]=myvector->at(i); };
q.w()=polaris_data_M2[3];
q.x()=polaris_data_M2[4];
q.y()=polaris_data_M2[5];
q.z()=polaris_data_M2[6];
polaris_dcm=q.normalized().toRotationMatrix();
Pm2 << polaris_dcm(0,0), polaris_dcm(0,1),
polaris_dcm(0,2),polaris_data_M2[0],
polaris_dcm(1,0), polaris_dcm(1,1),
polaris_dcm(1,2),polaris_data_M2[1],
polaris_dcm(2,0), polaris_dcm(2,1),
polaris_dcm(2,2),polaris_data_M2[2],
0, 0, 0,1;
offset<< 1, 0, 0, -19.105503,
0, 1, 0, -1.4255,
```

```
0, 0, 1, -157.6049,
0, 0, 0, 1;
Pm2=Pm2*offset;
cout<<"RMS1: "<<sqrt(pow(x-Pm2(0,3),2)+pow(y-Pm2(1,3),2)+pow(z-
Pm2(2,3),2))<<endl<<endl;
myfile <<"Tocka 1, razlike: X="<<x-Pm2(0,3)<<",  Y="<<y-Pm2(1,3)<<",
Z="<<z-Pm2(2,3)<<endl;
myfile <<"euclidean: "<<sqrt(pow(x-Pm2(0,3),2)+pow(y-Pm2(1,3),2)+pow(z-
Pm2(2,3),2))<<endl<<endl;
shm.destroy<MyVector>("MyVector");
}
if (polja=="sav1") {
for (int i=0; i<7; i++){polaris_data_M2[i]=myvector->at(i); };
q.w()=polaris_data_M2[3];
q.x()=polaris_data_M2[4];
q.y()=polaris_data_M2[5];
q.z()=polaris_data_M2[6];
polaris_dcm=q.normalized().toRotationMatrix();
Pm2 << polaris_dcm(0,0), polaris_dcm(0,1),
polaris_dcm(0,2),polaris_data_M2[0],
polaris_dcm(1,0), polaris_dcm(1,1),
polaris_dcm(1,2),polaris_data_M2[1],
polaris_dcm(2,0), polaris_dcm(2,1),
polaris_dcm(2,2),polaris_data_M2[2],
0, 0, 0,1;
offset<< 1, 0, 0, -19.105503,
0, 1, 0, -1.4255,
0, 0, 1, -157.6049,
0, 0, 0, 1;
Pm2=Pm2*offset;
cout<<"RMS2: "<<sqrt(pow(x-Pm2(0,3),2)+pow(y-Pm2(1,3),2)+pow(z-
Pm2(2,3),2))<<endl<<endl;
myfile <<"Tocka 1 korigirana, X="<<x-Pm2(0,3)<<",  Y="<<y-Pm2(1,3)<<",
Z="<<z-Pm2(2,3)<<endl;
myfile <<"euclidean: "<<sqrt(pow(x-Pm2(0,3),2)+pow(y-Pm2(1,3),2)+pow(z-
Pm2(2,3),2))<<endl<<endl;
shm.destroy<MyVector>("MyVector");
}
```

```
}//while petlja
myfile.close();
cout<<endl;
       shm.destroy<MyVector>("MyVector");
   return ;
}
```

Server prima podatke sa socket-a:

```
iResult = recv(ClientSocket, recvbuf, recvbuflen, 0);
   if (iResult > 0)
   printf("Bytes received: %d\n", iResult);
          //ispis primljenih podataka
   std::cout << "Received data = " << recvbuf<<"\n";
```

Naveden je dio koda koji prima podatak sa robota u formatu „string“ i pretvara ga u niz podataka tipa „float“.

```
if (recvbuf[0]=='p' && recvbuf[1]=='['){ //poruka započinje sa: p[....]
int i=0;
string s = recvbuf;
string delimiter = ",";     //zarez u stringu je graničnik
size_t pos = 0;
string token;
s.erase(0,2);              //briše prva dva elementa „p"i"["
s.erase(s.length()-1);     //briše zadnji element „]"
s.append(",");
//pretvaranje elemenata u tip „float"
while ((pos = s.find(delimiter)) != string::npos) {
token = s.substr(0, pos);
polja_primanje[i]=stof(token);
s.erase(0, pos + delimiter.length());
i++;}
}
else {
 //ako poruka ne započinje sa „p[", sprema poruku u varijablu „polja"
stringstream pp;
for(int i =0; i<4 ; i++){
pp<<recvbuf[i];
```

```
polja=pp.str();
}
}
```

Navedeni kod pretvara izračunate vrijednosti za vođenje robota iz tipa „float“ u tip „string“. Podatke je potrebno poslati na robata u „string“ formatu:

```
 std::stringstream ss;
ss<<"(";
for( int i = 0; i < 5; i++ )
{ss<<polja_slanje[i];
ss<<",";
 }
ss<<polja_slanje[5]<<")";
slanje=ss.str();
sendbuf = (char*)slanje.c_str();
```

Slanje podataka na robota i ispis poslanih podataka:

```
iSendResult = send( ClientSocket, sendbuf, (int)strlen(sendbuf), 0 );
printf("Data sent: %s \n", sendbuf);
```