3D rekonstrukcija i lokalizacija objekata pomoću robota

Domjanić, Filip

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Filip Domjanić

Zagreb, 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

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

Student:

Filip Domjanić

Zagreb, 2018.

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

Zahvaljujem se mentoru prof. dr. sc. Bojanu Jerbiću i asistentu dr. sc. Bojanu Šekoranji na pomoći, sugestijama i strpljenju prilikom izrade diplomskog rada.

Također se želim zahvaliti obitelji na podršci i razumijevanju tijekom studija.

Filip Domjanić

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

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

DIPLOMSKI ZADATAK

Student: **FILIP DOMJANIĆ** Mat. br.: 0035185973

Naslov rada na hrvatskom jeziku: **3D rekonstrukcija i lokalizacija objekata pomoću robota**

Naslov rada na engleskom jeziku: **3D object reconstruction and localization by robot**

Opis zadatka:

U sklopu zadatka potrebno je povezati laserski mjerač udaljenosti i robotsku ruku kako bi se oblikovao sustav za rekonstrukciju i lokalizaciju objekata. Spajanjem informacija dobivenih od laserskog senzora udaljenosti i prostornih položaja robota moguće je rekonstruirati oblak točaka koji predstavlja objekte sadržane u mjernom prostoru. Usporedbom rekonstruiranog oblaka točaka i izvornog (koji predstavlja objekte) moguće je pronaći objekte od interesa. Potrebno je razviti algoritam koji određuje prostorne transformacije između mjerenog i izvornog oblaka kako bi se odredila lokalizacija objekta u robotskom koordinatnom sustavu. Također, potrebno je izračunati greške sustava (mean, max, STD, RMS) na temelju podataka pohranjenih kroz automatski proces mjerenja pozicija i orijentacija.
Razvijenu primjenu provjeriti koristeći opremu dostupnu u Laboratoriju za projektiranje izradbenih i montažnih sustava.

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

Zadatak zadan:
08. ožujka 2018.

Rok predaje rada:
10. svibnja 2018.

Predviđeni datum obrane:
16. svibnja 2018.
17. svibnja 2018.
18. svibnja 2018.

Zadatak zadao:

prof. dr. sc. Bojan Jerbić

Predsjednica Povjerenstva:

prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS KRATICA

DC	**Direct current**
IP	**Internet Protocol**
TCP	**Transmission Control Protocol**
TP	**Teach Pendant**
OSI	**Open Systems Interconnection**
TTL	**Transistor Transistor Logic**
UART	**Universal Asynchronous Receiver/Transmitter**
UDP	**User Datagram Protocol**
USB	**Universal Serial Bus**
WWW	**World Wide Web**
LED	**Light-Emitting Diode**
CMOS	**Complementary Metal Oxide Semiconductor**
ASCII	**American Standard Code for Information Interchange**
TCP	**Tool Center Point**
ICP	**Iterative Closest Point**
NDT	**Normal distributions transform**
STL	**Stereolithography**
RANSAC	**RANdom Sample Consensus**
DH	**Denavit–Hartenberg**

POPIS OZNAKA

Oznaka	Jedinica	Opis
R	-	Matrica rotacije
T	-	Matrica homogene transformacije
t	-	Vektor translacije
ϕ	°	Kut valjanja
θ	°	Kut poniranja
ψ	°	Kut skretanja

SAŽETAK

U ovom radu je opisan način kako stvoriti 3D oblak točaka određenog objekta pomoću 6-osnog robota i triangulacijskog mjerača udaljenosti, te kako na osnovu dobivenog oblaka točaka indirektno omogućiti pozicioniranje robota u bilo koju ciljanu točku na tom objektu.

Kako bi se oblikovao sustav za rekonstrukciju i lokalizaciju objekata potrebno je povezati laserski mjerač udaljenosti AccuRange AR700, Arduino, Fanuc robota i računalo pomoću TCP/IP protokola, poslužitelj-korisnik vezom. Arduino mikrokontroler ćemo iskoristiti za premošćivanje komunikacije sa senzorom udaljenosti, s RS232 na TCP/IP. Za obavljanje te zadaće Arduinu su potrebni RS232 i Ethernet štitovi, za prilagodbu signala (iz mjernog uređaja i za mjerni uređaj) odnosno za povezivanje na mrežu. Centar ovog sustava će biti robot, on će zahtjevati podatke od mjerača udaljenosti (indirektno preko Arduina), te ih zajedno s podatcima o vrhu alata (svojim podatcima) prosljeđivati računalu, od kojeg nakraju očekuje podatke s kojima će se pozicionirati u željenu točku. Postavke robota za komunikaciju, slanje, prihvaćanje i prilagodbu podataka se namještaju programiranjem u robotovom programskom jeziku, Karelu. Treba napisati program tako da radi u pozadini programa napisanog na robotovom privjesku za učenje i da nastavlja s odvijanjem bez obzira na javljanje nekih grešaka koje bi ga inače zaustavile. Program napisan na privjesku za učenje će biti orijentiran na definiranje gibanja robota, ali u ovisnosti o informacijama koje dobiva ih pozadinskog Karel programa.

Unutar komunikacijske petlje (koja uključuje mjerni uređaj, Arduino mikrokontroler i robota) očekujemo kašnjenje, u nekom dijelu, koje će uzrokovati netočnost podataka zbog čega će ih biti potrebno sinkronizirati. Također nužan uvjet za kvalitetan oblak točaka je kvalitetna kalibracija alata, čije poboljšavanje je najveći izazov ovog rada jer vrh alata nije fizička točka nego točka na laserskoj zraci što komplicira proceduru.

Nakon uspostavljanja i usklađivanja komunikacija te nakon kalibriranja alata, spajanjem informacija dobivenih od laserskog mjerača udaljenosti i prostornih položaja robota moguće je rekonstruirati oblak točaka koji predstavlja objekte sadržane u mjernom prostoru, ali za to je potrebno obraditi podatke. U ovom radu to će biti obavljeno u programskom jeziku Matlab. Nakon što je oblak točaka stvoren moguće ga je pomicati, okretati, brisati mu neke točke, dodavati nove, spajati s drugim oblakom točaka... Usporedbom rekonstruiranog oblaka točaka i izvornog (koji predstavlja objekte) moguće je pronaći objekte od interesa razvijanjem algoritam koji određuje prostorne transformacije između mjerenog i izvornog oblaka kako bi se odredila lokacija željene točke na objektu u robotovom koordinatnom sustavu.

Također da se poveća autonomnost sustava potrebno je i omogućiti pronalazak objekta (bez dodatne opreme) u radnom prostoru robota prije procesa skeniranja.

Nakon pronalaska ciljane točke na objektu mjerenja, mjerene su greške sustava (mean, max, STD, RMS) na temelju podataka pohranjenih kroz automatski proces mjerenja pozicija i orijentacija.

Ključne riječi: robot, kalibracija, oblak točaka, komunikacija, mjerač udaljenosti

SUMMARY

This master thesis describes how to create 3D point cloud of object using a 6 degree of freedom industrial robot and triangulation distance gauge, and how with that point cloud, indirectly, robot can be positioned at any desired point on that object.

To made system for reconstruction and localization of objects, it is necessary to connect laser distance gauge AccuRange AR700, Arduino, Fanuc robot and computer using TCP/IP protocol with server-client communication. Arduino microcontroller will be used to bridge communication with distance sensor, from RS232 to TCP/IP. To perform that task, Arduino needs RS232 and Ethernet shields to adjust signals (from sensor and to sensor) and for connection to network. Center of this system will be a robot, who will require data from sensor (indirectly over Arduino) and forward it together with tool information (robot own data) to computer, from which it will expect information in return. From that information robot will get coordinates in which will move itself. Robot settings for communication, sending, receiving, and data modifications are configured by programming in the Fanuc robotic programming language, Karel. It is necessary to write program that will work in background of program written on the robots teach pendant and will continue with execution despite of some errors that would otherwise terminate it. Program written on teach Pendant will be oriented to defining kind of robot motion and speed, but depending on the information obtained from background Karel program.

Due to the communication loop which includes laser sensor, Arduino microcontroller and robot, there is an expected communication delay which could cause invalid data matching, so system (data) need to be synchronized. Also necessary prerequisite for obtain an accurate point cloud is to have precisely defined Tool Centre Point, whose improvements are biggest challenges of this work because top of tool is not physical point, but 'imaginary' point on laser beam which complicates procedure of calibration.

After communication is established and synchronised and tool is calibrated, it is possible (from information obtained from sensor and spatial position of the robot) to reconstruct point cloud which will represent objects from robot working space, but for that it is necessary to process data. In this work it will be performed in Matlab programming language. Once the point cloud has been created it is possible to move it, rotate, delete some points, add new ones, merge with another point cloud, split... By comparing reconstructed point cloud and original (representing objects) it is possible to find objects of interest by developing algorithm that determines

transformation between measured and original cloud to determine location of desired point on the object in the robot's coordinate system.

Also, to increase autonomy of system, it is necessary to be enable to found object in robot workspace before start of scan process (without any additional equipment).

After finding point of interest on measuring object, system errors (mean, max, STD, RMS) were measured based on the data stored through the automatic position and orientation measurement process.

Key words: robot, calibration, point cloud, communication, distance gauge

1. UVOD

Sposobnost spoznaje i automatsko manipuliranje s objektima je osnovna i uobičajena svrha svakog robota u međudjelovanju s okolinom. Međutim, ako su pozicija, dimenzije ili oblik objekta nepoznati, često ih je potrebno odrediti prije daljnje interakcije. To je moguće ostvariti korištenjem vizijskih sustava koji su danas važna sastavnica suvremenih proizvodnih procesa, no oni su često ograničeni uvjetima okolnog osvjetljenja. Jedna od alternativa za njihovo određivanje je, indirektno, stvaranjem oblaka točaka ciljanog dijela radnog područja. Oblak točak je moguće stvoriti na više načina, npr. pomoću stereovizijske ili RGB-D kamere ili, kao što će biti opisano u ovom radu, pomoću laserskog mjerača udaljenosti koji, doduše, nije namijenjen za tu svrhu već za mjerenja u robotskoj montaži jer daje podatke o udaljenosti za samo jednu točku.

Kombiniranjem robota i laserskog mjerača udaljenosti moguće je stvoriti 3D skener. 3D skeniranje je danas vrlo raširena grana koja se koristi u raznim područjima kao što su robotika, medicina, geodezija i proizvodna industrija odnosno povratno inženjerstvo. Moderni uređaji za skeniranje su u mogućnosti stvoriti vrlo precizne 3D virtualne modele, no iako im se tijekom vremena cijena smanjila i dalje su vrlo skupi što ih čini nepristupačnim institucijama s malim novčanim sredstvima. Stoga je konačan cilj ovog rada 3D skeniranje objekata i izrada njegovog trodimenzionalnog digitalnog modela s postojećem opremom iz fakultetskog laboratorija za robotiku. Uređaj koji će biti iskorišten za funkciju samog skeniranja je uređaj čija je prvotna namjena mjerenje udaljenosti, punog naziva AccuRange AR700 Laser Distance Gauge, a za njegovo pomicanje oko radnog objekta će biti korištena robotska ruka čijoj je upravljačkoj jedinici potrebno omogućiti upravljanje samim mjeračem udaljenosti. Također, nakon što je dobiven oblak točaka objekta (oblak točaka je skup točaka u istom koordinatnom sustavu najčešće definiranim s X, Y i Z koordinatama koje u većini slučajeva predstavljaju oplošje nekog objekta) potrebno je omogućiti usporedbu (preklapanje) tog oblaka s drugim oblakom točaka istog objekta (stvorenog u nekom CAD programu ili dobivenog ovim ili nekim drugim uređajem za skeniranje) te na osnovu usporedbe pozicionirati robota, odnosno vrh laserske zrake, u željenu poziciju koja je odabrana na objektu.

Prvi problem koji se javlja je taj što mjerač udaljenosti koristi isključivo serijsku RS232 komunikaciju, što otežava povezivanje sa suvremenim robotskim upravljačkim sustavima jer je na njima već odavno klasična serijska komunikacija zamijenjena suvremenijim protokolima, poglavito mrežnim sučeljima. Stoga je prvi korak ovog rada osmisliti ”međusklop” koji će

premostiti serijsku komunikaciju s mrežnim TCP/IP protokolom. Taj međusklop bi moglo biti samostojeće računalo no to rješenje nije praktično pa je za posrednika odabran Arduino mikrokontroler koji će se programirati tako da prilagođava signale mjeraču udaljenosti odnosno upravljačkom računalu robota.

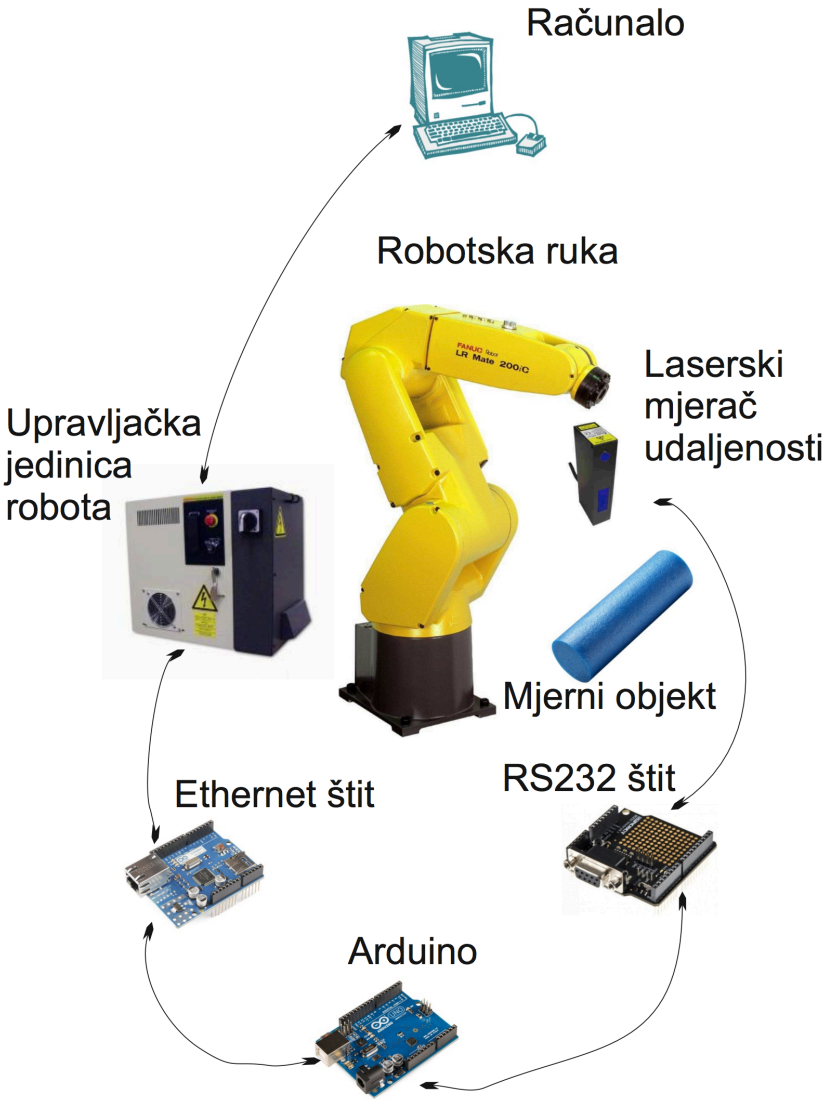
Osim upravljačke jedinice robota i mjerača udaljenosti u liniju komunikacije također je potrebno ubaciti i samostojeće računalo koje će pohranjivati, obrađivati i prikazivati podatke dobivene iz mjerača udaljenosti i podatke o poziciji i gibanju robotske ruke odnosno na kojem će biti prikazan model skeniranog objekta dobiven iz 3D oblaka točaka te koje će po završetku skeniranja poslati robotu podatke na osnovu kojih će pozicionirati točku lasera u željenu točku na objektu kojeg smo skenirali. Da bi se to ostvarilo potrebno je proučiti (i ostvariti) TCP/IP komunikaciju, Fanucov Karel programski jezik za upravljanje robotom, MATLAB-ove funkcije i alate za obradu oblaka točaka te CATIA 'Shape' i 'Surface' funkcije.

Najveći problemi koji su se pojavili u radu su kalibracija vrha alata robota (TCP), neusklađenost podataka iz robota i mjerača udaljenosti, te pronalazak načina kako uspješno preklopiti različite oblake točaka istog objekta različito orijentiranog i smještenog na drugom položaju u radnom prostoru robota.

Očekuje se da će predložena metoda kombiniranja mjerenja laserskog senzora i položaja te orijentacije robotske ruke dati zadovoljavajuće rezultate lokalizacije. Cilj rada je procijeniti je li ovaj pristup izvediv i odrediti pogrešku pozicioniranja pomoću ove metode.

2. DIJELOVI SUSTAVA

Glavni dijelovi sustava su laserski mjerač udaljenosti, Arduino mikrokontroler, robot i njegova upravljačka jedinica te samostojeće računalo. Komunikaciju između mjerača udaljenosti i upravljačke jedinice robota ostvarujemo Arduino mikrokontrolerom, no osim njega potrebni su i Arduino mrežni i RS232 štitovi (engl. Shield) koji se povezuju na njega, te mrežni i serijski kabel. Nakon nabavke svih dijelova preostaje još razviti Arduino program koji će i s aplikacijske strane omogućiti komunikaciju između mjerača udaljenosti i upravljačke jedinice robota. Za povezivanje računala i robotske upravljačke jedinice potreban je samo mrežni kabel, ali je također potrebno razviti program za komunikaciju, izmjenu i usklađivanje podataka.

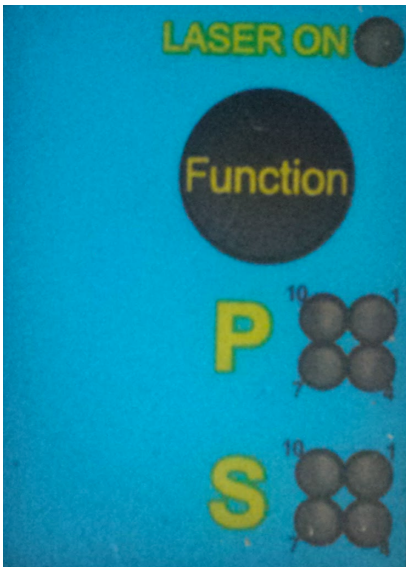


Slika 1. Smjerovi komunikacije

2.1. Laserski mjerač udaljenosti AccuRange AR700

Slika 2. Acuity AR700

AccuRange AR700 je triangulacijski senzor za mjerenje udaljenosti projiciranjem laserske zrake na željni objekt. Odbijeno svjetlo s površine pod određenim kutom pada na CMOS detektor koji zahvaljujući fotoefektu proizvede električni signal u ovisnosti o jačini svjetlosti koja je do njega stigla. Taj signal sadrži informaciju o kutu pod kojim je odbijena zraka pogodila detektor i informaciju o udaljenosti te točke na CMOS detektoru od samog izvora laserskog snopa u uređaju. Signal se potom šalje u mikroprocesor koji iz ta dva podatka proračunava udaljenost objekta. Mjerni raspon senzora je 304,8 mm. Za ispravan rad potreban mu je istosmjerni napon od 24 V, a za komunikaciju s upravljačkim uređajem koristi se RS232 sučelje. Osim što šalje informacije, uređaj može i primati informacije koje imaju za cilj promijeniti postavke uređaja. Te informacije također dolaze putem serijskog priključka, a alternativno je postavke moguće promijeniti i pomoću tipke koja se nalazi na samom uređaju [Slika 3].



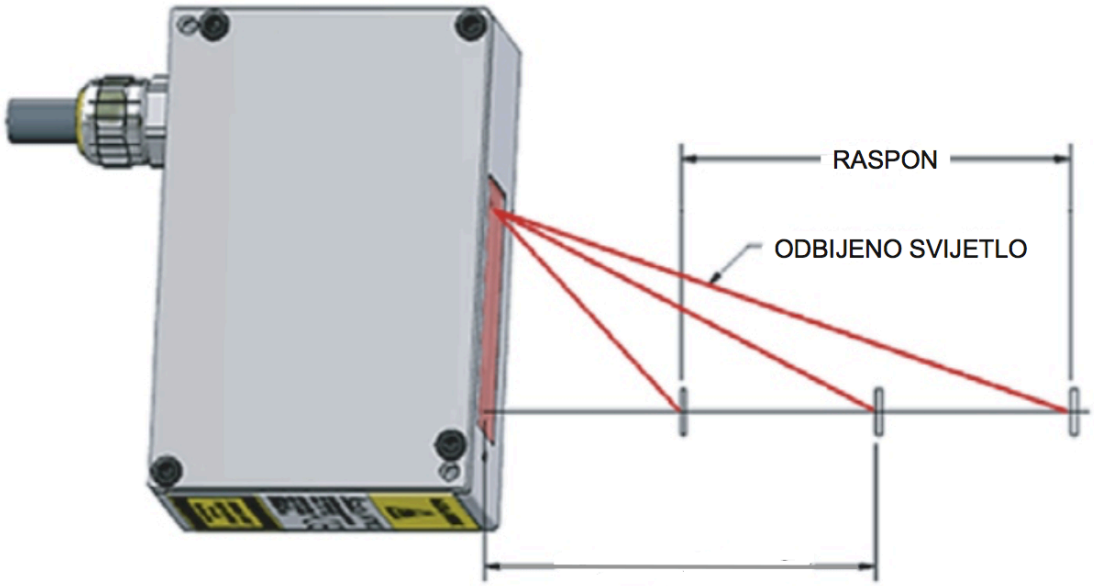
Slika 3. Jedina tipka na uređaju i svjetleće diode čije kombinacije upaljenih i ugašenih označuju koje su postavke zadane

Uređaj ima mnogo parametara koji se mogu mijenjati. Moguće je promijeniti broj izlaznih podataka u jedinici vremena. Najsporija situacija koja se može postaviti je da uređaj pošalje izlaznu informaciju svakih 5 sekundi, a maksimalno možemo postaviti da u sekundi pošalje 9600 informacija o udaljenosti. S odabirom broja izlaznih podataka u jedinici vremena treba biti oprezan jer je moguće da se zada kraći interval slanja podatka od vremena potrebnog da se taj podatak stigne poslati serijskom vezom. U slučaju da se jedan uzorak šalje, a drugi čeka da bude poslan, svaki slijedeći koji dođe će zamijeniti taj uzorak koji čeka na slanje (prethodni signal koji je čekao će biti izgubljen). Za prijenos jednog ASCII koda pri brzini veze od 9600 bit/s potrebno je oko 9 milisekundi što ograničava broj izlaznih podataka na 110 uzoraka u sekundi. Tvornički je postavljeno na 5 uzoraka u sekundi. Proizvođač je također omogućio korisniku da može uključiti ili isključiti opciju uklanjanja okolišnog osvjetljenja. Ako je opcija uklanjanja okolišnog osvjetljenja uključena biti će provedena dva mjerenja jedno s uključenim laserskim snopom i jedno s isključenim. Oduzimanjem ta dva signala efekti okolišnog osvjetljenja će biti uklonjeni. Korisnik također ima mogućnost odabira između kvalitete i kvantitete podataka. Ako želi da mu svi uzorci stignu, to će se osigurati smanjenjem vremena tijekom kojeg je detektor izložen svjetlu. U slučaju odabira kvalitete osigurano je da će detektor biti dovoljno dugo izložen svjetlosti, ali tada je broj uzoraka u jedinici vremena smanjen i teško je odredljiv. Uređaj se može podesiti i tako da na zahtjev korisnika pošalje samo jedan izlazni podatak o udaljenosti, što će se u ovom radu i koristiti.

Ostale karakteristike uređaja su vidljive u tablici 1.

Tablica 1. Karakteristike senzora Acuity AR700

AR700 model -12	
Raspon [mm]	**304,8**
Udaljenost od senzora do centra raspona [mm]	**432**
Linearnost [±µm]	**91**
Rezolucija [µm]	**15,2**
Promjer točke lasera [µm]	**135**
Valna duljina lasera [nm]	**670**
Snaga lasera [mW]	**5**
Boja lasera	**crvena**
Klasa lasera	**3R**
Radni napon [V]	**24 DC**



Slika 4. Prikaz mjernog područja

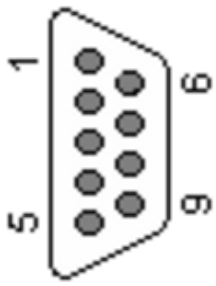
Kao što se može vidjeti na slici 4 raspon mjerenja ne počinje od samog senzora već je odmaknut, a samim time i vrijednost udaljenosti koju senzor daje na izlazu nije udaljenost od samog

uređaja. Da bismo dobili stvarnu udaljenost objekta od senzora, izlaznoj vrijednosti je potrebno dodati udaljenost početka mjernog područja. Prazna kota na slici 4 predstavlja udaljenost središta radnog područja od uređaja s tolerancijom od ±0,25 mm. Ona je dana u dokumentaciji samo radi lakšeg odabira i instalacije uređaja. U centru radnog područja je preciznost mjerenja najveća na cijelom radnom području.

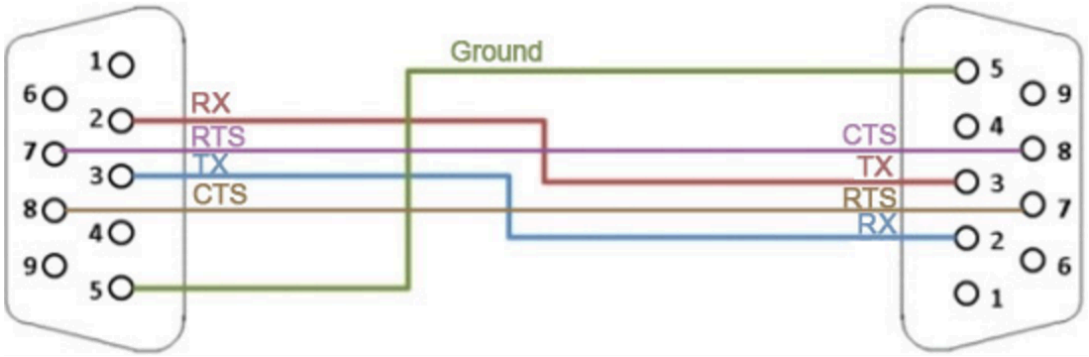
2.1.1 Upravljanje uređajem

Upravljanje se odvija serijskom RS232 komunikacijom. Uređaj posjeduje ženski DE-9 priključak i koristi kontakte 2, 3 i 5, a kontakte 7 i 8 opcionalno prema potrebi korisnika.

Tablica 2. DE-9 Priključak uređaja

Priključak (ženski)	**Kontakt**	**Boja**	**Opis funkcije**
	1	-	Ne koristi se
	2	Zelena	Slanje podataka (TX)
	3	Žuta	Primanje podataka (RX)
	4	-	Ne koristi se
	5	Crna	GND
	6	-	Ne koristi se
	7	Ljubičasta	(dozvola za slanje) CTS
	8	Plava	(Zahtjev za slanje) RTS
	9	-	Ne koristi se

Za uspostavljanje veze između mjerača udaljenosti i Arduinovog RS232 štita potreban je null-modem kabel. On služi za izravno serijsko povezivanje dvaju uređaja. Razlika između običnog i null modem kabela je u zamijenjenim kontaktima 2 i 3 te 7 i 8 [Slika 5]. Kontakti 7 (CTS) i 8 (RTS) služe za međusobno informiranje pošiljatelja i primaoca o trenutnom stanju. Pošiljatelj preko RTS kontakta traži dozvolu za slanje podataka te na svom CTS kontaktu očekuje potvrdan odgovor primatelja.

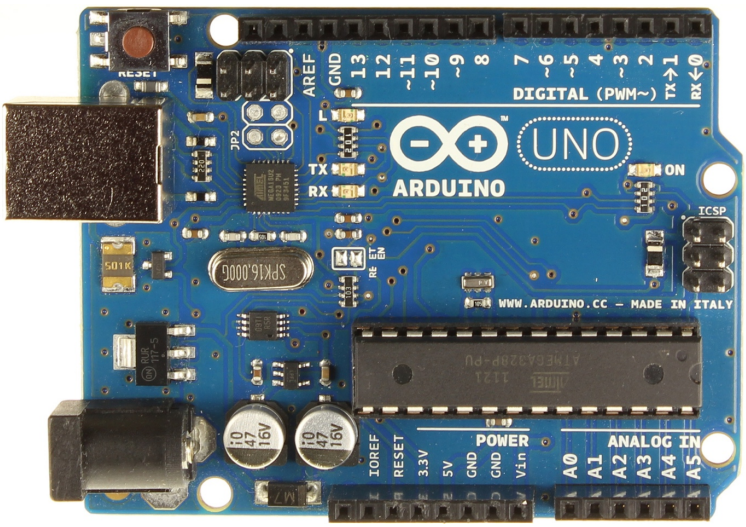


Slika 5. Null modem kabel

2.2. Arduino

Arduino je elektronička platforma otvorenog koda (engl. Open Source Platform) temeljena na sklopovlju (engl. Hardware) i programskoj podršci (engl. Software) koja je prilagodljiva i jednostavna za korištenje. Malene dimenzije obrnuto su proporcionalne mogućnostima realizacije mnogobrojnih tehničkih sustava.

Na Arduino elektroničkoj pločici je 8-bitni ATmega328P mikrokontroler s pripadajućim komponentama koje omogućavaju programiranje i povezivanje s mnogobrojnim senzorima, modulima, aktuatorima i sl. Uređaj je moguće napajati iz univerzalne serijske sabirnice (engl. Universal Serial Bus – USB) ili drugim vanjskim izvorom istosmjernog napona od 5 do 12 V. Velika prednost Arduina je normiran raspored kontakata koji omogućava lako povezivanje s dodatnim modulima. Kontakti 0 i 1 služe za serijsku TTL komunikaciju.

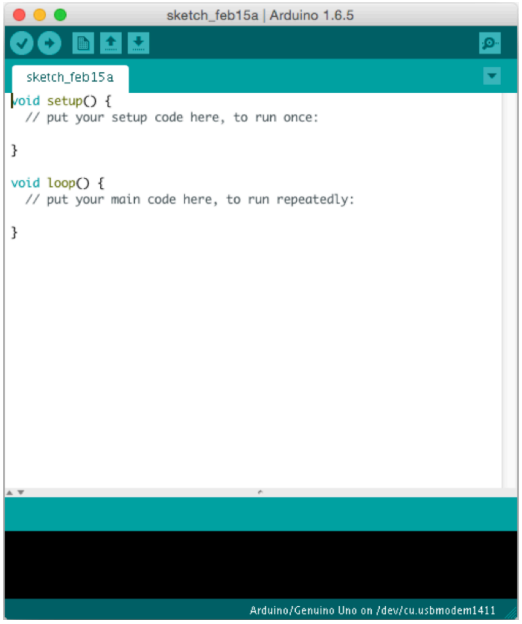


Slika 6. Arduino Uno

Tablica 3. Tehničke karakteristike Arduina

Mikrokontroler	**ATmega328P**
Radni napon	5 V
Napon napajanja (preporučeni)	7-12 V
Digitalni ulazi/izlazi	14
PWM digitalni ulazi/izlazi	6
Analogni ulazi	6
Struja na ulazu/izlazu	20 mA
Struja na 3,3 V	50 mA
Flash memorija	32 kB (0,5 kB koristi bootloader)
SRAM	2 kB
EEPROM	1 kB
Oscilator	16 MHz
Duljina	68,6 mm
Širina	53,4 mm
Masa	25 g

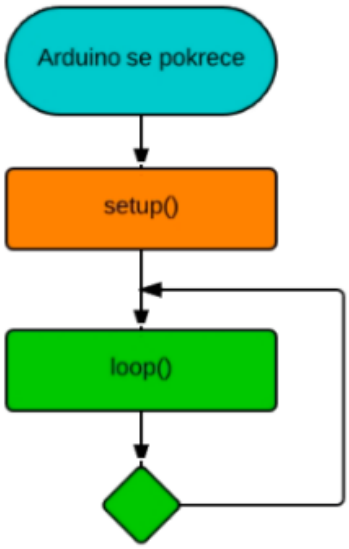
Mikrokontroler se programira putem Arduino programskog jezika nazvanog Arduino IDE koji je napisan u programskom jeziku Processing, isti je kombinacija Jave, C-a i C++-a. Temeljen je na C++ knjižnici 'Wiring' koja čini uobičajene ulazno izlazne operacije veoma jednostavnim. Arduino IDE ujedno služi za pisanje programa i za prenošenje samog programa na mikrokontroler. Za prenošenje koda na mikrokontroler, Arduinu za razliku od većine ostalih mikrokontrolera nije potreban fizički programator. Na samom Arduinu nalazi se Arduino 'bootloader', mali program koji služi za prenošenje programa na sam mikrokontroler bez dodatnog uređaja.



Slika 7. Arduino IDE

2.2.1 Programiranje u Arduinovom programskom jeziku

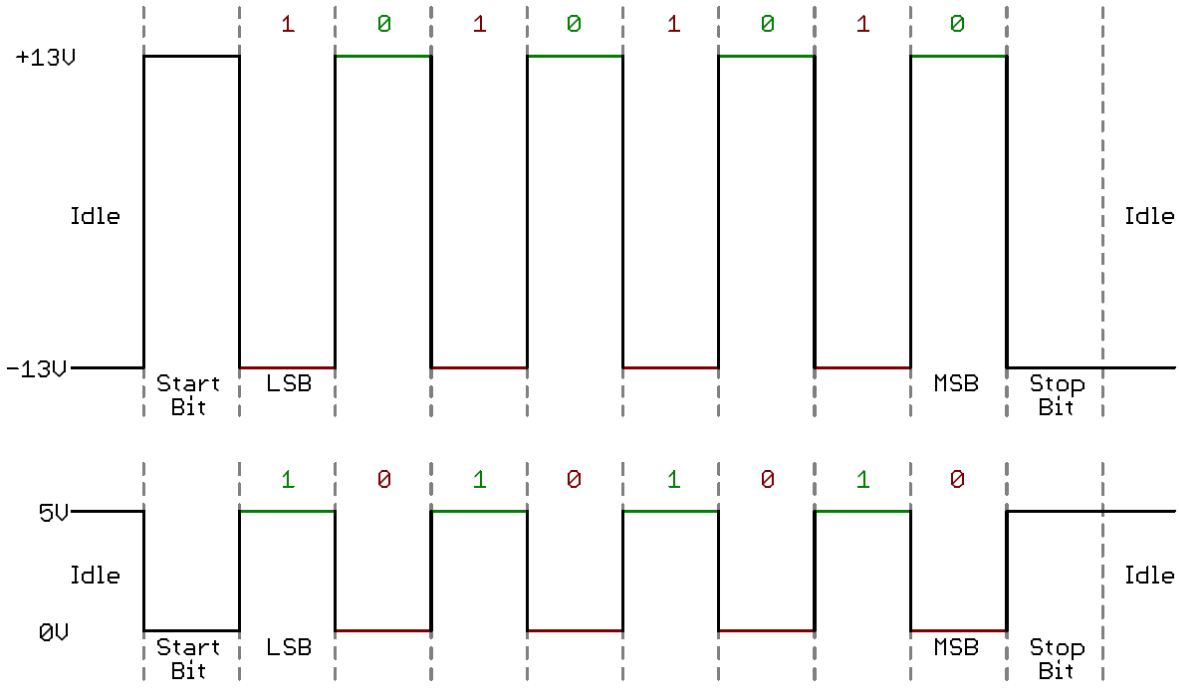
Na samom početku kreiranja novog programa mora se, ukoliko je to potrebno, uvesti određena vanjska knjižnica naredbi. U tim knjižnicama je veliki broj prethodno napisanih naredbi u C++ programskom jeziku koje uvelike olakšavaju i ubrzavaju proces programiranja. Ako je primjerice potrebno izračunati kvadratni korijen nekog broja, nije potrebno unositi kompliciranu algoritam za takvu operaciju, dovoljno je pozvati knjižnicu *math.h* u kojoj se nalazi široka skupina matematičkih algoritama, kvadratni korijen jednostavno se računa naredbom *sqrt()*.
Slijedeći korak je deklariranje varijabli. Varijable mogu biti različitih tipova, ovisno o tome kakvog tipa je podatak koji u nju pohranjujemo. Najčešće korištene varijable su slijedećih tipova: 'Interger' (cjelobrojni broj), 'Float' (decimalni), 'Char' (znak), 'String' (skup znakova - riječ).
Svaki Arduino program mora imati dvije glavne naredbe kako bi se ispravno izvršavao: *setup()* i *loop()*. Naredba *setup()* koristi se za postavljanje ulazno-izlaznih priključaka (engl. I/O ports) kao što su svjetleće diode (engl. Light Emitting Diode), senzori, motori, serijski priključci ili pak uspostavljanje neke veze, npr. serijske ili mrežne. *setup()* funkcija se izvršava samo jednom i to nakon što je Arduino pokrenut ili ponovno pokrenut (engl. restart). Po izvršenju *setup()* funkcije program nailazi na *loop()* funkciju, riječ je o beskonačnoj petlji u kojoj se kontrolira ulazno izlazne priključke pomoću uobičajenih funkcijskih petlji, primjerice: *if()*, *for()*, *while()*, *switch()*.



Slika 8. Dijelovi programskog koda

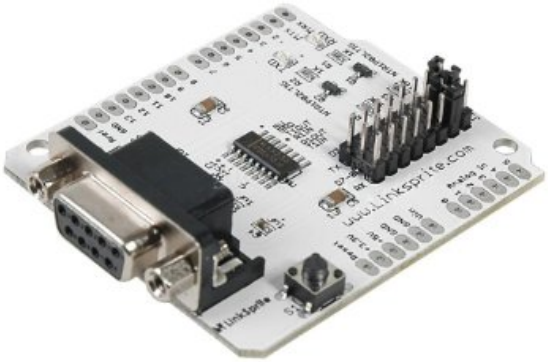
2.3. RS232 štit

Serijska veza je proces slanja podataka preko komunikacijskog kanala slijedno bit po bit. Norma uređaja određuje raspon napona za logička stanja 'jedan' i 'nula'. Logičko stanje 'nula' određuju naponi između +3 V i +25 V, a za logičko stanje 'jedan' naponi moraju biti između -3 V i -25 V. Naponi između -3 V i +3 V su u području zabranjene zone. Na većini računala naponi se u području između -13 V i +13 V. Visoki naponi omogućuju prijenos podataka na veće udaljenosti. Najmanje značajan bit (engl. Least Significant Bit - LSB) se kod serijske komunikacije šalje prvi, a najviše značajan bit (engl. Most Significant Bit - MSB) zadnji. Osim korisne informacije, uređaj šalje "start bit" i "stop bit" koji označavaju početak i kraj poruke te bit za provjeru valjanosti poruke. Serijski priključci mjerača udaljenosti i Arduina su kompatibilni samo aplikacijski, signali su im istih oblika, ali različitih vrijednosti. Arduino koristi UART, univerzalni asinkroni prijemnik/odašiljač koji se povezuje zajedno s RS232. UART prevodi podatke između paralelnih i serijskih formata te može podešavati brzine prijenosa i oblik slanja podataka. Za pravilan rad, UART na strani primatelja i na strani odašiljača mora biti isto podešen. Takva komunikacija na Arduinu je TTL serijska. Vrijednosti signala u serijskoj TTL komunikaciji su u području između 0 V i vrijednosti napona napajanja. Vrijednost napona za logičko stanje 'jedan' je vrijednost napona napajanja, dok je logičko stanje 'nula' određeno s 0 V.



Slika 9. TTL i RS232 signal

Da bismo uspješno povezali uređaj i Arduino potrebno je prilagoditi komunikacijske signale. Prvo se obavlja logička operaciju NE tj. komplementiraju se logička stanja, a zatim im se mijenja vrijednost napona. Najjednostavniji način za to je primjena MAX-232 integriranog kruga između spomenuta dva uređaja. MAX232 je temeljni dio RS232 štita koji prilagođava RS232 signal u TTL signal i obrnuto. Kada na ulaz dobije TTL logičko stanje 'nula' onda na izlazu vlada napon raspona +3 V i +15 V, a kada na ulaz dovedemo TTL logičko stanje 'jedan', na izlazu vlada napon između -3 V i -15 V. Analogna je prilagodba RS232 u TTL. Dovođenjem napona između +3 V i 15 V na ulaz, na izlazu će biti TTL logičko stanje 'nula', a dovođenjem napona između -3 V i -15 V na izlazu ćemo dobiti logičko stanje 'jedan'.

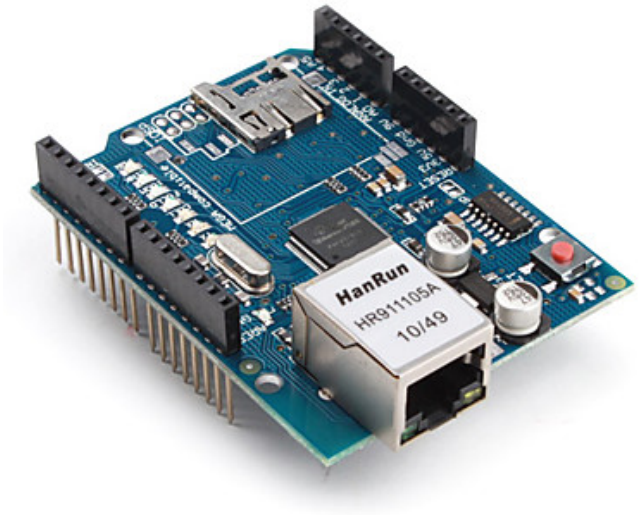


Slika 10. RS232 štit

Tablica 4. Razlike u naponima logičkih stanja

Logičko stanje	**RS232 napon**	**TTL napon**
“0”	+3 V do +15 V	0 V
“1”	-3 V do -15 V	5 V

2.4. Mrežni štit



Slika 11. Ethernet štit

Arduino mrežni štit (engl. Ethernet Shield) omogućuje Arduinovo spajanje na mrežu. Zasnovan je na Wiznet W5100 Ethernet čipu koji omogućava mrežnu komunikaciju koristeći TCP i UDP protokole. Podržava do četiri istovremena socket povezna kanala. Za pisanje programa koji koriste ovaj štit dostupna je Ethernet knjižnica. Povezivanje s Arduinom je izvedeno tako da su svi kontakti s Arduinove elektroničke pločice dostupni i na priključenom štitu, napajanje štita je također riješeno kroz spomenute kontakte. Time je omogućeno spajanje modula uz zadržavanje osnovnog rasporeda kontakata. Mrežni štit ima standardni RJ-45 mrežni priključak, s integriranim transformatorom i PoE (engl. Power over Ethernet) mogućnostima. Na elektroničkoj pločici se nalazi i utor za micro-SD karticu, koja se koristi za čuvanje datoteka koje će biti poslane preko mreže. Arduino komunicira sa W5100 i SD karticom SPI komunikacijom (engl. Serial Peripheral Interface). Taj priključak se nalazi na kontaktima 11, 12 i 13. Kontakt broj 10 se koristi za odabir W5100, a kontakt broj 4 za SD karticu. Iz tog razloga se ti kontakti ne mogu koristiti kao standardni ulazi/izlazi. Pošto W5100 i SD kartica dijele isti SPI (engl. Serial Peripheral Interface) priključak, ne mogu se koristiti u isto vrijeme. Tipka za ponovno pokretanje (engl. Reset) na štitu resetira i W5100 i osnovnu Arduino elektroničku pločicu. Na elektroničkoj pločici se nalazi nekoliko LED indikatora:

- PWR: označava da su Arduino i štit uključeni

- LINK: označava prisustvo mrežne konekcije i treperi kada štit šalje ili prima podatke
- FULLD: označava da mreža istovremeno izmjenjuje podatke u oba smjera (engl. Full - Duplex)
- 100M: označava prisustvo 100 Mb/s mrežne konekcije (u suprotnom brzina je 10 Mb/s)
- RX: treperi kada štit prima podatke
- TX: treperi kada štit šalje podatke
- COLL: treperi u slučaju kolizije na mreži

Postoje dvije klase *korisnik (engl. client)* i *poslužitelj (engl. server)*. Klasa *korisnik* se koristi za kreiranje korisnika na Arduinu koji se spaja na neki vanjski poslužitelj i s njime razmjenjuje podatke. Klasa *Poslužitelj* se koristi za kreiranje poslužitelja na Arduinu koji šalje i prima podatke od vanjskih korisnika koji su priključeni na njega.

2.4.1 TCP/IP komunikacija

TCP/IP je oznaka za grupu protokola pomoću kojih se izmjenjuju podatci između računala. Naziv potječe od dva najvažnija protokola te skupine TCP (engl. Transmission Control Protocol) te od IP (engl. Internet Protocol) protokola. TCP/IP omogućava povezivanje mrežnih čvorova određivajući kako se podatci moraju konvertirati u pakete, adresirati, slati, prenositi te primati na odredištu. TCP/IP spada u treći i četvrti sloj OSI referentnog modela (engl. Open Systems Interconnection Basic Reference Model). OSI je najkorišteniji apstraktni opis arhitekture mreže. Opisuje komunikaciju sklopovlja, aplikacija i protokola pri mrežnim komunikacijama. Koriste ga proizvođači pri projektiranju mreža, kao i stručnjaci pri proučavanju mreža.

Fizički sloj definira električka i fizička svojstva mrežnih uređaja, naponske razine, brojeve kontakata te uređaje kao što su ponavljači, mrežni koncentratori itd.

Podatkovni sloj kontrolira razmjenu podataka između mrežnih uređaja i korigira možebitne greške na fizičkom sloju.

Mrežni sloj je zadužen za pretvaranje logičke IP adrese u fizičku MAC adresu kako bi podatci stigli od jednog mrežnog čvora do drugog.

Prijenosni sloj se brine o paketima koji putuju između dva računala. Primjeri protokola na prijenosnom sloju su TCP i UDP. U slučaju nestanka nekog paketa, TCP će zatražiti od pošiljatelja da ponovno pošalje taj isti paket

Sloj sesije bavi se uspostavom i sinkronizacijom veze između krajnjih korisnika.

Prezentacijski sloj služi za izvođenje kodiranja za sustav koji koristimo npr. TXT-datoteke na Unix-u, MacOS-u ili Windowsima na različite načine označavaju prelazak u novi red. Prezentacijski sloj zadužen je za usklađivanje kodiranja.

Aplikacijski sloj je najbliži korisniku, pruža mrežne usluge korisničkim aplikacijama. Od ostalih slojeva OSI modela razlikuje se po tome što ne pruža usluge drugim slojevima, već samo aplikacijama van OSI modela.

Tablica 5. OSI model

OSI referentni model
7. aplikacijski sloj
6. prezentacijski sloj
5. sloj sesije
4. prijenosni sloj
3. mrežni sloj
2. podatkovni sloj
1. Fizički sloj

2.4.1.1 Korisnik/poslužitelj (engl. Client/Server) komunikacija

Svako računalo i usmjernik (engl. Router) ima jedinstvenu IP adresu, 32 bitni broj, podijeljen u 4 grupe, od kojih svaka sadrži jedan 8 bitni broj; na primjer, 192.168.123.55. Te adrese su nužne da bi se paketi upućeni s izvorišnog računala mogli preusmjeriti do odredišnog. Osnovna je zamisao da postoji jedno centralno računalo koje ima ulogu poslužitelja i koje obavlja dio posla te pruža podatke i izvršava zahtjeve drugog računala koje nazivamo korisnik, odnosno računalnog programa na tom računalu. Takvim rješenjem računalo koje postavlja zahtjeve

(korisnik) zapravo ima ulogu korisničkog sučelja za program koji se izvršava na centralnom računalu, poslužitelju. Budući da na jednom poslužitelju najčešće ima više poslužilačkih aplikacija koje nude usluge korisnicima (web server, ssh, ftp, itd.) potrebno je razlikovati različite aplikacije na poslužitelju. To se čini pomoću jednog broja koji se naziva 'port' i koji je pridružen svakoj poslužiteljskoj aplikaciji. Kada korisnik želi započeti komunikaciju s nekim poslužiteljem, on otvara utičnicu (engl. socket). Socket je apstrakcija mrežne programske podrške koji vrši komunikaciju preko mreže. Sa stanovišta aplikacijskog programera komunikacija se sastoji od pisanja u socket i čitanja iz njega. Može ga se opisati kao jedan od dva kraja na kanalu komunikacije između poslužitelja i korisnika.

Principi rada:

- Aplikacija (na poslužitelju) radi na određenom uređaju (zadanom s IP adresom) i "osluškuje" na unaprijed određenom portu
- Korisničkoj aplikaciji je poznata adresa poslužitelja i port na kojem sluša te mu šalje zahtjev za uspostavu komunikacije (pokušava otvoriti socket za komunikaciju)
- Kada je poslužitelj spreman komunicirati s korisnikom, on otvori novi socket (različit od onog na kojem osluškuje) te sada poslužitelj i korisnik komuniciraju kroz tako otvoreni kanal komunikacije.
- Kada je komunikacija završena i poslužitelj i korisnik zatvaraju svoje sockete i na taj način se zatvaraju kanali komunikacije.

2.5. Fanuc LR Mate 200iC5L

Fanuc robot LR Mate, serije 200iC/5L manji je robot iz Fanucove ponude. Idealan je za razne primjene u automatizaciji. Ima 6 stupnjeva slobode gibanja, nosivost mu je do 5 kg, ponovljivost do ±0,03 mm. Za razliku od osnovnog modela ovaj robot ima veći radni doseg. Na taj robot će biti postavljen mjerač udaljenosti. Tehničke specifikacije su mu sljedeće:

Tablica 6. Tehničke karakteristike Fanuc robota LR Mate

Broj osi	6
Masa [kg]	29

Doseg [mm]		892
Točnost ponavljanja [mm]		±0,03
Nosivost [kg]		5
Opseg gibanja [°]	J1	340
	J2	230
	J3	373
	J4	380
	J5	240
	J6	720
Maksimalna Brzina [°/s]	J1	270
	J2	270
	J3	270
	J4	450
	J5	450
	J6	720

Slika 12. Fanuc LR Mate 200iC5L

Upravljanje robotom vrši upravljačka jedinica R-30iA Mate na brzom operativnom sustavu koji se nije sukladan s Windows operativnim sustavima. Može upravljati s do 40 stupnjeva slobode gibanja. Pokrene se u manje od jedne minute. Za komunikaciju s drugim uređajima koristi mrežni sustav (engl. Ethernet) s RJ-45 priključkom.

Slika 13. R-30iA upravljačka jedinica

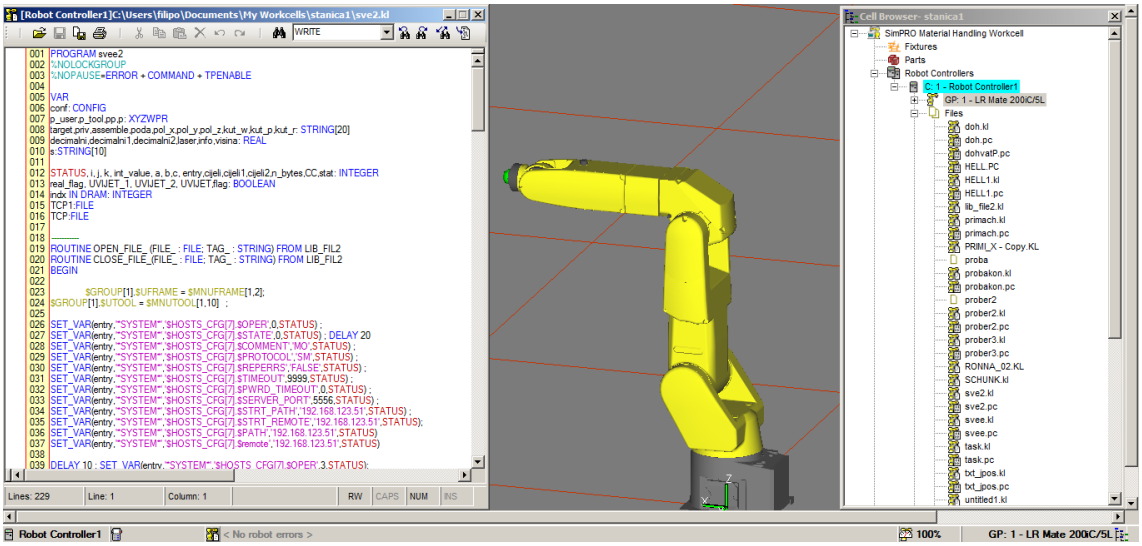
Programiranje robota se vrši preko privjeska za učenje. Privjesak za učenje je složeni ručni uređaj preko kojega se mogu kreirati novi programi, učitavati i prepravljati postojeći, može se direktno pomicati robota, obavljati kalibracije te mnoge druge funkcije. Ima zaslon u boji na kojem se može otvoriti do tri prozora, što uvelike olakšava programiranje i praćenje interakcija.

Slika 14. Privjesak za učenje, iPendant

2.5.1 Programiranje Fanuc robota

Robotov privjesak za učenje omogućava rješavanje raznih zadataka, od najosnovnijih kao što su odabir vrste gibanja, kontroliranja brzine, pohranjivanja koordinata vrha alata, pa do nešto zahtjevnijih kao što je rad s petljama, palatalizacija ili rad s vizijskim sustavima i mnogih drugih. No, neke operacije, kao što je npr. komunikacija s računalom, se na njemu ipak ne mogu obaviti te se one rješavaju programiranjem pozadinskih programa u programskom jeziku KAREL.

KAREL je programski jezik niske razine specifičan za Fanuc robote, strukture vrlo slične programskom jeziku PASCAL. Karel programi se najčešće pišu u programskom paketu ROBOGUIDE u kojem se program prije prevađanja u strojni jezik i slanja u upravljačku jedinicu robota može testirati u virtualnom okruženju na računalu. Generiranje strojnog koda (.pc datoteke) i slanje u upravljačku jedinica robota je maksimalno pojednostavljeno te se ostvari u samo nekoliko klikova. Nakon što je datoteka uspješno poslana upravljačkoj jedinici robota, tamo postaje vidljiva te ju je moguće pozvati na izvršavanje u programu napisanom na privjesku za učenje.



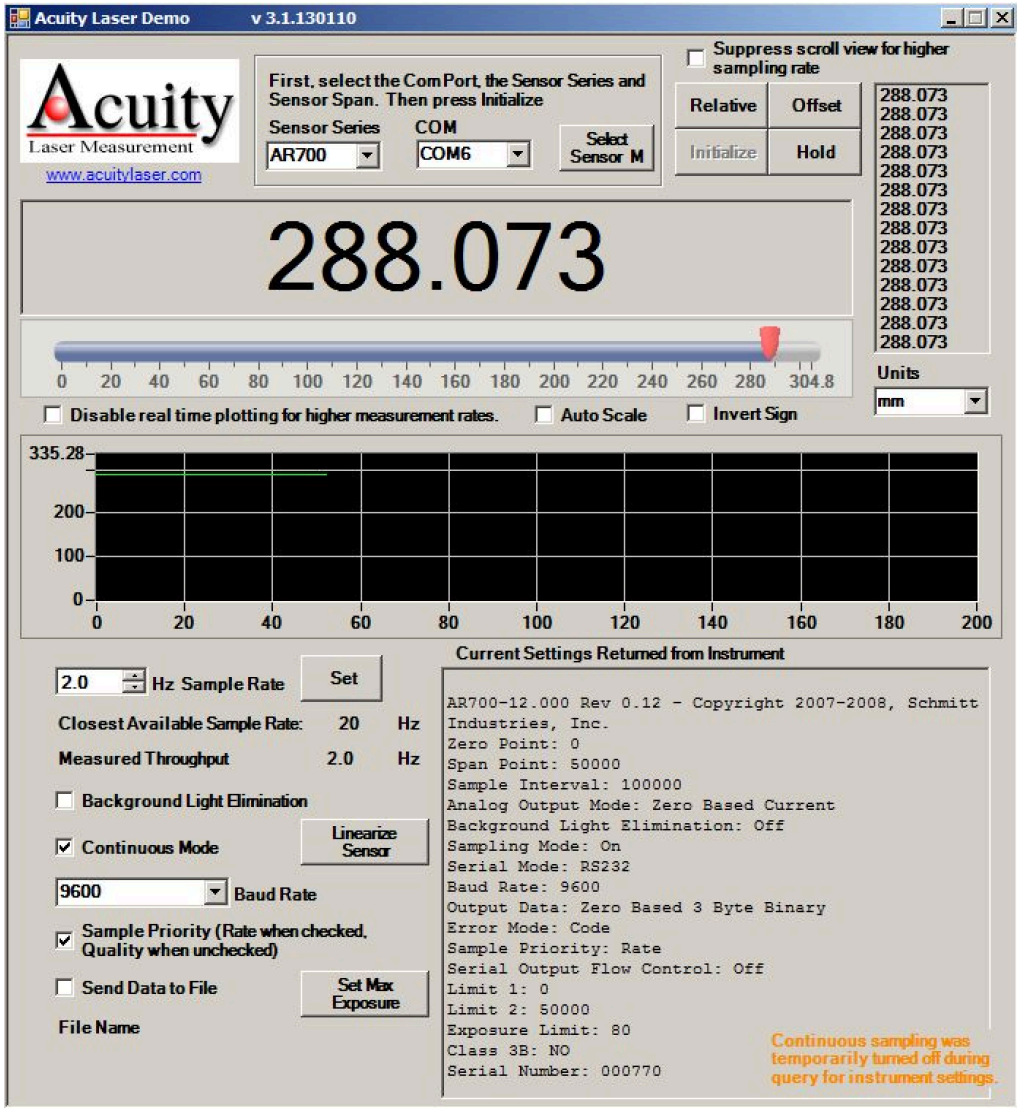
Slika 15. Karel program u programskom paketu Roboguide

2.6. Računalo

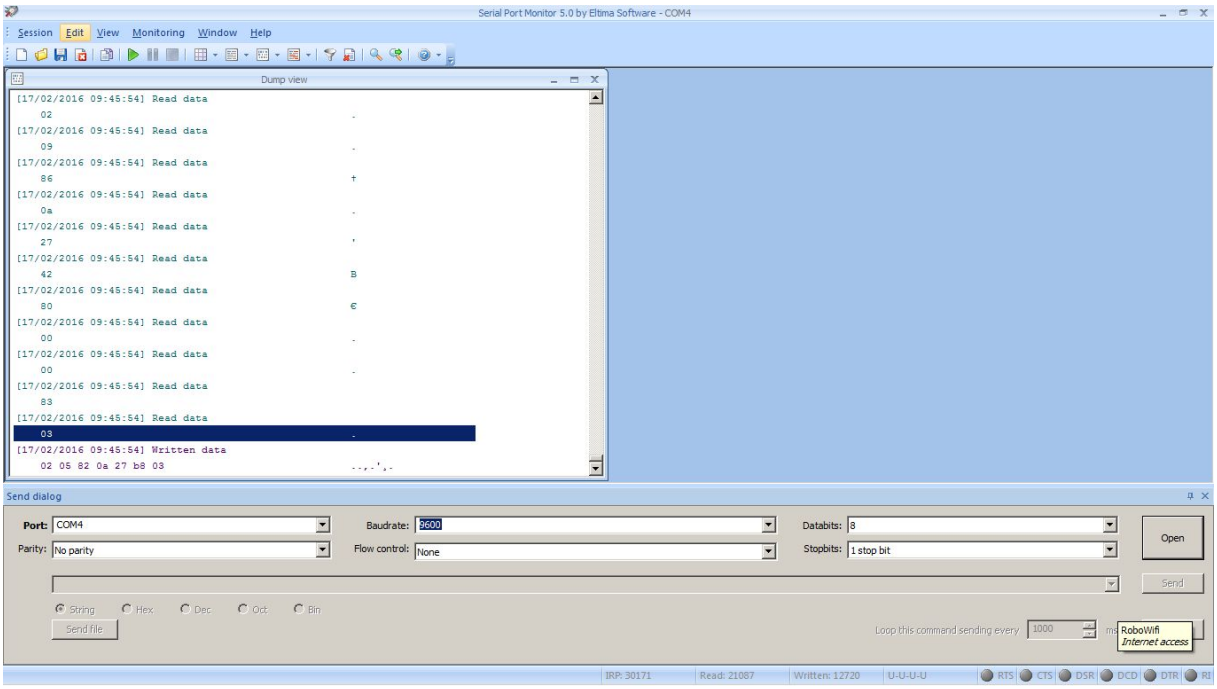
Kao što je već spomenuto računalo će u ovom sustavu prihvaćati, obrađivati, spremati i prikazivati dobivene podatke, a te zadatke ćemo riješiti pomoću MATLAB programskog paketa i pomoću CAD programa CATIA.

3. RAZVOJ ARDUINO UPRAVLJAČKOG PROGRAMA ZA MJERNI UREĐAJ PREKO TCP/IP PROTOKOLA

Da bi se moglo krenuti s razvojem Arduino programa prvo je potrebno shvatiti način upravljanja i komunikaciju sa uređajem. Na internetskoj stranici proizvođača dostupna je demonstracijska aplikacija – Acuity Demo, koja se koristi za probno konfiguriranje senzora i za prikaz dobivenih podataka. Za vrijeme slanja naredbi iz 'Acuity Demo' aplikacije u uređaj, računalo bi u pozadini imalo pokrenutu aplikaciju 'Serial Monitor' koja bi pratila i spremala poruke koje prolaze serijskom vezom. To je jedan od načina da se stekne uvid u poruke koje uređaj očekuje i kakve šalje uređaju koji s njome upravlja.



Slika 16. PowerCube aplikacija



Slika 17. Serial Port Monitor aplikacija

3.1. Naredbe za laserski mjerač udaljenosti AccuRange AR700

Prije početka mjerenja udaljenost, uređaj je potrebno podesiti prema potrebama. Slanjem naredbe ***Sx***, (na mjesto x dolazi broj), određujemo broj poslanih uzoraka u sekundi. Npr. ako uređaju pošaljemo ***S40***, tom naredbom smo broj uzoraka poslanih u sekundi postavili na 5000. Da bismo postavili željen broj uzoraka u sekundi, taj broj moramo podijeliti s 200000 te dobiven broj šaljemo u kombinaciji sa znakom S. (200000/5000=40). Maksimalan broj uzoraka po sekundi je oko 9434, a minimalan 0,5.

Eliminiranje pozadinskog osvjetljenja uključujemo slanjem naredbe ***L1***. Kada je ta opcija uključena senzor za jedan odaslani podatak o udaljenosti obavlja dva mjerenja jedno s uključenim, a jedno s isključenim laserskim snopom, što prepolovljuje maksimalni broj uzoraka u sekundi (oko 4717). Eliminaciju isključujemo slanjem naredbe ***L2.***

Ciklus mjerenja ne traje uvijek isto. Kada se mjeri udaljenost tijela koja imaju slabije reflektirajuću površinu ili onih koja se nalaze na udaljenijem rubu mjernog područja, tada uređaj produljuje vrijeme izloženosti senzora reflektiranom svijetlu. Kod takve situacije moguće je da to vrijeme izloženosti traje dulje neko što bi trebalo po zahtjevu postavljenom ***S*** naredbom (broj uzoraka u jedinici vremena). Da bi se izbjegao taj svojevrsni konflikt, uvedena je naredba

prioriteta. Tom naredbom odlučujemo što nam je važnije, da li kvaliteta mjerenja ili ispoštovanje broja uzoraka zadanih ***S*** naredbom. Slanjem naredbe ***P1*** u uređaj odabrali smo kvalitetu, broj uzoraka u sekundi će se smanjiti po potrebi i nije upravljiv. Ako nam je važno da stignu svi podatci koji su zadani ***S*** naredbom, onda se moramo odlučiti za ***P2*** naredbu, ali tada postoji mogućnost za smanjenjem kvalitete podataka (preciznosti mjerenja). Nakon što smo postavili uređaj u potrebni način rada, ako želimo zadržati tu konfiguraciju i nakon gašenja uređaja moramo ju spremiti u trajnu memoriju uređaja. S obzirom da to uređaj ne radi automatski, spremanje ostvarujemo naredbom ***W1234***, čije korištenje treba izbjegavati u programskim petljama, jer je predviđeni trajni vijek memorije milijun zapisivanja. Ako promijenimo neke parametre, a nismo ih zapisali u memoriju i poželimo vratiti stare postavke koje smo zapisali u memoriju, to možemo ostvariti naredbom ***R.*** Uvid u opcije koje su aktivirane ostvarujemo naredbom ***V1234***. Na tvorničke postavke uređaj vračamo slanjem naredbe ***Q8.***

```
AR700-12.000 Rev 0.12 - Copyright 2007-2008, Schmitt
Industries, Inc.
Zero Point: 0
Span Point: 50000
Sample Interval: 100000
Analog Output Mode: Zero Based Current
Background Light Elimination: Off
Sampling Mode: On
Serial Mode: RS232
Baud Rate: 9600
Output Data: Zero Based 3 Byte Binary
Error Mode: Code
Sample Priority: Rate
Serial Output Flow Control: Off
Limit 1: 0
Limit 2: 50000
```

Slika 18. Podatci koje pošalje uređaj nakon naredbe V1234

Uređaj može poslati samo jedan podatak o udaljenosti, kao direktan odgovor na naredbu ***E***. No, češće se koristi kontinuirani izlaz podataka koji aktiviramo naredbom ***H1***. Kontinuirani izlaz podataka prekidamo naredbom ***H2*** kojom ujedno gasimo i laserski snop, a ukoliko želimo samo prekinuti izlaz podataka, a ostaviti laser snop upaljen onda koristimo naredbu ***H3***. Svaki izlazni podatak, podatak koji šalje senzor, se sastoji od 2 okteta (engl. byte), jednog tzv. niskog okteta (L) koji može poprimiti vrijednosti od 0 do 127 i jednog visokog (H) koji poprima vrijednosti između 128 i 255. Manji oktet (L) uvijek dolazi prije većeg (H). Iz ta dva okteta udaljenost izračunavamo na slijedeći način:

$$udaljenost = \frac{304.8[(H-128)128+L]}{16378} \quad (1)$$

Dobivena udaljenost je udaljenost od početka mjernog područja uređaja. Ukoliko želimo informaciju o udaljenosti od samog uređaja, onda dobivenoj vrijednosti moramo pridodati 228,6 mm, odnosno vrijednost udaljenosti najbliže točke mjernog područja od uređaja.

Ako vrijednost u uglatoj zagradi premaši broj 16378 znači da imamo neku grešku. Ako je ta vrijednost 16379 onda je mjerni objekt malo izvan mjernog područja, sa strane bliže mjernom uređaju. U slučaju vrijednosti 16381 mjerni objekt je također malo izvan radnog područja, ali sa strane udaljenije od mjernog uređaja. Iz vrijednosti 16380 zaključujemo da mjerni objekt nije u mjernom području niti u njegovoj neposrednoj blizini.

3.2. Dijelovi Arduino upravljačkog programa

```
#include <SPI.h>
#include <Ethernet.h>
#include <math.h>
```

Slika 19. Uvoz knjižnica

Na početku programa su pozvane tri knjižnice naredbi. Taj poziv se ostvaruje naredbom *#include*. *SPI.h* je knjižnica koja Arduinu omogućuje komunikaciju s SPI (engl. Serial Peripheral Interface) uređajima i treba ju pozvati ako koristimo mrežni štit jer je on SPI uređaj. SPI je vrsta brze serijske komunikacije koja se koristi za manje udaljenosti. Osim Ethernet štita, za povezivanje na mrežu nam treba i knjižnica *Ethernet.h* koja omogućuje da Arduino bude poslužitelj ili korisnik. Moguće su četiri istovremene konekcije: dolazne, odlazne ili kombinacije istih.

U *math.h* knjižnici su spremljene matematičke funkcije.

```
byte mac[] = {0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02};
IPAddress ip(192, 168, 123, 50);
IPAddress server(192, 168, 123, 25);
EthernetClient client;
```

Slika 20. Definiranje adresa i Arduina klijentom

mac[] je adresa Ethernet adaptera Arduino uređaja, nalazi se na naljepnici na Arduinu. U *ip* varijablu je spremljena IP adresa Arduina, a u *server* IP adresa robota. S funkcijom *EthernetClient* arduino je postavljen za korisnika, odnosno poslužitelj će biti upravljačka jedinica robota. Nakon toga slijedi deklaracija glavnih i pomoćnih varijabli te petlja *setup()*.

```
void setup() {
   Serial.begin(9600);

        for (int i = 0; i < 6; i = i + 1)
            Serial.write(P1[i]);
        for (int i = 0; i < 6; i = i + 1)
            Serial.write(L1[i]);
        for (int i = 0; i < 6; i = i + 1)
            Serial.write(H1[i]);

        Ethernet.begin(mac, ip);
        delay(1000);
        client.connect(server, 5555);
}
```

Slika 21. setup () petlja

Funkcija *Serial.begin()* pokreće serijsku komunikaciju. U zagradu se upisuje brzina prijenosa. Najčešće su: 4800, 9600, 14400, 19200. S obzirom da se *setup()* uvijek izvršava samo nakon što je Arduino pokrenut ili ponovno pokrenut (engl. restart) korisno je u uređaj poslati naredbe za postavljanje najčešće korištene kombinacije konfiguracija. Rad s mrežom započinje s pokretanjem naredbe *Ethernet.begin (mac, ip, dns, gateway, subnet);*. Mora se navesti jedino *mac* adresa, ostale se mogu dodijeliti automatski. Opis argumenata je:

- **mac** je adresa Ethernet adaptera Arduino uređaja
- **ip** je IP adresa Arduina. Ne mora se navesti ako koristimo uređaj koji koristi DHCP (engl. Dynamic Host Configuration Protocol) - mrežni protokol korišten od strane mrežnih računala za dodjeljivanje IP adresa i ostalih mrežnih postavki
- **gateway IP** je adresa mrežnog prolaza (engl. Gateway)
- **subnet** je maska mreže

S *client.connect()* spajamo Arduino na *ip adresu* poslužitelja i 'port' 5555.

```
void loop() {
  while (client.available()) {
    char newchar = client.read();
    if (newchar == 0x4b) {
      funkcija(commandString);
```

Slika 22. Početak loop() petlje

Uređaj koji upravlja senzorom Arduinu šalje poruku u ovome obliku: **P*naredba*K**. *'P'* označava početak poruke, *'naredba'* nam služi za mijenjanje postavki uređaja, dok *'K'* označava kraj poruke.

Definirane poruke su slijedeće:

"PsingleK" – zahtijevanje od uređaja slanje samo jednog podatka o udaljenosti

"PkonstK" – zahtijevanje od uređaja kontinuirano slanje podataka o udaljenosti

"PSonLonK" – zahtijevanje od uređaja kontinuirano slanje podataka o udaljenosti i uključivanje laserskog snopa

"PSoffLoffK" – zahtjev za prekidanje kontinuiranog slanja podataka i gašenja laserskog snopa

"PSoffLonK"– zahtjev za prekidanje kontinuiranog slanja podataka ali ne i laserskog snopa

"PbleOnK"– zahtjev za uključenje eliminacije pozadinskog osvjetljenja

"PbleOffK"– zahtjev za isključenje eliminacije pozadinskog osvjetljenja

"PrateK"– zahtjev za postavljanje broja izlaznih podataka uređaja kao prioritet

"PqualityK"– zahtjev za postavljanje kvalitete izlaznih podataka uređaja kao prioriteta

Uz pomoć naredbe *client.avaliable()* ispituje se da li ima podataka poslanih od poslužitelja prema korisniku. Sve dok podaci pristižu, znak po znak se čita i sprema u varijablu tipa String 'commandString'. Kada se upiše znak *'K'*, simbol za kraj poruke, String se šalje u funkciju *'funkcija'*. U toj funkciji se ispituje kakva je poruka stigla. Nakon što se otkrila funkcija pristigle naredbe, ta naredba se i izvršava. Ako je riječ o naredbi koja pokreće mjerenje udaljenosti, mjerni uređaj počinje Arduinu slati poruke s informacijom o udaljenosti ili, ako nešto nije u redu, poruku greške.

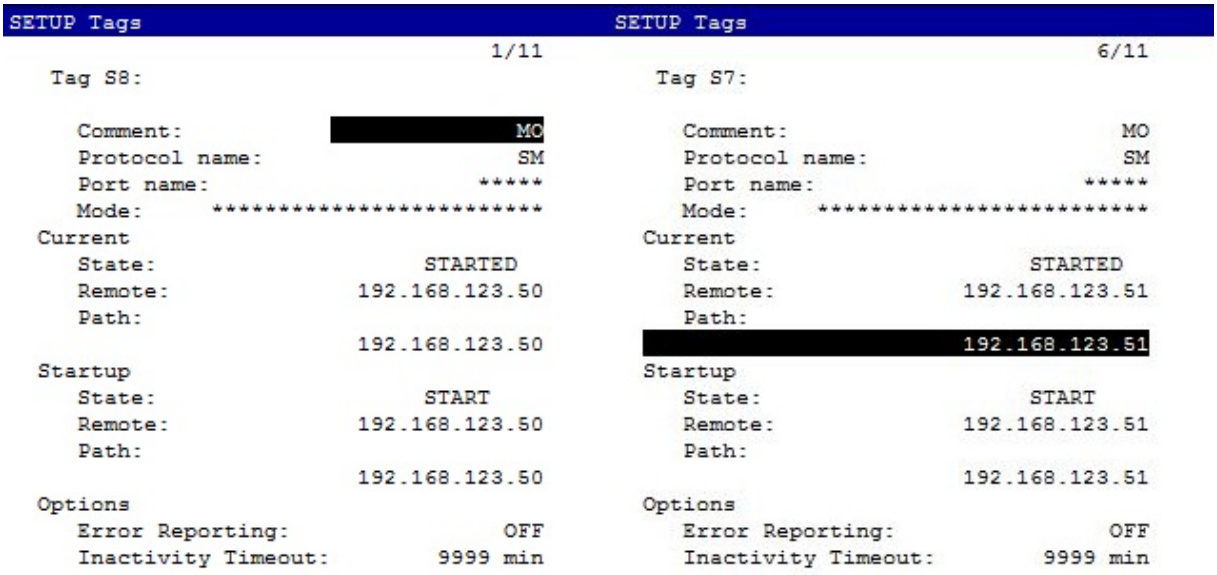
4. RAZVOJ KAREL PROGRAMSKOG KODA

Ključna riječ kojom počinje svaki KAREL program je 'PROGRAM + *ime_programa'*. Odmah nakon toga dolaze postavke kojima ćemo definirati da program radi u pozadini programa napisanog na robotovom privjesku za učenje → *%NOLOCKGROUP* i da nastavlja s odvijanjem bez obzira na javljanje nekih grešaka ili drugih situacija koje bi inače zaustavile odvijanje programa → *%NOPAUSE=ERROR + COMMAND + TPENABLE*.

Zatim je potrebno definirati sve varijable i konstantne različitih vrsta koje će se koristiti u programu, također u ovom dijelu programa, se pozivaju i vanjske 'Routine' (funkcije) koje će biti potrebne kasnije u programu. Konkretno u ovom slučaju to su dijelovi drugog Karel programa koji služi za izmjenu podataka *poslužitelj – korisnik* komunikacijom. Osim klasičnih varijabli (integer, float, string...), potrebno je podesiti i sistemske varijable (varijable predefinirane u upravljačkoj jedinici robota) korisničkog i alatnog koordinatnog sustava:

$GROUP[1].$UFRAME = $MNUFRAME[1,2]
$GROUP[1].$UTOOL = $MNUTOOL[1,10]

Od početnih postavki još je preostalo postaviti informacije o korisnik - poslužitelj komunikaciji (oznake, 'portove', IP adrese) [Slika 24]. Određeno je da poslužitelj bude robot, koji je konfiguriran direktno u postavkama kojima se može pristupiti pomoću privjeska za učenje [Slika 23], a Arduino i računalo će biti korisnici.



Slika 23. Postavki poslužitelja oznaka S7 i S8 na privjesku za učenje

```
025
026 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$OPER',0,STATUS) ;
027 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$STATE',0,STATUS) ; DELAY 20
028 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$COMMENT','MO',STATUS) ;
029 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$PROTOCOL','SM',STATUS) ;
030 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$REPERRS','FALSE',STATUS) ;
031 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$TIMEOUT',9999,STATUS) ;
032 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$PWRD_TIMEOUT',0,STATUS) ;
033 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$SERVER_PORT',5556,STATUS) ;
034 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$STRT_PATH','192.168.123.51',STATUS) ;
035 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$STRT_REMOTE','192.168.123.51',STATUS);
036 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$PATH','192.168.123.51',STATUS)
037 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$remote','192.168.123.51',STATUS)
038
039 DELAY 10 ; SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$OPER',3,STATUS);
040 DELAY 10 ;
041 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$STATE',3,STATUS) ;
042
043
044
045 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$OPER',0,STATUS) ;
046 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$STATE',0,STATUS) ; DELAY 20
047 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$COMMENT','MO',STATUS) ;
048 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$PROTOCOL','SM',STATUS) ;
049 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$REPERRS','FALSE',STATUS) ;
050 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$TIMEOUT',9999,STATUS) ;
051 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$PWRD_TIMEOUT',0,STATUS) ;
052 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$SERVER_PORT',5555,STATUS) ;
053 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$STRT_PATH','192.168.123.50',STATUS) ;
054 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$STRT_REMOTE','192.168.123.50',STATUS);
055 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$PATH','192.168.123.50',STATUS)
056 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$remote','192.168.123.50',STATUS)
057
058 DELAY 10 ; SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$OPER',3,STATUS);
059 DELAY 10 ;
060 SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$STATE',3,STATUS) ;
```

Slika 24. Isječak KAREL koda u kojem su definirane postavke korisnika za poslužitelj - korisnik komunikaciju

Konfiguriranjem svih potrebnih postavki, omogućen je početak razvoja glavnog dijela programa u kojem će se obrađivati i preusmjerivati podatci dobiveni s drugih dijelova sustava odnosno korisnika. Osim standardnih programskih funkcija poput *if()* i *for()* korištene su i neke karakterističnije za robotske sustave koje će biti opisane u nastavku. Prije samog ulaska u beskonačnu petlju potrebno je promijeniti početne vrijednosti pojedinih podatkovnih registara robota koji će se koristiti u programu. To se ostvaruje funkcijama:

SET_INT_REG(broj_registra, cjelobrojna_vrijednost_koju_postavljamo, statusna_varijabla)

SET_REAL_REG(broj_registra, realna_vrijednost_koju_postavljamo, statusna_varijabla)

Ulaskom u beskonačnu petlju upravljačka jedinica robota šalje, socket komunikacijom, zahtjev mjeraču udaljenosti za informaciju o udaljenosti objekta, slijedećom naredbom:

```
WRITE TCP('PsingleK',CR)
```

A odmah nakon toga potrebno je i pohraniti informacije o trenutnoj poziciji i orijentaciji vrha alata koje se spremaju u specijalnu pozicijsku varijablu:

p=CURPOS(0,0), gdje prva nula označava osi koje su izvan ograničenja, a druga označava amplitudu gibanja strojnog dijela izvan granica potrebnog da ostvari zadano. Za prosljeđivanje tih podataka računalu, potrebno ih je pretvoriti iz brojčanih vrijednosti u znakovne nizove slijedećem funkcijom, u kojoj je prvi parametar brojčana vrijednost koju pretvaramo, drugi, minimalna duljina novonastale znakovne varijable, treći, broj znakova iza decimalne točke, a četvrti parametar, znakovni niz u koji spremamo pretvorene podatke:

```
CNV_REAL_STR(p.x, 7, 3, pol_x)
CNV_REAL_STR(p.y, 7, 3, pol_y)
CNV_REAL_STR(p.z, 7, 3, pol_z)
CNV_REAL_STR(p.w, 7, 3, kut_w)
CNV_REAL_STR(p.p, 7, 3, kut_p)
CNV_REAL_STR(p.r, 7, 3, kut_r)
```

Mjerač udaljenosti odgovara zahtjevu upravljačke jedinice te joj šalje informaciju o udaljenosti u formatu niza znakova (engl. string) koje je potrebno dohvatiti i spremiti. Naredbom

READ TCP (poda::1) se čita znak po znak, te se jedan za drugim sprema u znakovnu varijablu 'assemble' sve do trenutka kada ne stigne znak ';' koji označava kraj poruke.

```
WHILE poda <> ';' DO
assemble = assemble + poda
READ TCP (poda::1)
ENDWHILE
```

Slijedeći korak je prosljeđivanje podataka računalu, također socket komunikacijom, jednom porukom odvojenih simbolima '@':

```
WRITE TCP1('P',pol_x,'@',pol_y,'@',pol_z,'@',kut_w,'@',kut_p,'@',kut_r,'@',assemble,'@',CR);
```

Svakim prolazom kroz WHILE petlju, odnosno jednog ciklusa prikupljanja i slanja podataka, provjerava se da li je na privjesku za učenje pritisnuta tipka F3 ili F4, a u slučaju da je tipka pritisnuta program izlazi iz beskonačne WHILE petlje i preskače u drugi dio programa određen u naredbi GOTO:

```
IF TPIN[132] THEN;GOTO van_;ENDIF -- F3
IF TPIN[133] THEN;GOTO nova_;ENDIF - F4
```

Pritiskom na tipku F3 program završava i izmjena podataka prestaje, a pritiskom na tipku F4 pokrećemo na računalu postupak usporedbe novonastalog oblaka točaka s drugim, prethodno određenim, oblakom točaka istog objekta, te nakon obavljene usporedbe upravljačka jedinica prihvaća koordinate točaka (X,Y,Z,W,P,R) u koju će pozicionirati vrh laserske zrake. Podatke odmah (nakon pretvorbe u brojčanu vrijednost) spremamo u registre tako da su odmah dostupni i na robotovom privjesku za učenje:

SET_REAL_REG(13,info,STATUS), gdje prvi parametar broj registra u koji spremamo podatak, a 'info' taj podatak koji u njega spremamo.

U slijedećem primjeru je pokazano dohvaćanje X i Y koordinate jedne točke, postupak za ostale podatke je identičan.

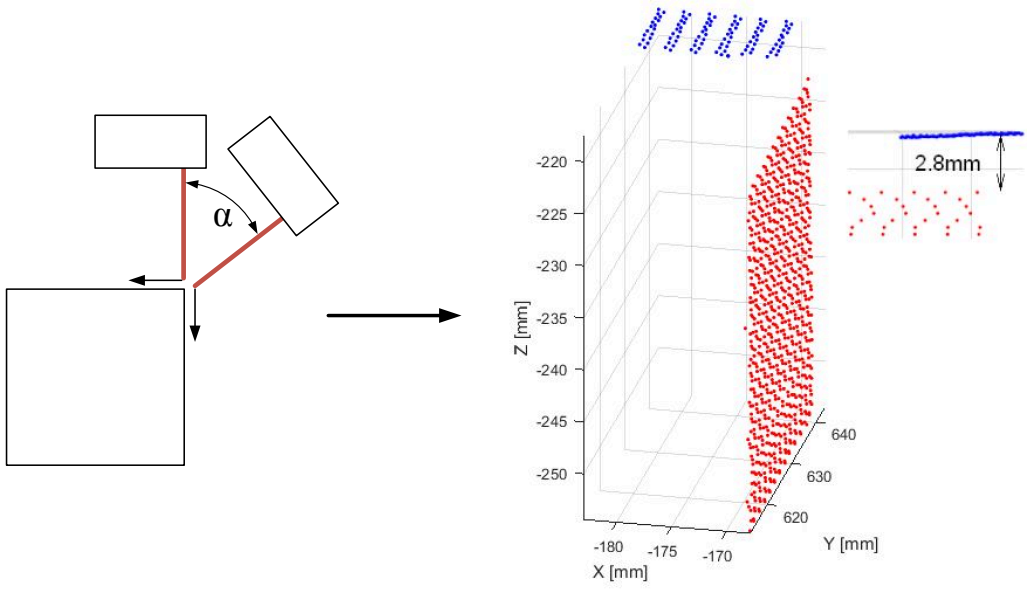
```
WHILE poda <> ';' DO
        assemble = assemble + poda
        READ TCP1 (poda::1) --X
        ENDWHILE
        CNV_STR_REAL(assemble,info)
        SET_REAL_REG(13,info,STATUS)
        READ TCP1 (poda::1)
        assemble = ''
WHILE poda <> ';' DO
        assemble = assemble + poda
        READ TCP1 (poda::1) --Y
        ENDWHILE
        CNV_STR_REAL(assemble,info)
        SET_REAL_REG(14,info,STATUS)
```

Dolaskom zadnjeg podatka s računala mjenjamo vrijednost još jednom registru, čijom promjenom upravljačka jedinica robota kreće s pozicioniranjem vrha laserske zrake u željene pozicije na objektu.

5. PROBLEMI U RAZVOJU SUSTAVA

5.1. Kalibracija

Prvi problem koji se javio nakon uspješne uspostave komunikacija između robota, arduina, laserskog mjerača udaljenosti i računala je nedovoljno dobra kalibracija vrha alata robota (engl. Tool Centre Point) čija preciznost je nužna za stvaranje točnog, realnog oblaka točaka objekta. U idealnoj situaciji vrh alata robota (TCP) bi trebao biti smješten negdje na laserskoj zraci tako da mu je Z os paralelna s njom. Prva kalibracija u radu je provedena standardnom robotovom metodom tri točke čija procedura zahtjeva pozicioniranje vrha alata u istu nepomičnu točku iz tri različita smjera (međusobno razmaknutih za barem 90°) nakon čega se izračunava koordinatni sustav alata. Vrh alata u ovom slučaju, kada se upotrebljava laserski mjerač udaljenosti, nije fizička točka već točka na laserskoj zraci, odnosno njen kraj, vrh, ali to nas ne sprječava u korištenju navedene metode jer robota možemo, zahvaljujući informaciji koju dobivamo iz mjerača udaljenosti, pozicionirati na poznatu udaljenost od nepomične točke namijenjene za kalibraciju, a nakon što smo pozicioniranje ponovili tri puta s tri različite pozicije TCP može biti izračunat. Rezultati dobiveni nakon takve kalibracije nisu dovoljno točni, grešku primjećujemo nakon zakretanja alata [Slika 25].



Slika 25. Pokusno testiranje

Slika 25 pokazuje sken gornje plohe kocke (plavo), dobiven vertikalnom orijentacijom alata i jedne bočne stranice kocke (crveno) dobivene skeniranjem iz pozicije koja je zakrenuta u

odnosu na vertikalu. Rubovi tih dviju ploha bi trebali biti otprilike preklopljeni, ali nisu zbog greške koja je posljedica kinematike robota i, još i više, koordinatnog sustava alata. Točke koje bi trebale imati iste koordinate se razlikuju otprilike za 2,8 mm. S ciljem unaprjeđenja kalibracije TCP-a korišten je algoritam razvijen u [18] gdje je korišten drugi robot pa je stoga bilo potrebno promijeniti Denavit – Hartenbergove (DH) parametre koji su prikazani u tablici 7.

Tablica 7. DH parametri FANUC LRMate 200iC/5L robota

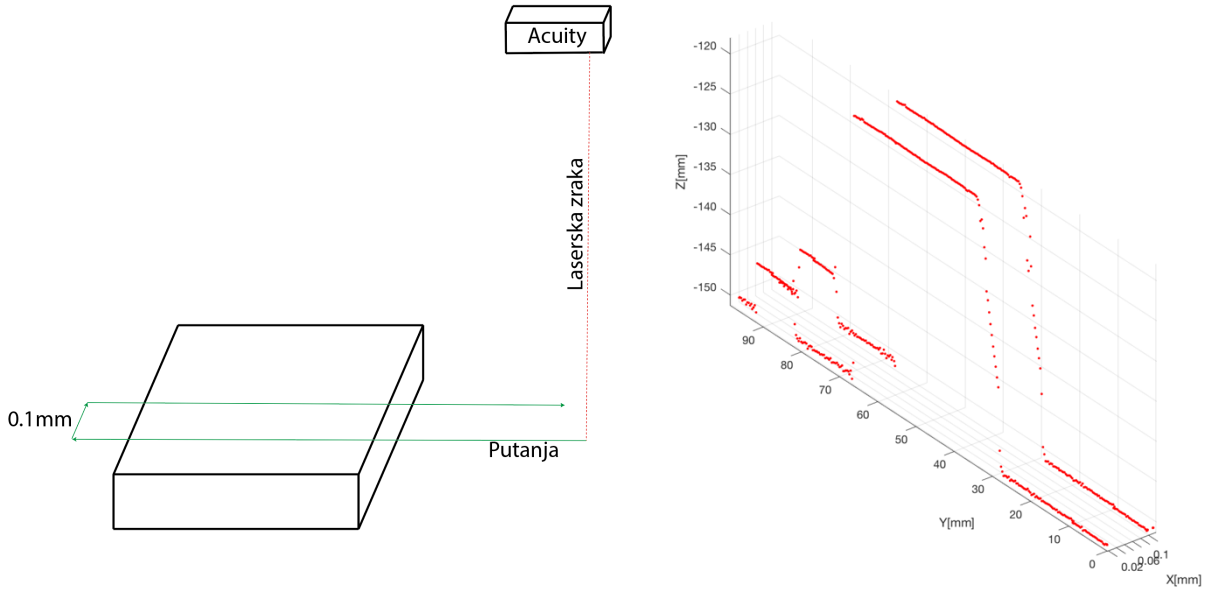
	d (mm)	θ (°)	a (mm)	α (°)
Članak 1	330	0	0	0
Članak 2	0	-90	75	-90
Članak 3	0	0	400	0
Članak 4	410	0	75	-90
Članak 5	0	0	0	90
Članak 6	80	0	0	-90

Nakon toga je TCP određen na sličan način samo s mnogo više pozicija oko fiksirane točke. Svakim dolaskom virtualnog TCP-a u fiksiranu točku (iz različitih smjerova i različitom orijentacijom) spremani su kutovi zakreti robotovih zglobova kako bi se mogli kasnije koristiti u algoritmu. Nakon što su podatci spremljeni iz dovoljnog broja dolazaka (u ovom slučaju 20) može se krenuti u izračunavanje. Algoritam iteracijama traži koordinatni sustav alata koji će imati najmanju grešku. S tim novim koordinatnim sustavom alata, greška se na istom testu [Slika 25] smanjila s 2,8 na 1,2 mm.

5.2. Sinkronizacija podataka između robota i laserskog mjerača udaljenosti

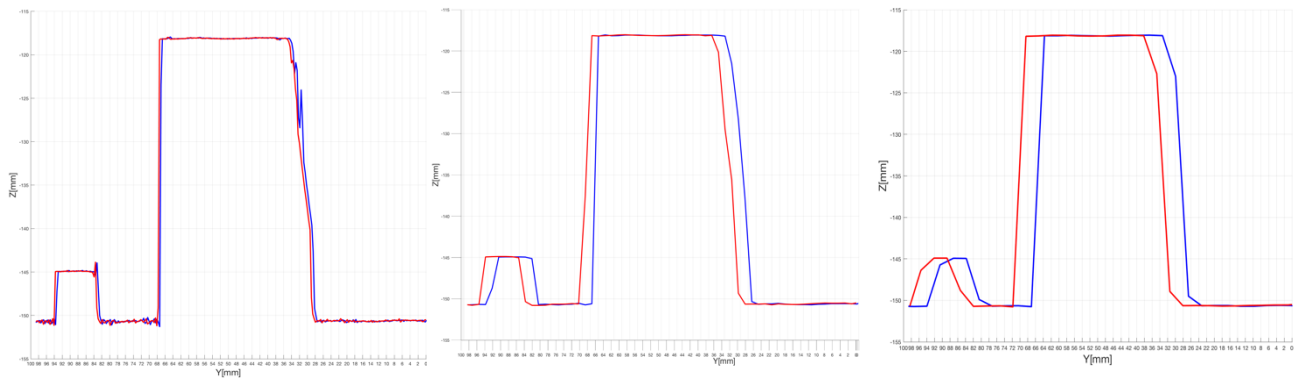
Nakon što je TCP određen nužno je provjeriti da li su podatci iz senzora (udaljenost) i podatci iz robota (pozicija vrha alata) sinkronizirani. Unutar komunikacijske petlje koja uključuje senzor, Arduino mikrokontroler i robota očekujemo kašnjenje u nekom dijelu koje će uzrokovati netočnost podataka. S ciljem utvrđivanja neusklađenosti podataka objekt je skeniran po jednoj liniji u jednom smjeru s jednom velikom promjenom visine na putanji. Nakon toga

smo senzor pomaknuli za malu udaljenost (0,1 mm) okomito na tu liniju, a zatim ponovno skenirali objekt u suprotnom smjeru paralelno s prvom linijom [Slika 26].



Slika 26. Točke dobivene za testiranje usklađenosti

Da su podatci usklađeni onda bi linije dobivene u testu trebale biti praktički preklopljene (u Z-Y ravnini), čak i u područjima s naglom promjenom visine. Osim kod ekstremno male brzine gibanja robota utvrđena su odstupanja među podatcima, i to, očekivano, sve veća što je i brzina gibanja robota veća.

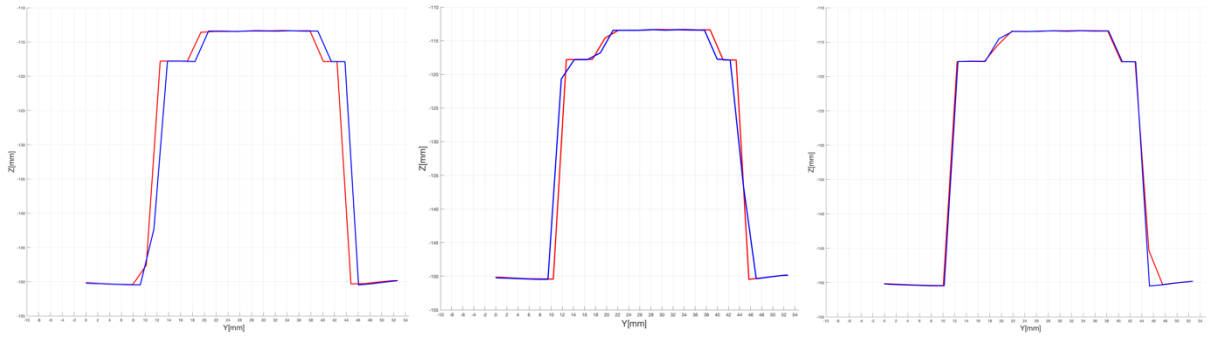


a) b) c)

Slika 27. Odstupanja pri 1mm/s, 5mm/s i 10mm/s

Za testiranje je odabrana brzina gibanja robota u iznosu od 7mm/s jer će ta brzina osigurati da proces skeniranja neće biti odviše dugotrajan, a ujedno će trajati dovoljno dugo da dohvati

dovoljan broj točaka (za objekte do otprilike 5 cm, za veće objekte bi trebalo odabrati veću brzinu iz razloga da proces ne bude dugotrajan). Za usklađivanje uređaja za odabranu brzinu potrebno je odrediti i postaviti vremensku odgodu (engl. delay) između slanja zahtjeva za udaljenošću mjeraču udaljenosti i zahtjeva za dohvaćanje pozicije i orijentacije alata robota. Eksperimentalno je određeno da ta vremenska odgoda mora iznositi 85 ms [Slika 28].



a) b) c)

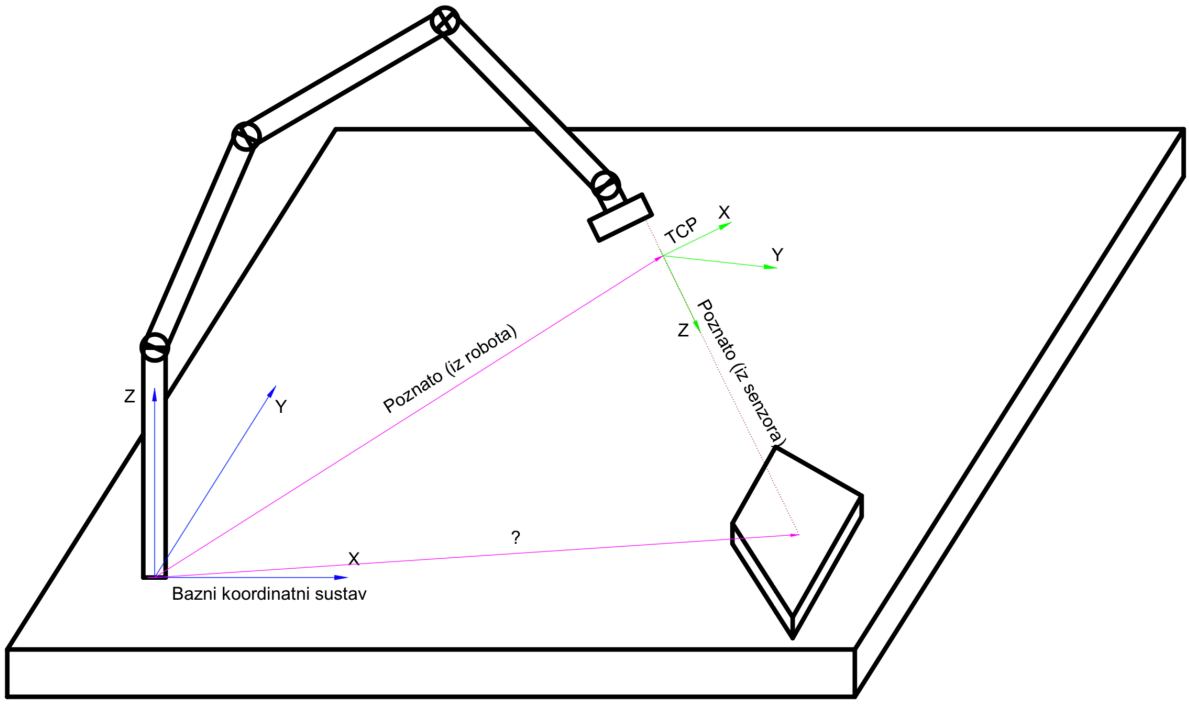
Slika 28. a) 7mm/s bez odgode; 7mm/s s odgodom od 90ms; 7mm/s s odgodom od 85ms

6. OBRADA PODATAKA

MATLAB, programski jezik visoke razine, koji se uz mnogobrojne dodatke i proširenja koristi za programiranje, obradu signala i fotografije, pregledan prikaz rezultata, statistiku, regulaciju, razne simulacije i još mnogo toga, je u ovom radu korišten za:

- povezivanje s upravljačkom jedinicom robota
- izmjenu podataka
- obradu podataka
- prikaz rezultata
- prilagodbu rezultata za moguće korištenje u drugim programima

Kao što je već ranije rečeno, računalo je u ovom sustavu komunikacije korisnik, koji se povezuje na poslužitelja – upravljačku jedinicu robota. Postavljanje parametara za komunikaciju, njena uspostava i prihvaćanje podataka se ostvaruje na početku glavnog Matlab koda pozivom vanjskih funkcija koje su razvijene za tu svrhu u nekim prijašnjim radovima. Od upravljačke jedinice robota dobivamo pakete koji sadrže 7 informacija, 3 prostorne koordinate vrha alata robota, 3 podatka o njegovoj orijentaciji i podatak o udaljenosti od objekta u tom trenutku. Prostorne koordinate vrha alata su koordinate u baznom koordinatnom sustavu robota, također, podatci o orijentaciji vrha alata su u odnosu na taj isti bazni koordinatni sustav. Iz tih podataka je potrebno izračunati položaj točke na objektu, odnosno koordinate vrha laserske zrake, naravno i one će biti u baznom koordinatnom sustavu.



Slika 29. Geometrijski odnosi dijelova sustava

Podatci o orijentaciji vrha alata (odnosno njegovog koordinatnog sustava čije ishodište je upravo vrh alata) dolaze u obliku tzv. Eulerovih kutova od kojih svaki određuje zakrenutost oko pojedine osi. Može se reći da su Eulerovi kutovi zapravo 3 uzastopne rotacije oko osi koordinatnog sustava čiji je redoslijed jako bitan jer različiti redoslijedi rezultiraju različitom dobivenom konačnom orijentacijom. Od 12 različitih, u praksi korištenih, Fanuc roboti koriste X(ϕ)-Y(φ)-Z(ψ) redoslijed rotacija. Za daljnje računanje potrebno je transformirati Eulerove kutove u matricu rotacije koja se dobije množenjem triju matrica rotacije (važećim redoslijedom) od kojih svaka predstavlja zakret oko jedne osi [13]:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2}$$

$$R_y(\varphi) = \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{bmatrix} \tag{3}$$

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$R(\phi,\varphi,\psi) = R_z(\psi)R_y(\varphi)R_x(\phi) = \\ = \begin{bmatrix} \cos\varphi\cos\phi & \sin\psi\sin\varphi\cos\phi - \cos\psi\sin\phi & \cos\psi\sin\varphi\cos\phi + \sin\psi\sin\phi \\ \cos\varphi\sin\phi & \sin\psi\sin\varphi\sin\phi + \cos\psi\cos\phi & \cos\psi\sin\varphi\sin\phi + \sin\psi\cos\phi \\ \sin\varphi & \sin\psi\cos\varphi & \cos\psi\cos\varphi \end{bmatrix} \quad (5)$$

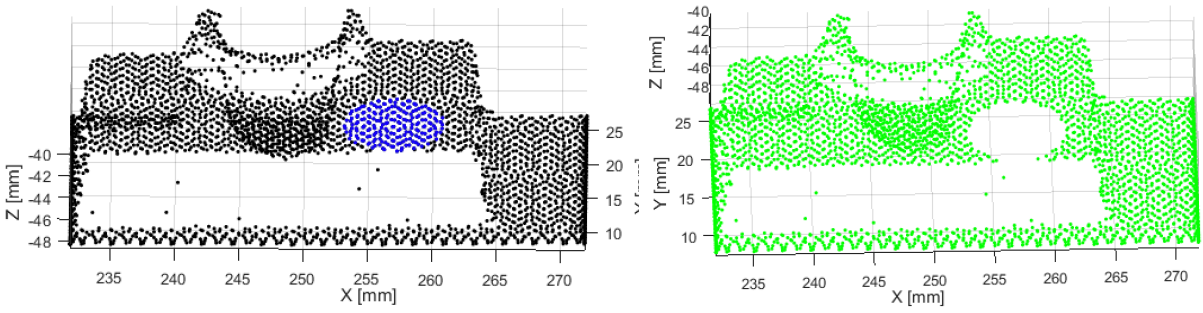
S obzirom da je vrha alata kalibriran (poglavlje 5.1) tako da je os Z koordinatnog sustava alata upravo laserska zraka, za izračun koordinata točaka na objektu je dovoljno proširenu matricu rotacije pomnožiti s vektorom stupcem koji sadrži vrijednost iz mjerača udaljenosti:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\varphi\cos\phi & \sin\psi\sin\varphi\cos\phi - \cos\psi\sin\phi & \cos\psi\sin\varphi\cos\phi + \sin\psi\sin\phi & 0 \\ \cos\varphi\sin\phi & \sin\psi\sin\varphi\sin\phi + \cos\psi\cos\phi & \cos\psi\sin\varphi\sin\phi + \sin\psi\cos\phi & 0 \\ \sin\varphi & \sin\psi\cos\varphi & \cos\psi\cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ l-10 \\ 1 \end{bmatrix} \quad (6)$$

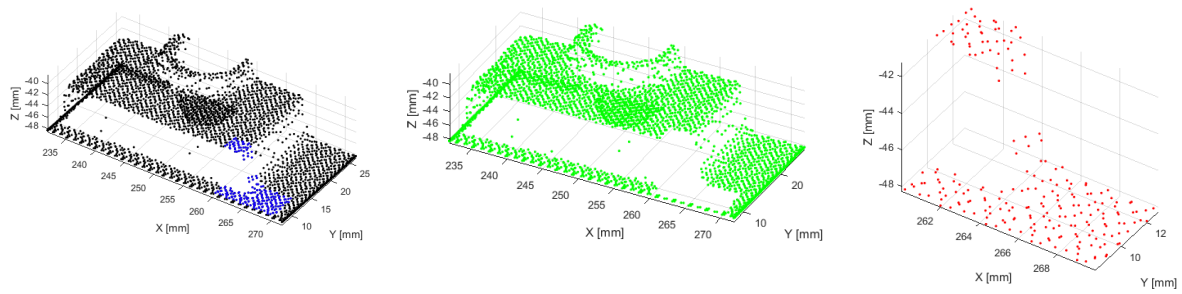
Gdje je *l* udaljenost koju daje laserski mjerač udaljenosti, koja mora biti umanjena za 10 mm jer vrh alata nije postavljen na sam početak mjernog područja mjerača udaljenosti, već je odmaknut za 10 mm. X,Y i Z su koordinate točke na objektu (vrha laserske zrake). Skupljanjem više točaka s istog objekta dobivamo njegov oblak točaka kojeg potom obrađujemo.

6.1. Analiza prikupljenih podataka

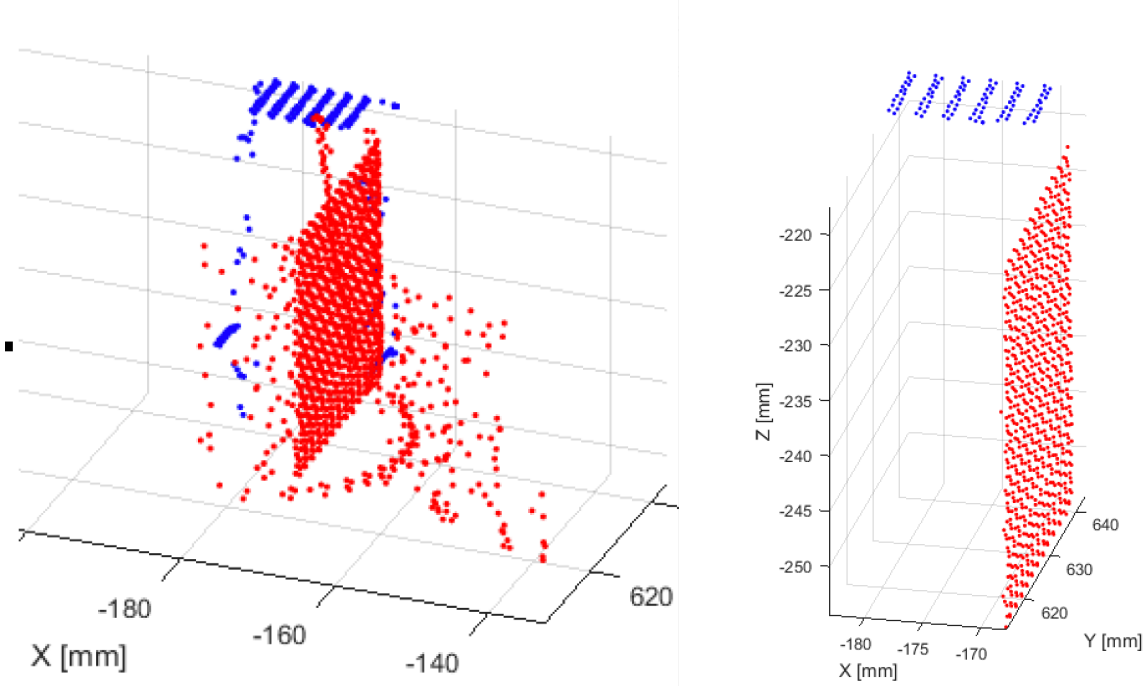
Nakon skeniranja objekta uvijek postoje i neke neželjene točke koje su rezultat greške ili najčešće rezultat skeniranja okoline ciljanog objekta. Obično je prvi korak u procesu njihovo uklanjanje. Oblak točaka u .ply formatu je strukturiran tako da je svakoj točci oblaka pridružena brojčana oznaka pomoću kojih se mogu pronaći točke unutar definirane sfere ili uspravnog kvadra, selektirati i jednostavno izuzeti iz oblaka, tj. stvori se novi oblak iz postojećeg, ali bez tih točaka ili samo s tim točkama.



Slika 30. Uklanjanje točaka definiranjem sfere

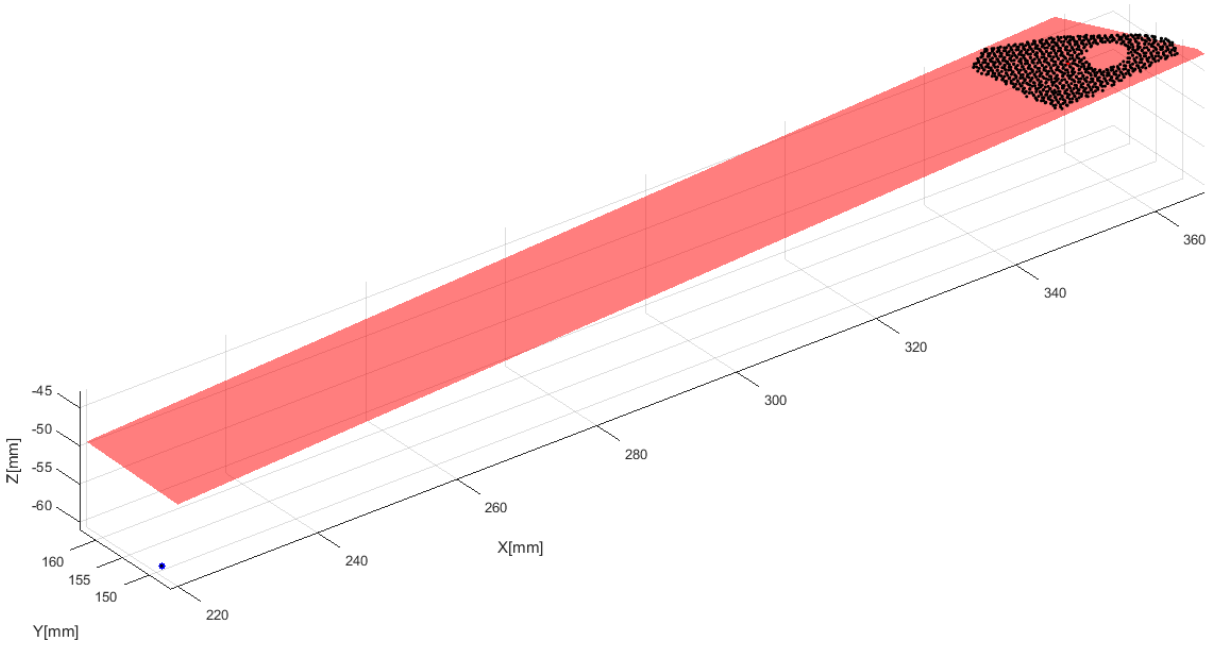


Slika 31. Uklanjanje točaka definiranjem uspravnog kvadra: a) izvorni oblak, b) oblak bez označenih točaka, c) oblak koji sadrži samo izbačene točke

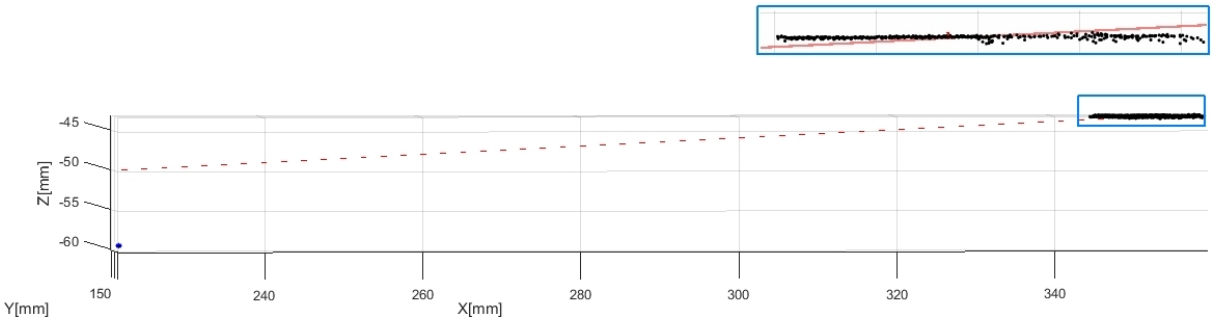


Slika 32. Oblaci točaka prije i poslije uklanjanja nepotrebnih točaka

Posjedovanjem velikog broja prikupljenih točaka često dolazi do potrebe pronalaska matematičkog modela plohe koja ih najbolje opisuje. Ploha, u ovom slučaju ravnina, se određuje tako da je zbroj kvadrata euklidskih udaljenosti točaka od nje, minimalan. Nedostatak, kod takvog pristupa pronalaska, je velik utjecaj točaka koje znatno odstupaju od većine ostalih te time jako smanjuju kvalitetu aproksimacije kompletnog skupa točaka. Takvu, lošu, aproksimaciju zbog samo jedne 'krive' točke (od njih 964) možemo vidjeti na slikama 33 i 34 .



Slika 33. Prikaz loše aproksimacije oblaka točaka



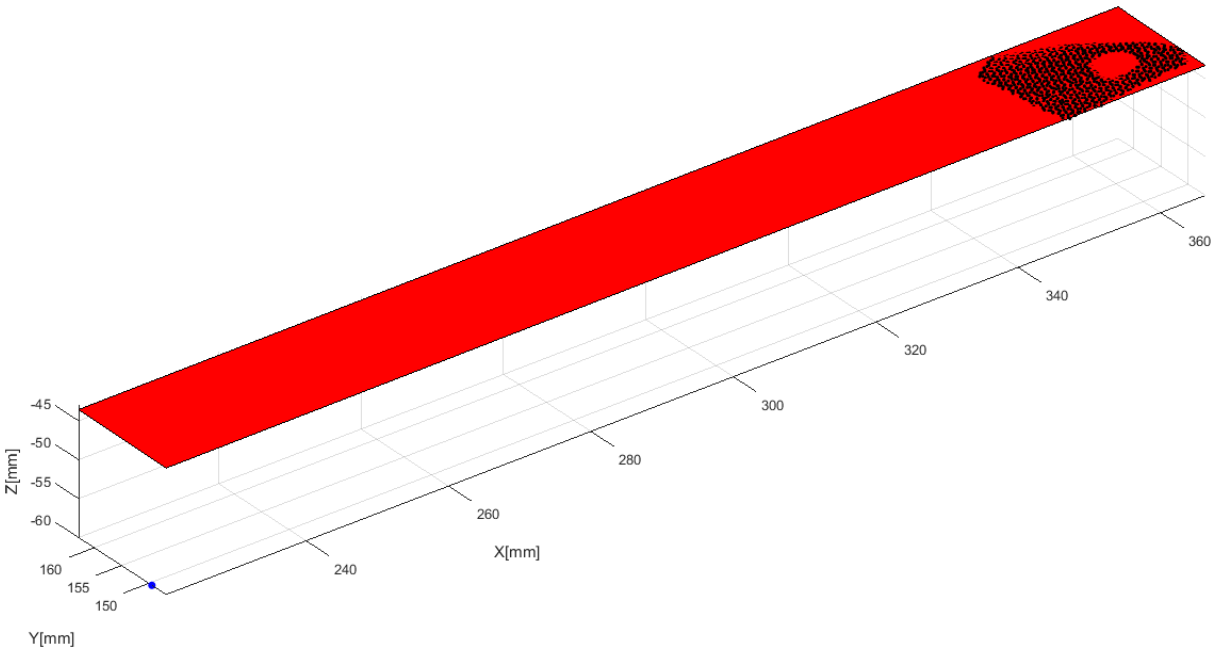
Slika 34. Prikaz loše aproksimacije oblaka točaka (s boka)

Jedan od načina kako suzbiti utjecaj udaljenih točaka u postupku pronalaska ravnine je upotreba metoda estimacije konsenzusom slučajnih uzoraka - RANSAC algoritmom (engl. RANdom Sample Consensus) koja zanemaruje izrazito udaljene podatke te model procjenjuje samo na temelju točaka koji su u skladu s pretpostavljenim modelom. Koraci njegovog izvođenja su slijedeći:

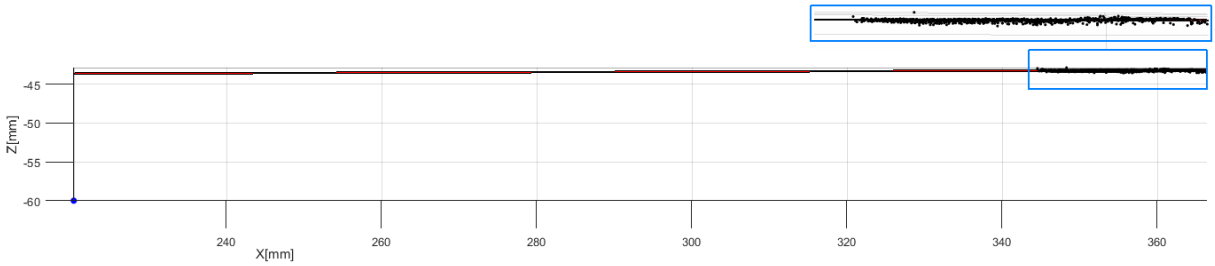
1. Nasumično odabiranje triju točaka iz skupa i određivanje ravnine koja prolazi kroz njih.
2. Za svaku točku oblaka provjeriti da li se njena udaljenost od te ravnine nalazi u granicama dozvoljenog odstupanja (ulazna vrijednost koju određuje korisnik) i prebrojiti koliko ih je.

3. Ponavljati dok se ne pronađe ravnina kod koje je dovaljan broj točaka (većina) unutar zadanog odstupanja.
4. Za skup podataka pronađen u prethodnom koraku (točke unutar dozvoljenog odstupanja) provesti postupak prilagođavanja ravnine metodom najmanjih kvadrata.

Rezultati dobiveni s tom metodom su mnogo bolji, a mogu se vidjeti na slikama 35 i 36.



Slika 35. Prikaz dobre aproksimacije oblaka točaka



Slika 36. Prikaz dobre aproksimacije oblaka točaka (s boka)

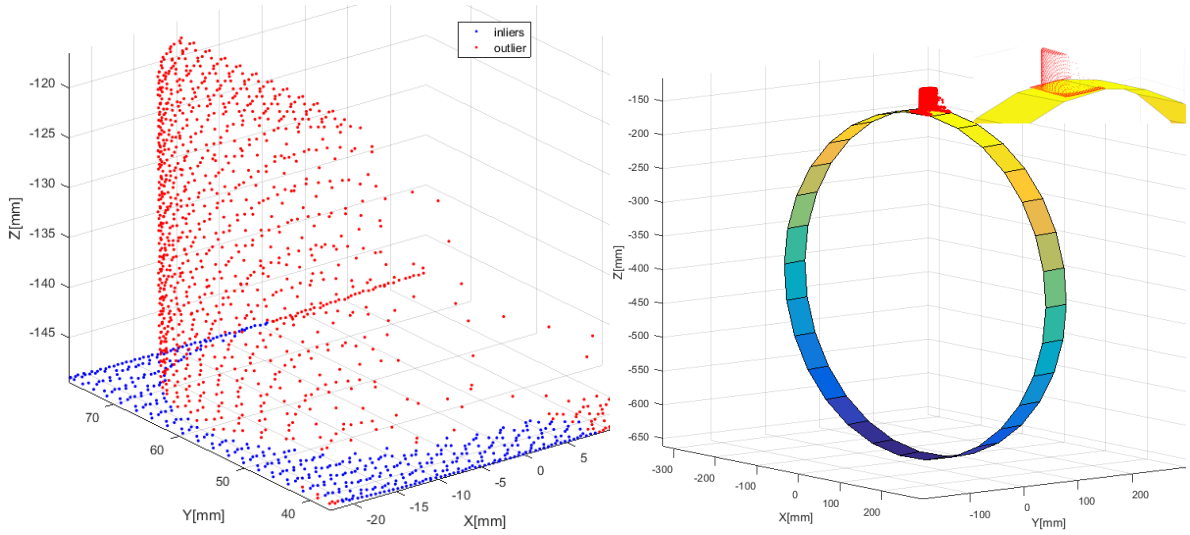
Upotrebom RANSAC algoritma ne može se predvidjeti koje će se 3 točke iskoristit za izradu pokusne ravnine, pa stoga dobivamo različite rezultate iz uzastopnih računanja s istim ulaznim vrijednostima (oblak točaka i dopušteno odstupanje te broj pokušaja). U ovom slučaju su razlike zanemarive, provjera ponovljivosti s istim ulaznim podacima je dana u tablici 8. (Ekstremna točka (ionako naknadno, ručno dodana u oblak točaka) s koordinatama 220,150,-60 je izuzeta

iz daljnjih testova, čisto radi bolje preglednosti grafova i utjecaja na prosječnu vrijednost svih točaka od ravnine).

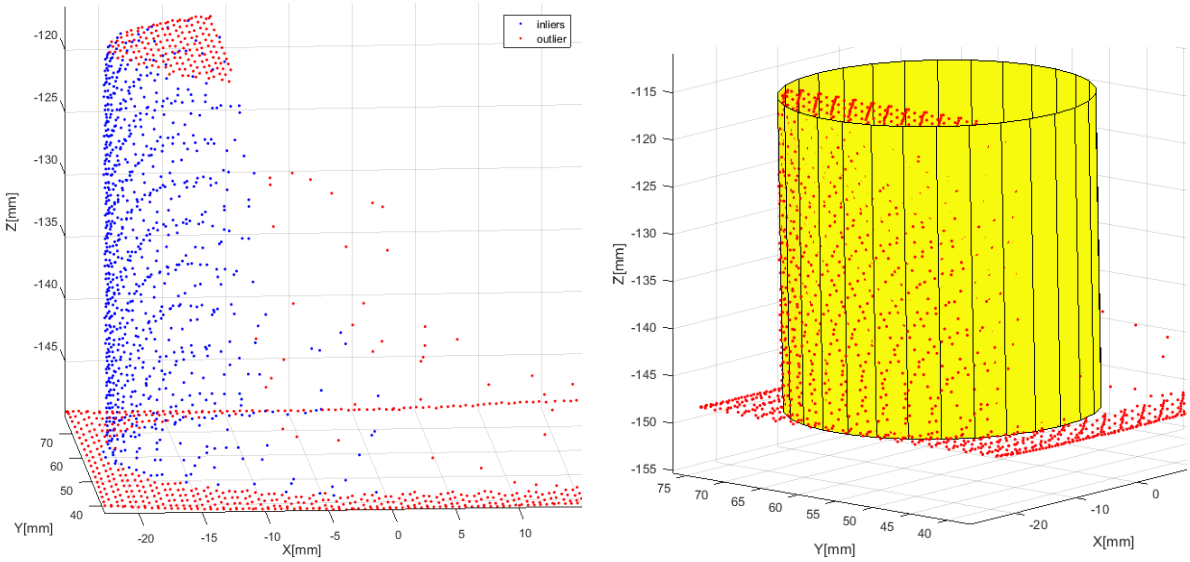
Tablica 8. Ponovljivost RANSAC metode kod aproksimacije ravnine

	Parametri ravnine iz opće jednadžbe: Ax+By+Cz+D=0	Broj populacijskih točaka (engl. inliers)	Prosječna udaljenost populacijskih točaka od ravnine [mm]	Prosječna udaljenost svih točaka od ravnine [mm]
1	-0,0073; -0,0237; 0,9997; 49,4128	637 (66%)	0,0265	0,0939
2	0,0088; 0,0185; -0,9998; -49,2058	684 (71%)	0,0301	0,0801
3	0,0055; 0,0178; -0,9998; -47,9186	671 (70%)	0,0256	0,0821
4	0,0072; 0,0143; -0,9999; -47,9874	698 (72%)	0,0268	0,0772
5	0,0045; 0,0179; -0,9998; -47,5718	674 (70%)	0,026	0,0808
6	0,0088; 0,0071; -0,9999; -47,4851	633 (66%)	0,0329	0,0785
7	0,0014; 0,0105; -0,9999; -45,3457	645 (67%)	0,0315	0,0819
8	0,0078; 0,0178; -0,9998; -48,7132	673 (70%)	0,0259	0,0828
9	-0,0074; -0,0219; 0,9997; 49,2408	666 (69%)	0,0288	0,0843
10	-0,0036; -0,0112; 0,9999; 46,2704	695 (72%)	0,0307	0,0740

Na slici 37 se može vidjeti primjer situacije gdje odabir prvih triju točaka itekako nije zanemariv jer je oblak točaka nastao skeniranjem više ravnina. U oba pokretanja algoritma su korišteni isti ulazni podatci, a rezultati su drastično različiti. U prvoj situaciji [Slika 37] su 3 točke odabrane na donjoj plohi, a u drugoj [Slika 38], na bočnoj željnoj plohi dijela plašta valjka.



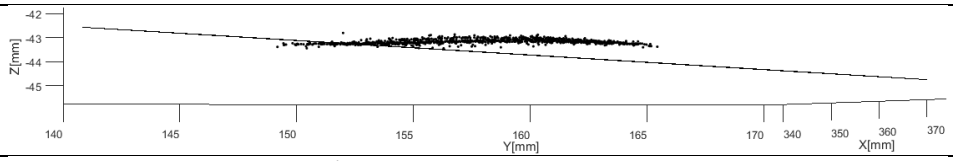
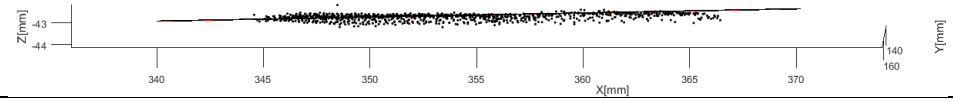
Slika 37. Utjecaj lošeg odabira prvih triju točaka

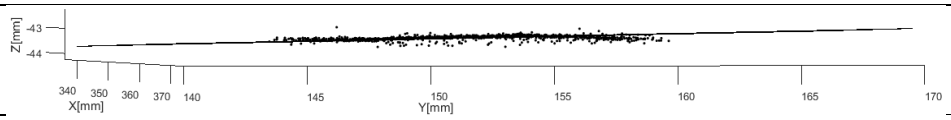
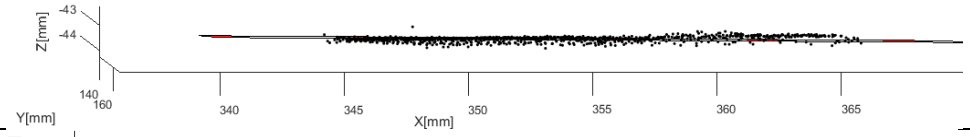
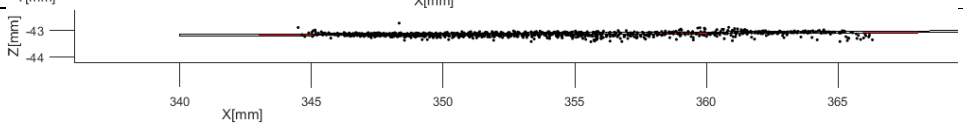


Slika 38. Dobar odabir prvih triju točaka

U tablici 9 je prikazan utjecaj dopuštenog odstupanja na dobivenu ravninu. Predviđanje da će ravnina to bolje opisivati oblak točaka što je dozvoljeno odstupanje manje, se potvrdilo.

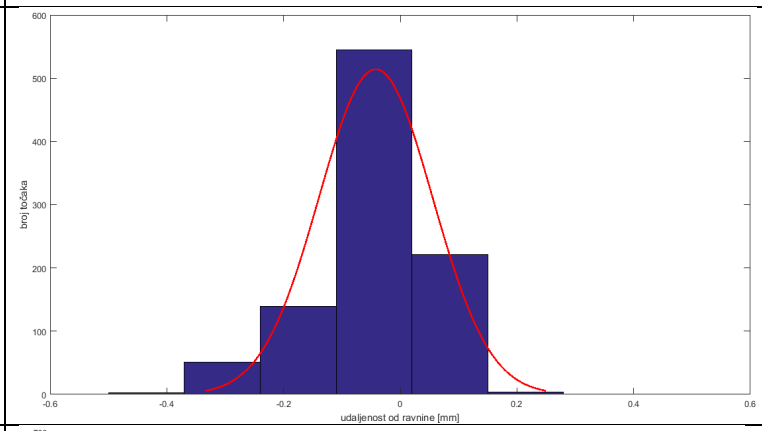
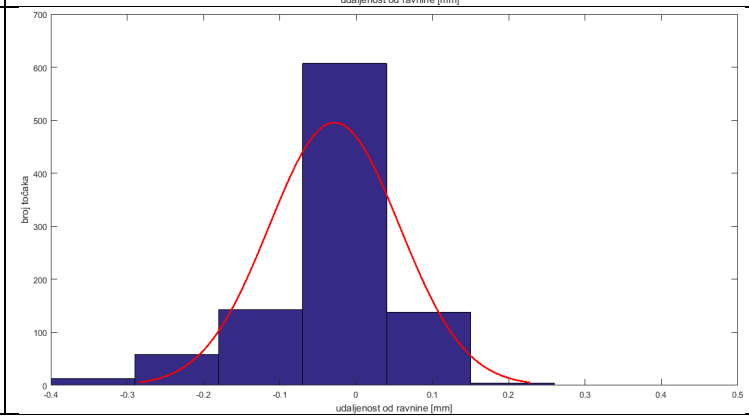
Tablica 9. Utjecaj dopuštenog odstupanja na dobivenu ravninu

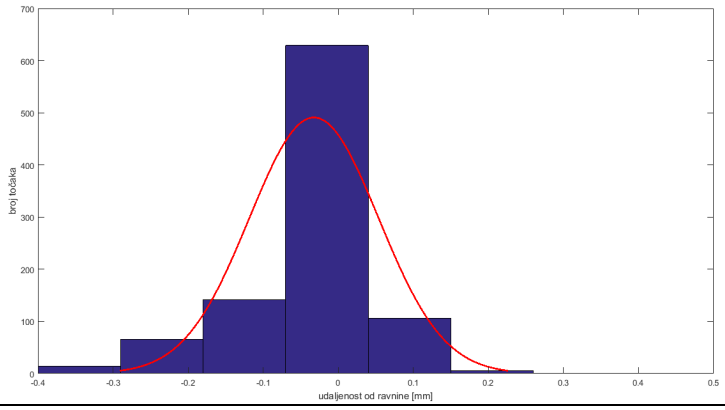
Dozvoljeno odstupanje	Dobivena ravnina (i oblak točaka za koji tražimo ravninu)
2 mm	
1 mm	

0,5 mm	
0,25 mm	
0,07 mm	

S tri uzastopna skeniranja ravne plohe pleksiglasa te stvaranjem ravnine RANSAC algoritmom s dozvoljenim odstupanjem od nje u iznosu od 0,07 mm su dobiveni slijedeći rezultati:

Tablica 10. Podatci o rezultatima skeniranja ravne plohe pleksiglasa

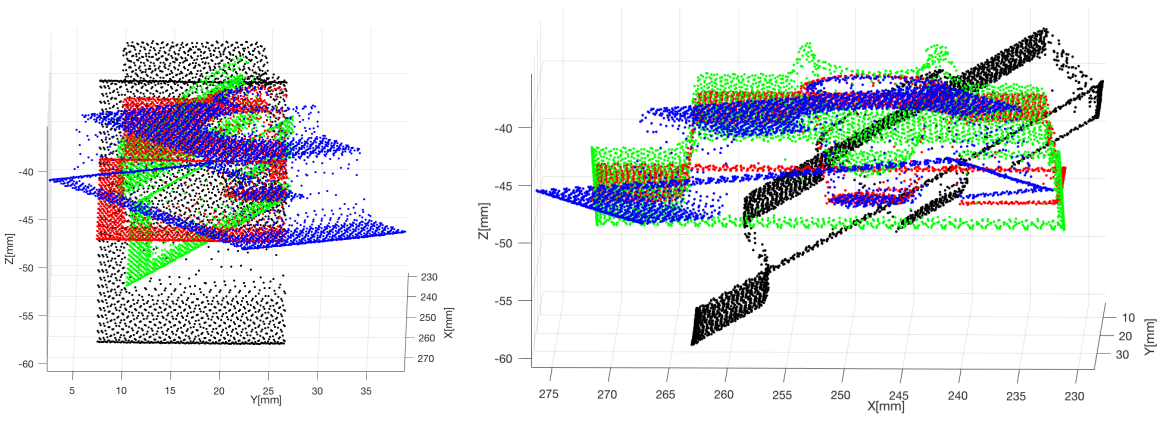
Broj populacijskih točaka	Najveća udaljenost točke od ravnine	Prosječna vrijednost udaljenosti točaka od ravnine	Prikaz distribucije udaljenosti točaka od ravnine
667	0,4958	0,0696	
710	0,4735	0,0601	

699	0,4643	0,061	

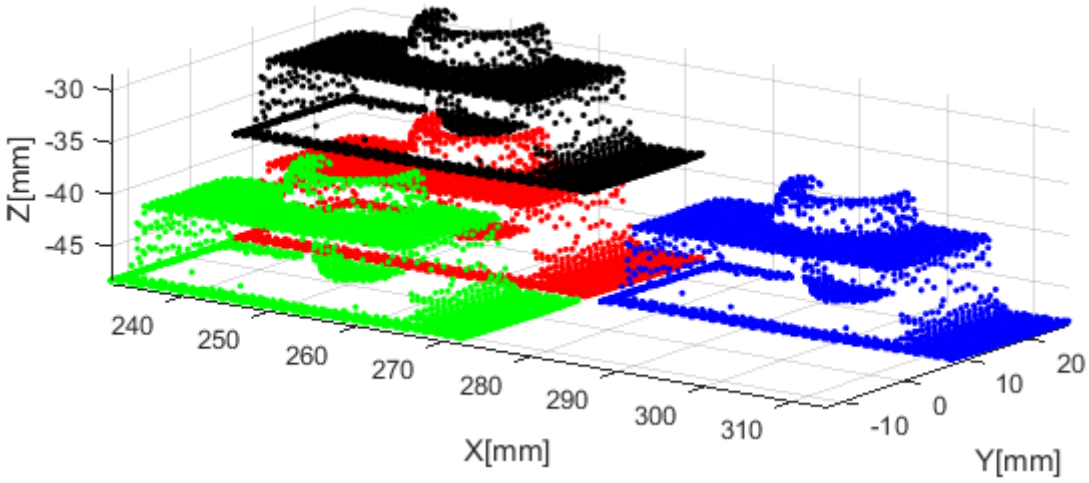
6.2. Preklapanje oblaka točaka

Za preklapanje oblaka točaka (engl. scan matching) potrebno je oblake točaka pomicati i rotirati. To je također moguće ostvariti u Matlabu, definiranjem matrica homogene transformacije. To je matrica 4x4 koja se sastoji od matrice rotacije **R** i vektora pomaka **t** i tako definira rotaciju i pomak, prikazana je slijedećom jednadžbom:

$$\boldsymbol{T} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & 0 \\ R_{21} & R_{22} & R_{23} & 0 \\ R_{31} & R_{32} & R_{33} & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{0} \\ \boldsymbol{t} & 1 \end{bmatrix} \tag{7}$$



Slika 39. Rotacija oblaka točaka



Slika 40. Translacija oblaka točaka

Dva najčešća algoritma za preklapanje oblaka točaka su ICP (engl. Iterative Closest Point) i NDT (engl. Normal distributions transform). NDT algoritam je brži, točniji i prikladniji za rjeđe oblake točaka, ali je nepouzdaniji u slučajevima kada su početni položaji oblaka točaka značajno odmaknuti (a u ovoj primjeni je to vrlo čest slučaj) pa je stoga za korištenje u ovom radu odabran ICP algoritam.

ICP algoritam za preklapanje dvaju oblaka minimizira udaljenost između svake točke jednog oblaka i njenog para u drugom oblaku. Jedan oblak točaka se definira kao nepomičan dok se drugi pomiče i rotira kako bi najbolje 'preklopio' nepomičnog. Algoritam iterativno mijenja transformaciju (kombinaciju translaciju i rotacije) kako bi se smanjila greška, odnosno suma najmanjih kvadrata udaljenosti između koordinata točaka dvaju oblaka. Razlika, funkcija greške se zapisuje u obliku:

$$rmse(\boldsymbol{R},\boldsymbol{t}) = \sum \left|\boldsymbol{R}\boldsymbol{p}_{\mathrm{i}} + \boldsymbol{t} - \boldsymbol{q}_{\mathrm{j}}\right|^2 \tag{8}$$

Gdje je:

- $\boldsymbol{R}$ - matrica rotacije
- $\boldsymbol{t}$ - vektor translacije
- $\boldsymbol{p}_{\mathrm{i}}$ - prvi oblak točaka
- $\boldsymbol{q}_{\mathrm{j}}$ - drugi oblak točaka

Generalno, koraci algoritma su slijedeći:

1. Za svaku točku u pomičnom oblaku točaka pronaći najbližu točku u nepomičnom
2. Metodom najmanjih kvadrata odrediti kombinaciju rotacija i translacija koje će najbolje poravnati svaku točku pomičnog oblaka s njegovim parom u nepomičnom oblaku točaka i odbaciti najlošije parove
3. Transformirati pomični oblak točaka koristeći dobivenu transformaciju
4. Iterativno ponavljati (ponovno pridruživati točke, sve dok se ne stekne uvjet za prestanak)

Ako oblaci točaka ne sadrže jednak broj točaka funkcija greške se modificira i sadrži vektor težinskih koeficijenata. Izlazni podatci funkcije su: transformirani oblak točaka, matrica transformacije i greška preklapanja. Ulazni podatci za funkciju su: dva oblaka točaka (mirujući i pomični), uvjet za prestanak iteracije, metoda poravnanja, početna transformacija pomičnog oblaka (opcionalno) i maksimalan broja iteracija (opcionalno). Funkcija omogućava izbor između dvije metode poravnanja: 'točka na točku' (engl. point to point) i 'točka na tangentnu plohu' (engl. point to plane), čije usporedbe možemo vidjeti u tablici 11.

Tablica 11. Usporedba metoda poravnanja [14]

'točka na točku'	'točka na tangentnu plohu'
	
- koristi metodu najmanjih kvadrata za pronalaženje transformacije koja poravnava točke s njihovim odgovarajućim parovima	- inkrementalno poboljšava poravnavanje pronalazeći male rotacije i translacije koje pomiču točke bliže tangentnim plohama (engl. tangent planes) njihovih odgovarajućih parova
- točke jedne površine su pridružene odgovarajućim točkama druge površine	- točke su pridružene odgovarajućim tangentama

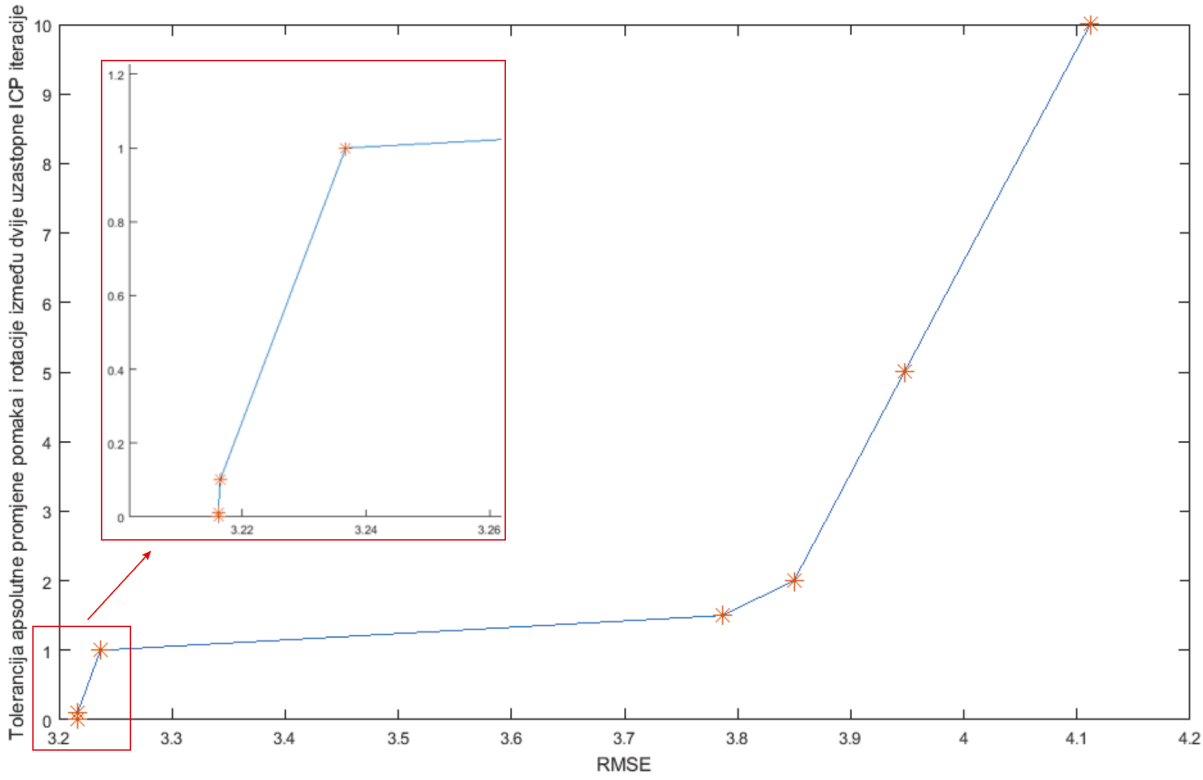
- može dugo trajati	- već nakon nekoliko iteracija pronalazi rješenje
- može divergirati	- najčešće konvergira

Korištenje metode 'točka na tangentnu plohu' smo odmah na početku izbacili iz mogućnosti za upotrebu jer zahtjeva previše dobro pogođenu početnu transformaciju za iteraciju što nije optimalno za primjenu u ovom radu jer položaj objekta ne mora biti poznat [Slika 41].

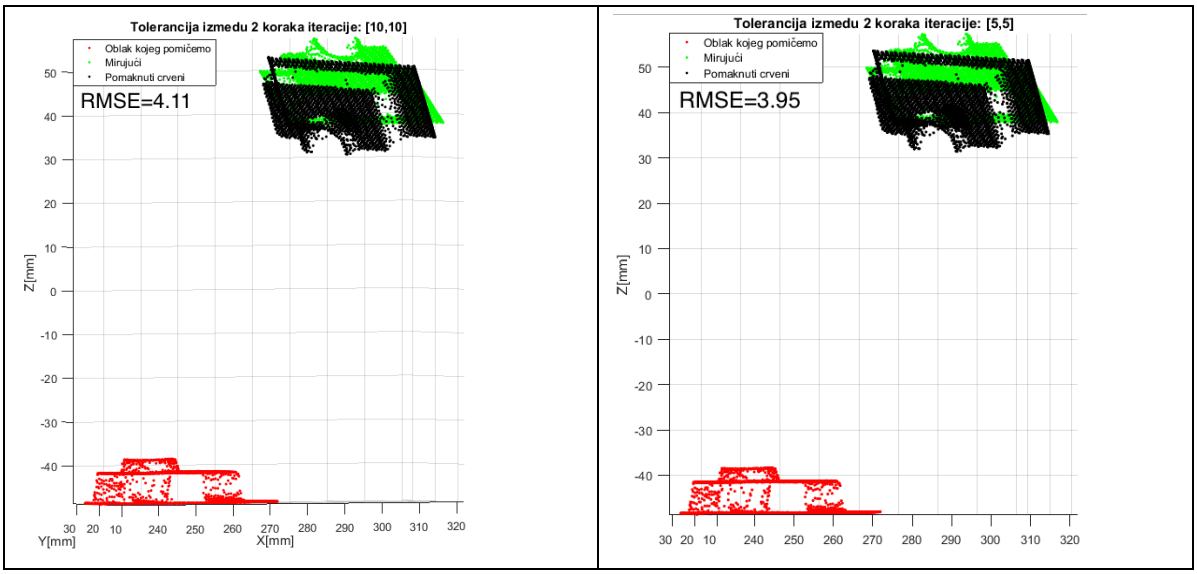
```
Error using pcregrigid (line 259)
Unable to register the point clouds. Consider providing a better initial transformation using 'InitialTransform' name-value pair.
```

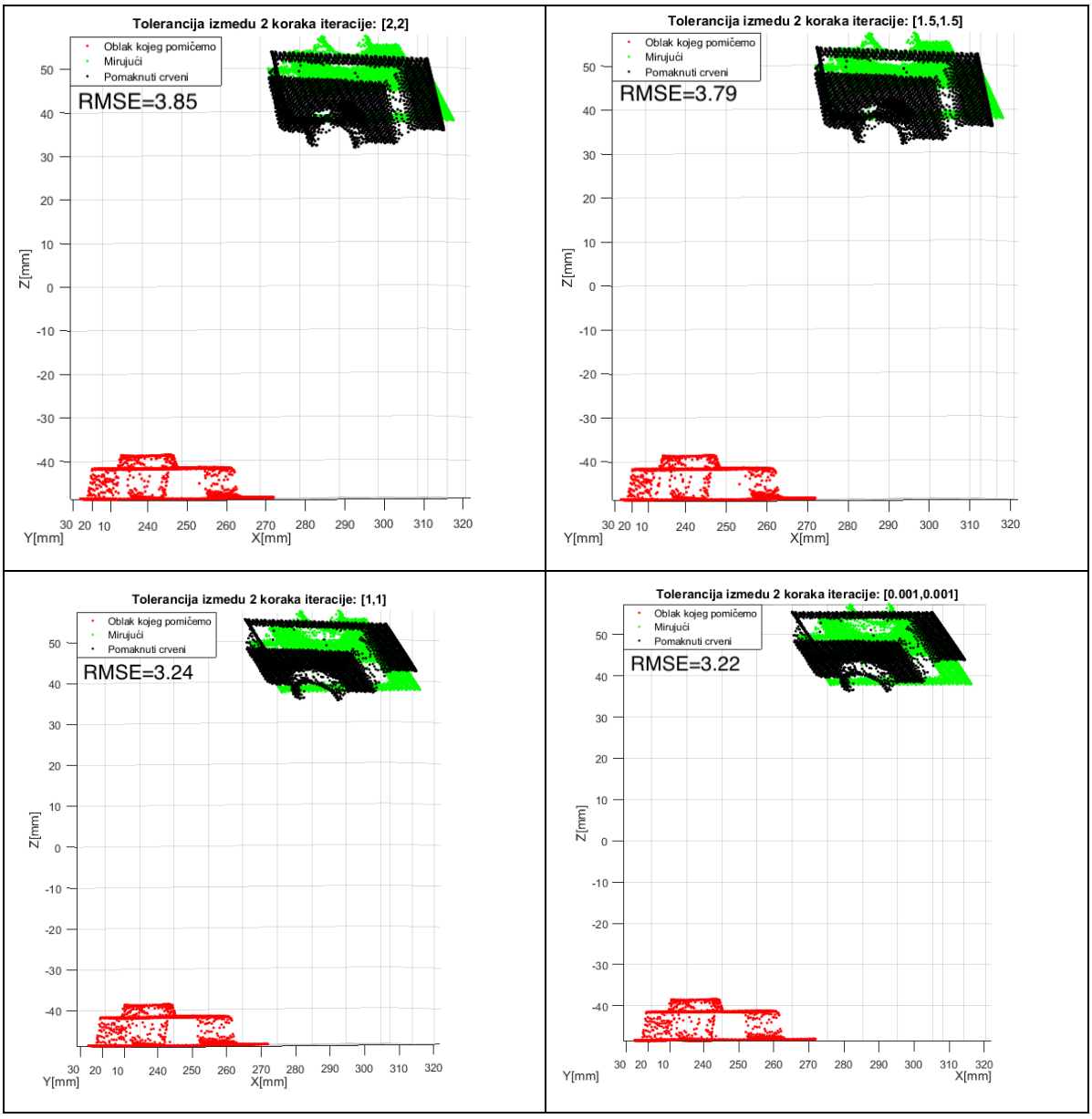
Slika 41. Greška kod korištenja 'točka na tangentnu plohu' metode

Algoritam se može zaustaviti limitiranjem broja iteracija (neprikladan način jer je teško procijeniti okviran broj iteracija), određivanjem praga greške (neprikladno zbog mogućnosti odlaska programa u beskonačnu petlju) ili definiranjem minimalne vrijednosti (tolerancije) apsolutne razlike između uzastopnih translacija. Za ovaj rad to je najprikladniji način zbog dva razloga. Prvi je taj što je eksperimentalno provjereno da će se tim načinom program uvijek zaustaviti (neće otići u beskonačnu petlju), a drugi je taj što optimalna vrijednost tih minimuma nije premala odnosno puno ne povećava trajanje procesa. Optimum je također određen eksperimentalno, tako što su pronađene vrijednosti čijim dodatnim smanjivanjem ne bi došlo do značajno boljih rezultata. Definira se vektorom koji sadrži dva člana, minimalnu razliku u translaciji i minimalnu razliku u rotaciji. Na slijedećem dijagramu [Slika 42] prikazano je kako se greška preklapanja smanjuje smanjenjem tolerancije minimalne transformacije. Iz dijagrama se može zaključiti kako je nepotrebno toleranciju smanjivati ispod 1mm za translaciju odnosno 1° za rotaciju, jer je smanjenje greške zanemarivo malo, a značajno se povećava trajanje procesa. Dijagram [Slika 42] je dobiven eksperimentiranjem s oblacima prikazanim na slici 43. Ti oblaci su zapravo identični jer smo time htjeli osigurati da greška preklapanja kod eksperimentiranja s parametrima bude u ovisnosti samo o njima (i naravno o početnom položaju, ali on je u svakoj probi bio isti, tako da je uvijek imao jednak doprinos).



Slika 42. Dijagram ovisnosti greške preklapanja o toleranciji minimalne transformacije

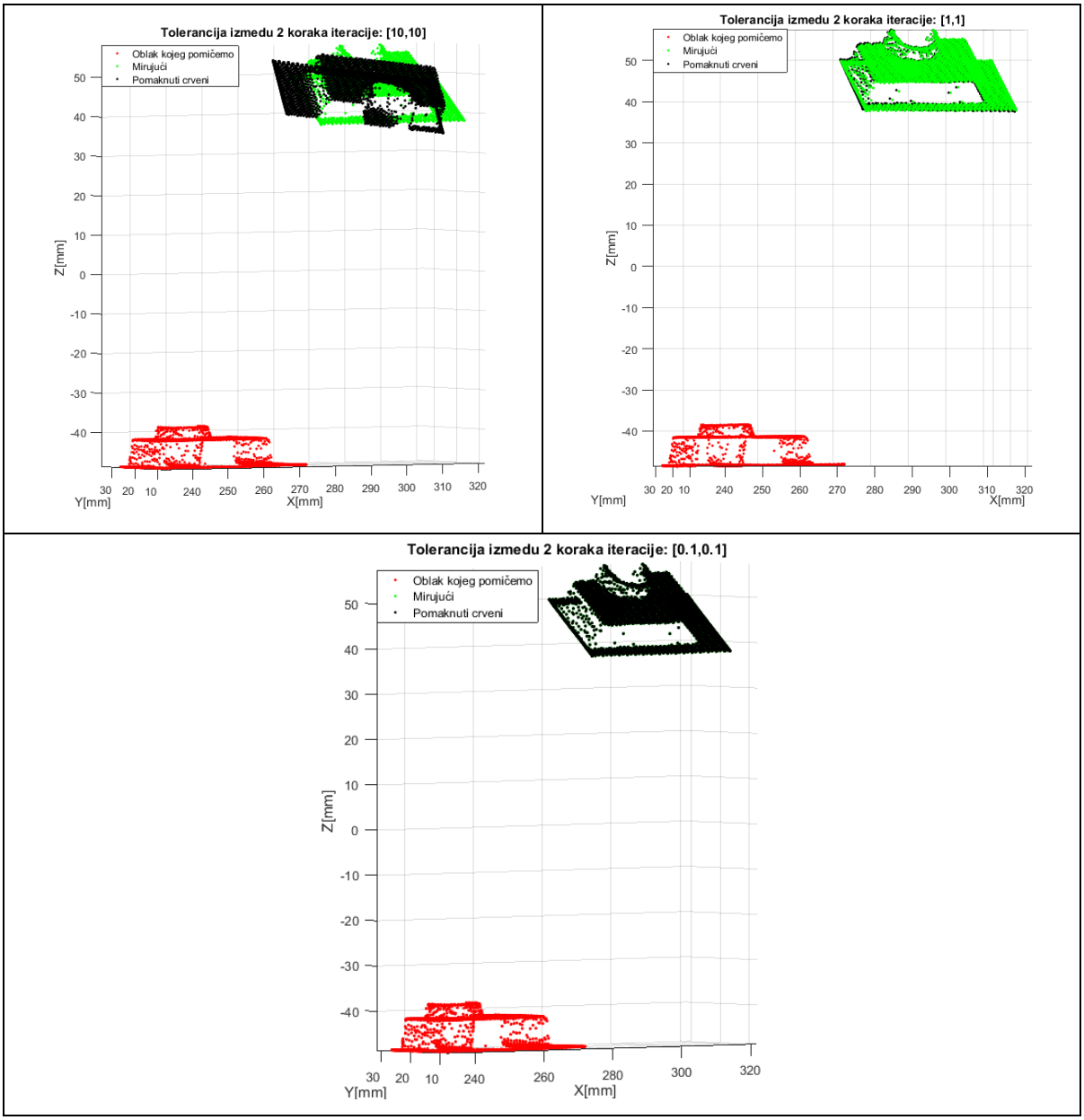




Slika 43. Prikaz preklapanja oblaka ovisno o smanjenu tolerancije minimalne transformacije

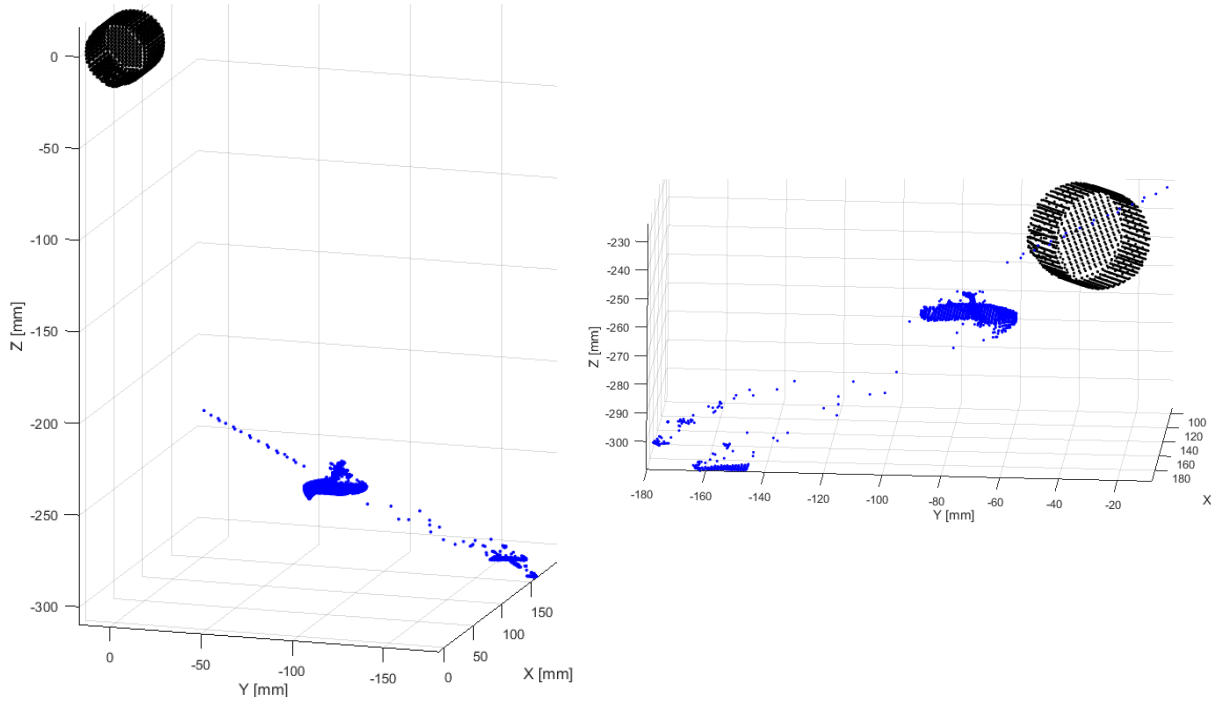
Na slici 43 se može jasno vidjeti da preklapanje oblaka točaka nije uspjelo. Crni oblak točaka je otprilike za 180° krivo zakrenut u odnosu na ciljani, zeleni, oblak, ali ujedno slika 43 dokazuje i da daljnjim smanjenjem tolerancije ne bismo ništa značajno promijenili. Kao što je već rečeno, u svakoj iteraciji algoritma točke pomičnog oblaka koje su jako udaljene od najbliže točke u nepomičnom oblaku točaka se nastoje (transformiranjem) približiti čak i pod cijenu da se druge točke pomaknu dalje od svoje najbliže točke iz prethodnog koraka. Ali u slijedećem koraku te točke koje već imaju 'blizak' par više neće tražiti novi još 'bliži' nego će algoritam u takvoj situaciji transformacijama tražiti bolje i bolje rješenje sve dok ne postigne lokalni optimum. Upravo je lokalni optimum pronađen i u slučaju na slici 43. Da bismo ipak postigli

dobro preklapanje (globalni optimum) potrebno je promijeniti početni položaj pomičnog oblaka točaka (odnosno funkciji kao ulazni podatak dostaviti matricu homogene transformacije s prvom promjenom), ali ne znamo kako i ni koliko. Eksperimentalno smo došli do zaključka da će se u gotovo svakom slučaju pronaći globalni optimum samo mijenjajući početnu orijentaciju oblaka točka (translacija se ne mijenja). S obzirom da ne znamo za koji je to kut potrebno (i da li je uopće uvijek potrebno mijenjati ga) u programu će se algoritam izvršiti za 24 različita početna kuta oko svake osi te spremati dobivenu konačnu grešku svakog izvršenja. Ako dođe do situacije da tih 72 izvršenja neće biti dovoljna u programu se lako promijene 24 različita početna kuta oko svake osi na još veći broj.



Slika 44. Prikaz preklapanja oblaka točaka nakon promjene početne orijentacije oblaka

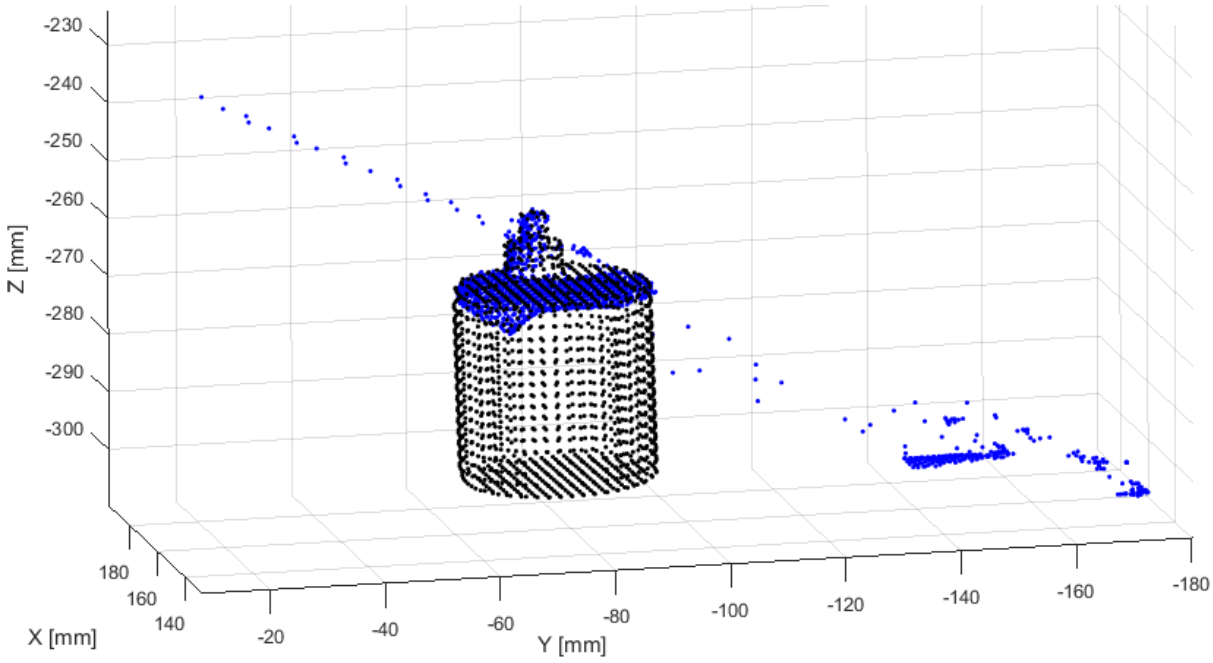
U situacijama kada postoji mnogo okolnih točaka i to dosta udaljenih od objekta mjerenja i kada je skeniran samo dio mjerenog objekta, može doći do situacije kada neće biti dovoljno samo mijenjati orijentaciju početnog položaju. Na slici 45a možemo vidjeti primjer oblaka točaka za kojeg niti jedna od mnogobrojnih kombinacija početne orijentacije ne dovodi do pronalaska globalnog optimuma već oblak točaka 'zapne', konvergira u neki lokalni minimum [Slika 45b].



a) b)

Slika 45. a) početne pozicije oblaka točaka, desno, b) loše preklapanje oblaka točaka

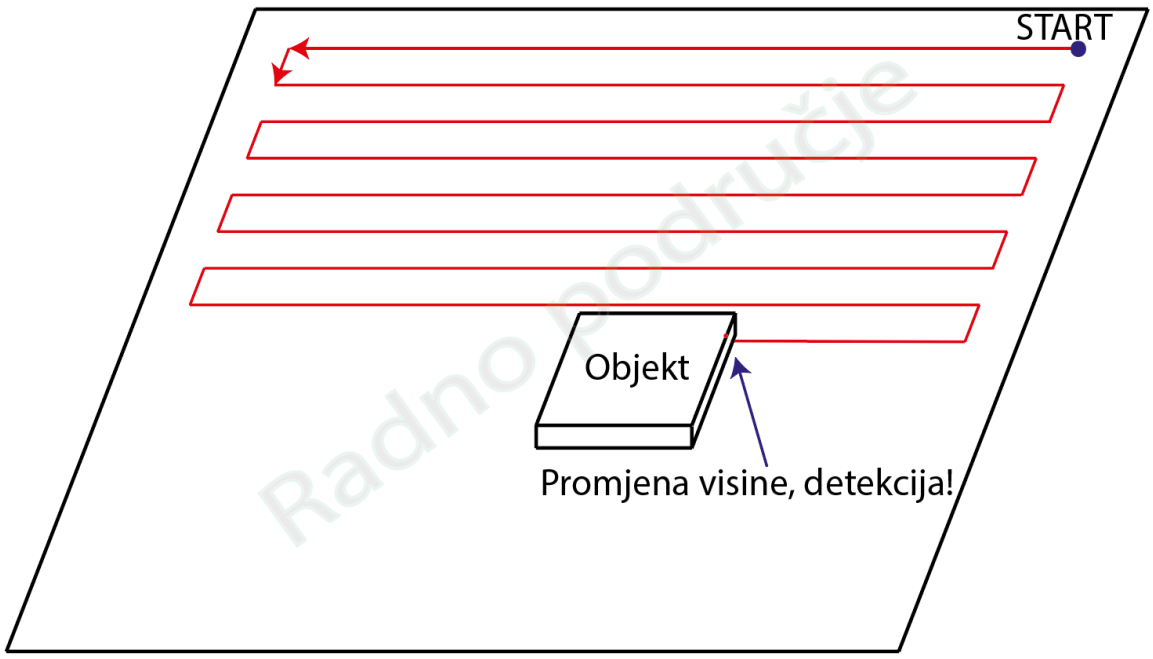
Da bi ipak došlo do dobrog preklapanja (globalnog minimuma) potrebno je promijeniti i početnu poziciju. Postavljanjem početne pozicije u područje gdje je najveća koncentracija točaka (upravo mjereni objekt) te mijenjanjem orijentacije kao i u prethodnom primjeru i s takvim 'krnjim' oblacima točaka može doći do dobrog preklapanja [Slika 46].



Slika 46. Dobro preklapanje nakon mijenjanja početnog položaja

6.3. Lokalizacija

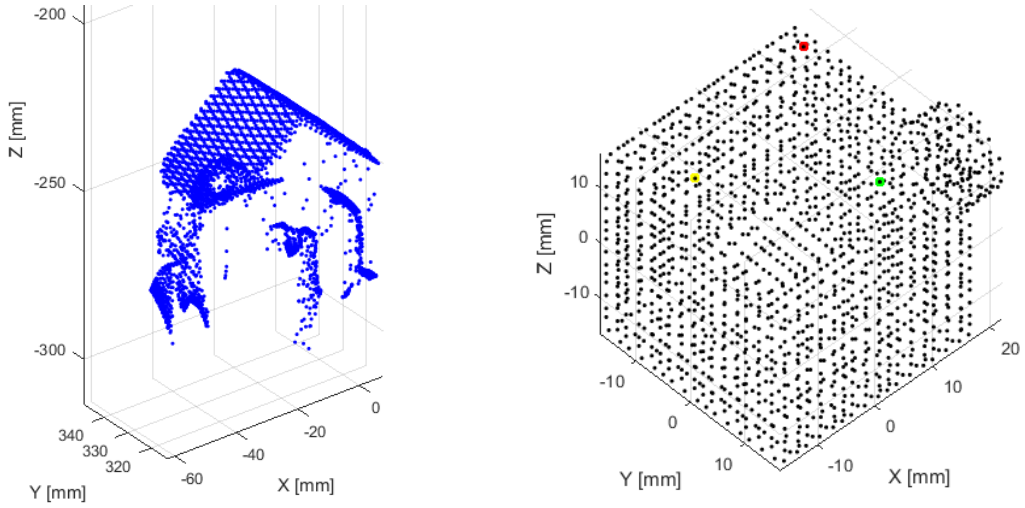
S ciljem izrade autonomnog sustava, potrebno je omogućiti i lokalizaciju objekata u robotovom radnom prostoru. Pronalazak objekata se temelji na pomicanju laserske zrake kroz radno područje i pronalaženju drastičnih promjena visine. Prva, okvirna detekcija je ostvarena korištenjem putanje sa slike 47.



Slika 47. Detekcija objekta

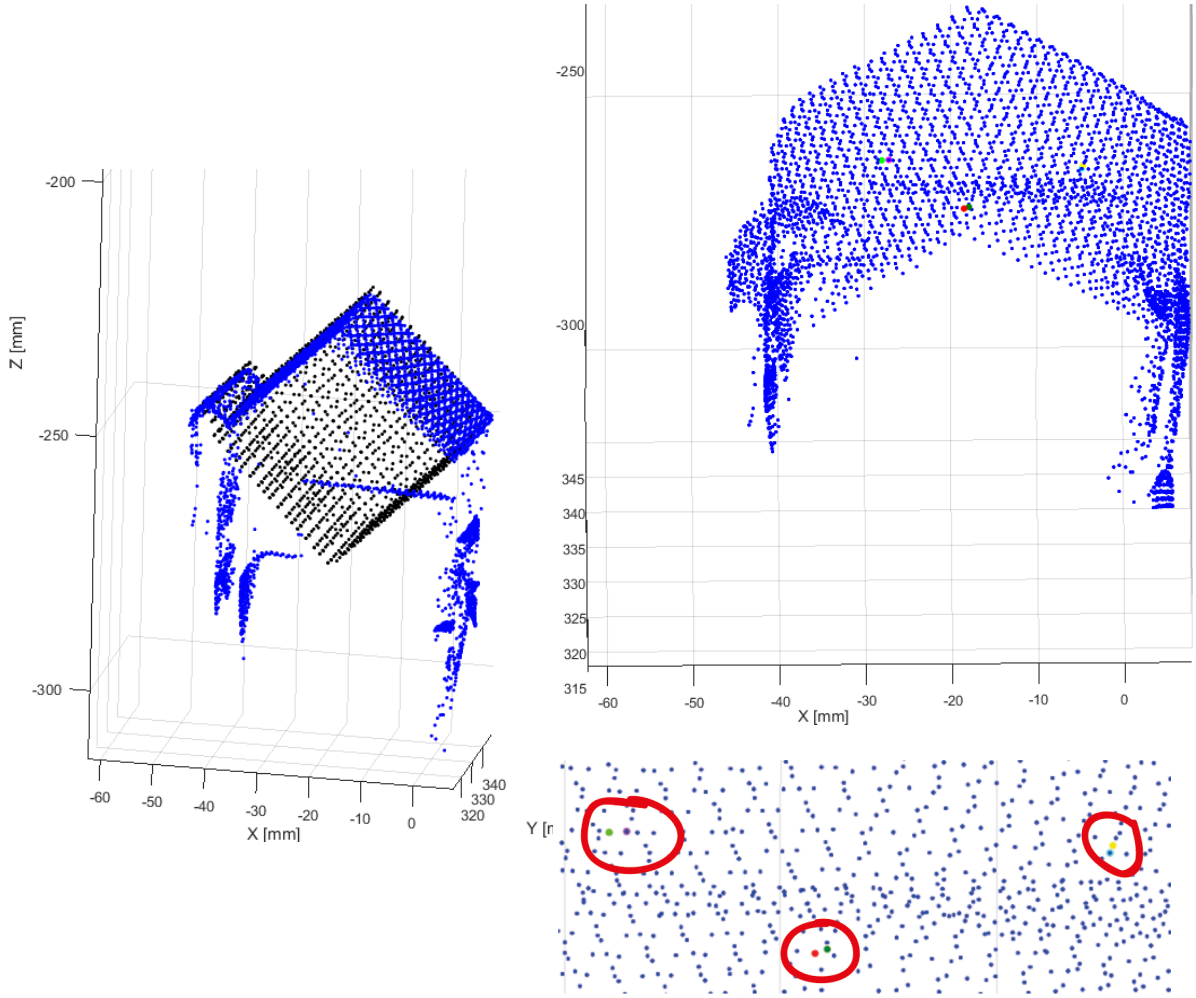
Da bismo bili sigurni da nećemo promašiti objekt, razmak između dviju linija skeniranja mora biti primjeren. U slučaju kada su nam poznate dimenzije objekta, razmak može biti jednak najmanjoj dimenziji objekta. U trenutku kada je objekt detektiran, moguće je odmah početi sa skeniranjem, ali u tom slučaju, kako bismo bili sigurni da ćemo odskenirati cijeli objekt lasersku zraku moramo odmaknuti i u smjeru osi X i Y za najveću dimenziju objekta (jer ne znamo na kojem dijelu objekta se dogodila detekcija) što će u nekim slučajevima uzrokovati stvaranje velikog oblaka točaka s mnogo nepotrebnih točaka iz okruženja te posljedično oduljiti proces. Bolji način je da, nakon što je prva točka detektirana, pronađemo još dvije točke na objektu, jednu s najmanjom X koordinatom i jednu s najmanjom Y koordinatom i nakon toga iskoristimo te koordinate za 'stvaranje' točke iz koje će početi proces skeniranja. Oblak točaka dobiven takvim načinom će imati i dalje, u nekim orijentacijama objekta, mnogo nepotrebnih točaka iz okruženja, ali ako ćemo tražiti još točaka na objektovom rubu izgubit ćemo još više vremena tak da je bolje skenirati nakon pronađene samo tri točke.

Nakon što je oblak točaka dobiven i eventualno promijenjen na neki način (translatiran, rotiran, obrisane neželjene točke...) potrebno je provesti konačan cilj ovoga rada, odnosno pozicionirati robota (vrh laserske zrake) u određeni dio objekta koji je skeniran. Za to nam je potreban drugi oblak točaka istog objekta, koji može biti stvoren u nekom CAD programu (postupak je dan u prilogu 5), ili je mogao nastati također skeniranjem, ovim ili nekim drugim sustavom. Na njemu označimo točku (ili više njih) u koju želimo da kasnije pokazuje laserska zraka, na objektu. Oblaci su prikazani na slici 48.

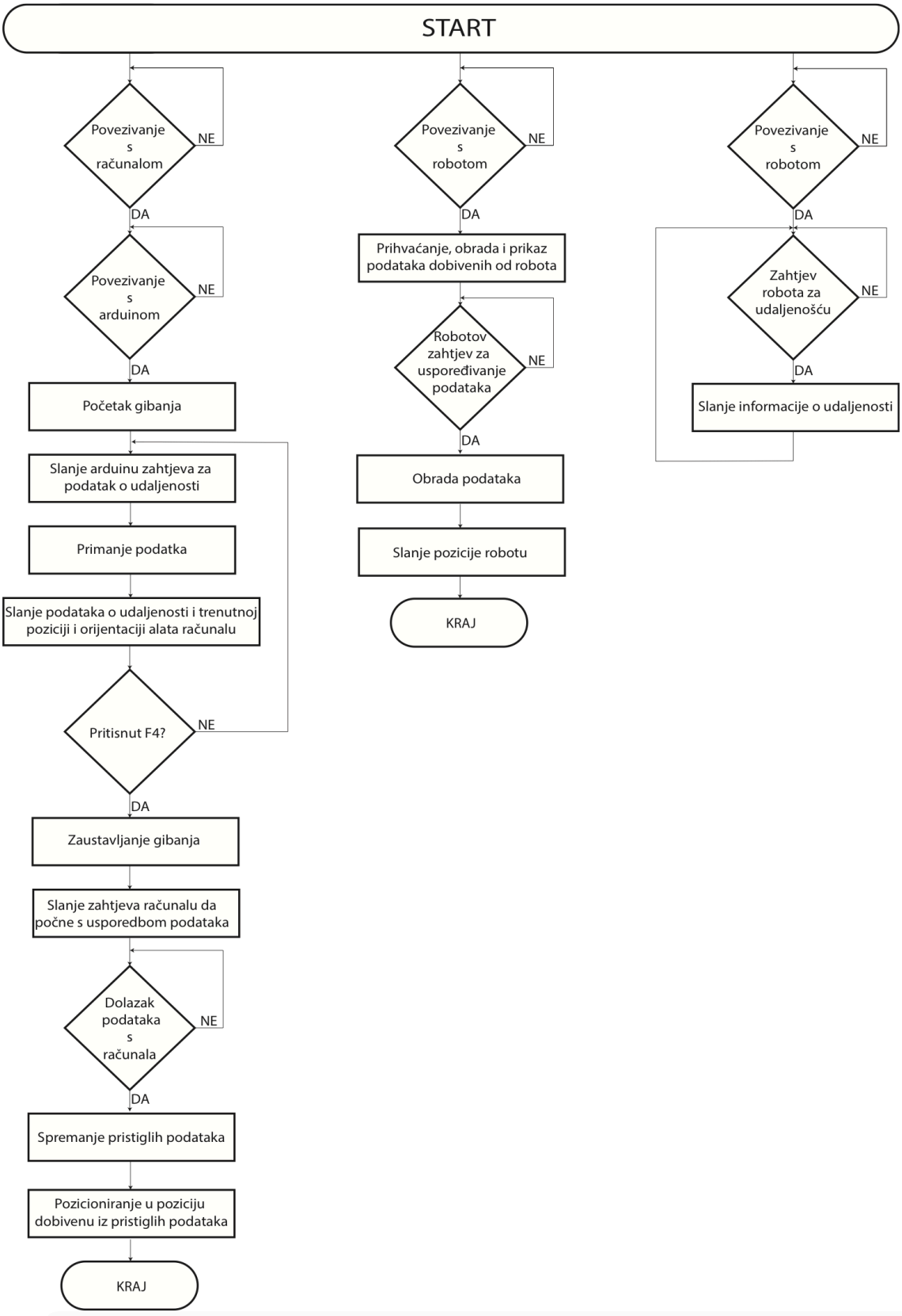


Slika 48. Lijevo, oblak točaka dobiven iz sustava, desno, oblak točaka stvoren u CAD-u s tri točke u koju želimo pozicionirati lasersku zraku

Nakon odabiranja točke provodi se postupak preklapanja dobivenog oblaka točaka i oblaka iz CAD-a ranije objašnjenim ICP algoritmom. Dobiveni oblak točaka je nepomičan, a oblak iz CAD-a se zakreće i pomiče sve dok najbolje 'ne poklopi' nepomičan. Nakon preklapanja, na dobivenom oblaku točaka tražimo točke koje su najbliže točkama koje smo označili na oblaku iz CAD-a [Slika 49]. Crvena, svijetlo zelena i žuta točka su točke iz oblaka dobivenog u CAD-u (odabrane), a ljubičasta, tamno zelena i svijetlo plava su njima najbliže točke dobivenog oblaka, u njih ćemo pozicionirati lasersku zraku. To je moguće jer je uz svaku točku oblaka spremana informacija o poziciju i orijentaciji vrha alata iz koje je točka izračunata. Potrebno je samo poslati koordinate upravljačkoj jedinici robota koja konačno usmjeri lasersku zraku u željene pozicije. Pojednostavljeni dijagram toka konačnog sustava je prikazan na slici 50.



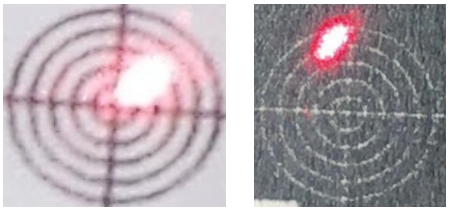
Slika 49. Lijevo, preklop oblaka, desno dobiveni oblak točaka (dolje, desno uvećano područje s odabranim točkama)



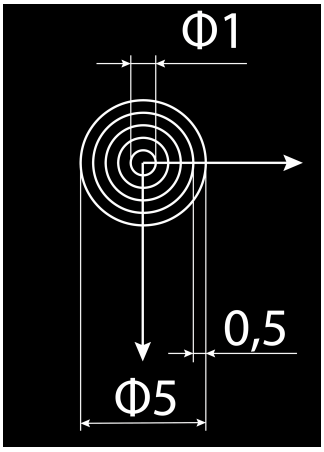
Slika 50. Pojednostavljeni dijagram toka sustava

6.4. Greške sustava

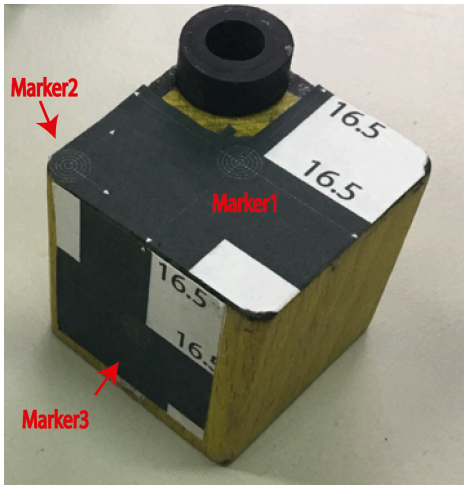
Da bismo mogli provjeriti koliko smo blizu željene točke usmjerili lasersku zraku, na objekt smo nalijepili kružnice s namjerom da im je središte poklopljeno sa željenim točkama. Promjer najmanje je 1 mm dok je promjer svake slijedeće za 1 mm veći [Slika 52]. Odabrana je crna podloga jer je na njoj točka lasera malo bolje vidljiva. [Slika 51].



Slika 51. Usporedba bijeli i crne pozadine

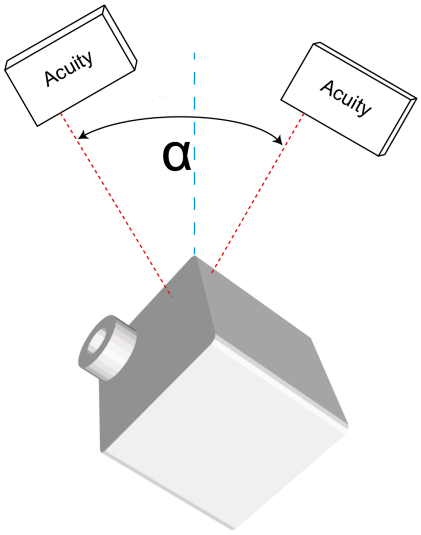


Slika 52. Marker

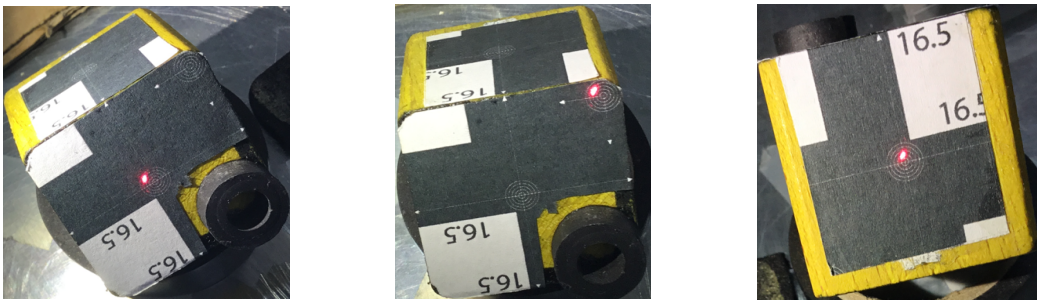


Slika 53. Markeri na objektu mjerenja

Mjerenja smo provodili tako da smo za svako novo povećali kut između dviju pozicija skeniranja. Iz svake pozicije je skenirana jedna ploha [Slika 54]. Počeli smo s kutom 0°, odnosno kod prvog skeniranja nije bilo promijene pozicije.



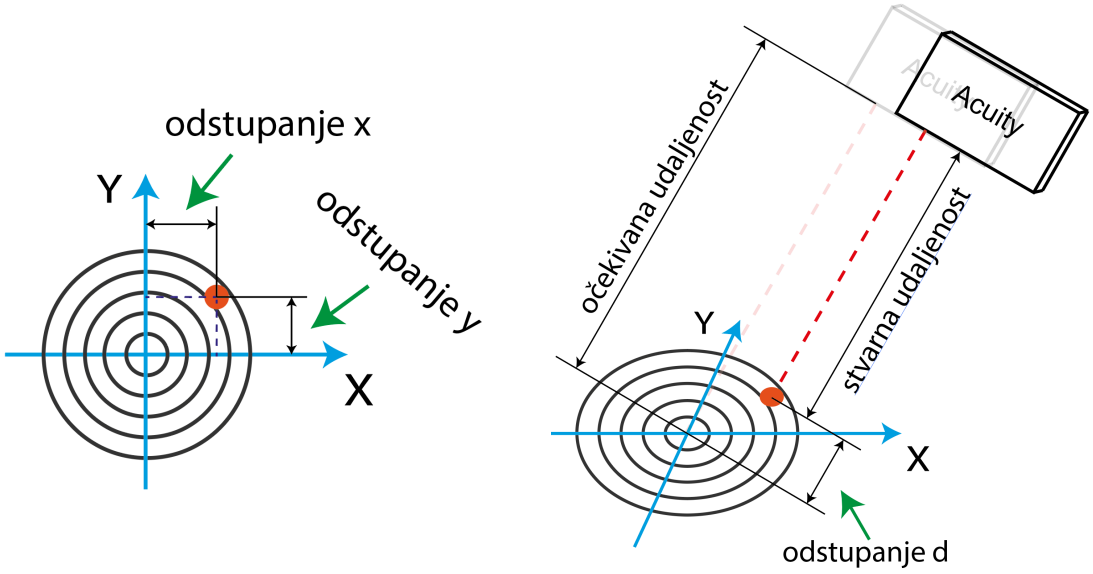
Slika 54. Skica načina mjerenja



Slika 55. Rezultat jednog mjerenja

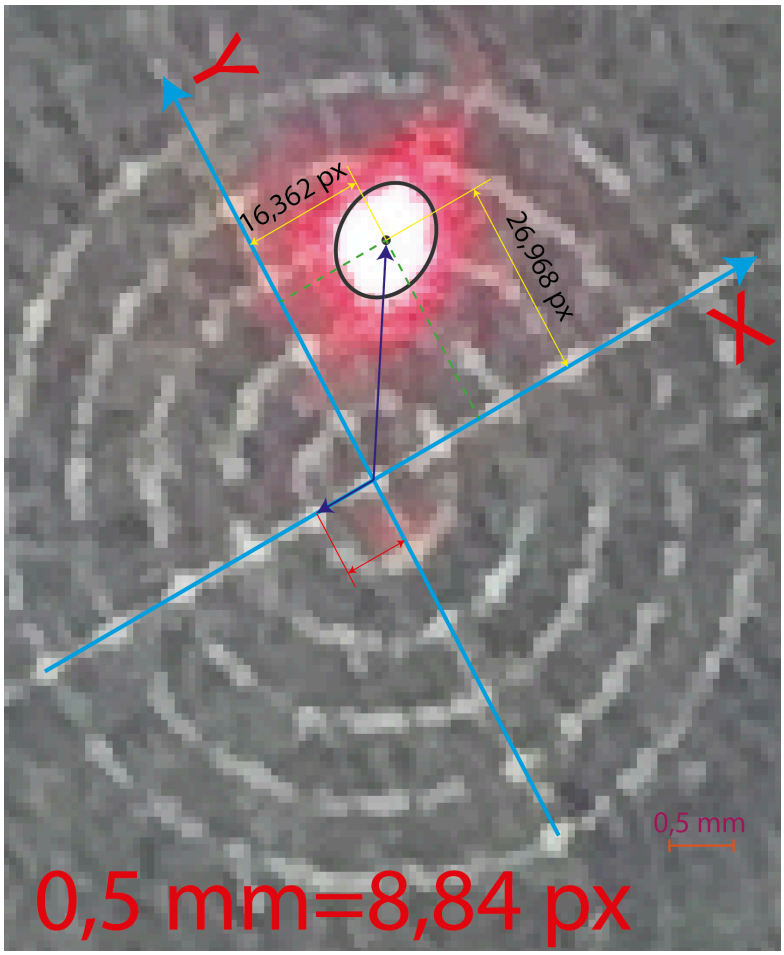
Nakon skeniranja smo mjerili (na fotografijama [Tablica 12]) koliko točka lasera odstupa od centra kružnica, odnosno očekivane, željene pozicije [Slika 56a i Slika 57], i kolika je razlika udaljenosti između dobivenog položaja i senzora te željenog položaja (centra kružnice) i senzora, odnosno razlika u visini – odstupanje d [Slika 56b]. Rezultati mjerenja su dani u tablici 13. Iz tih podataka izračunavamo prostorne greške [Tablica 15] jednostavnim izrazom za duljinu vektora:

$$Greška = \sqrt{X^2 + Y^2 + d^2} \tag{9}$$



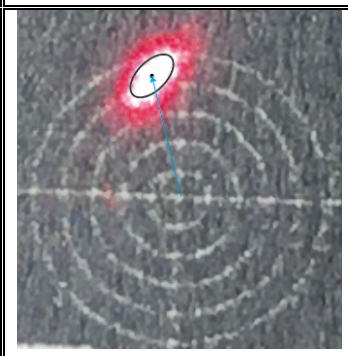
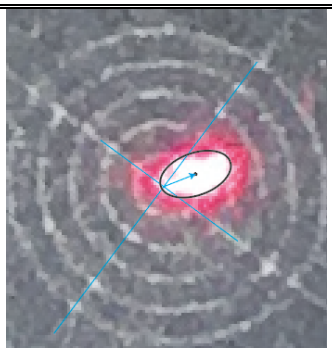
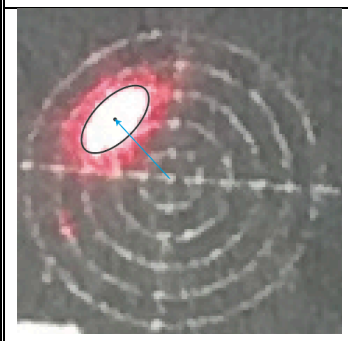
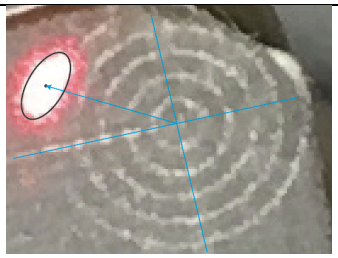
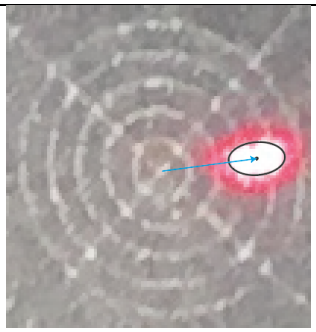
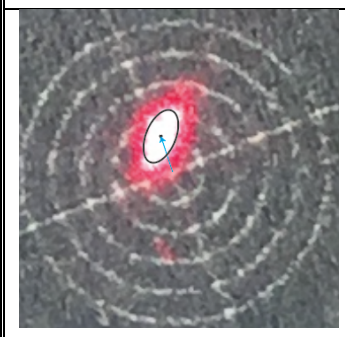
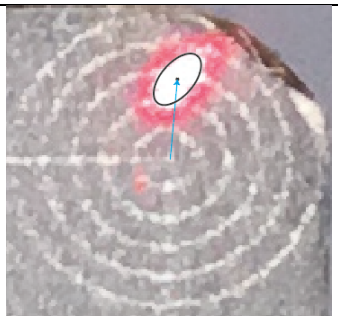
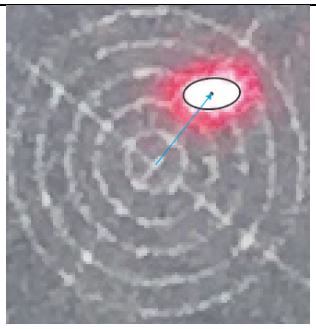
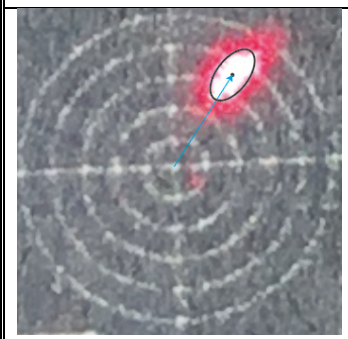
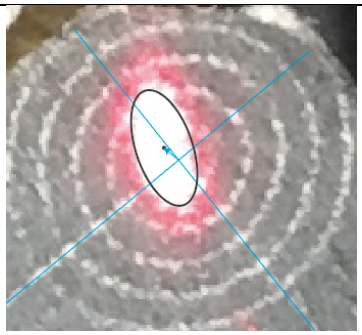
a) b)

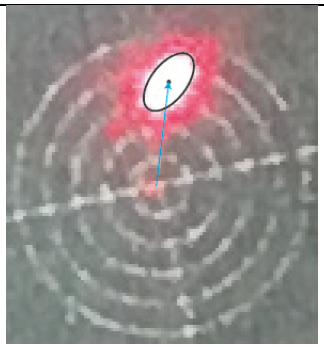
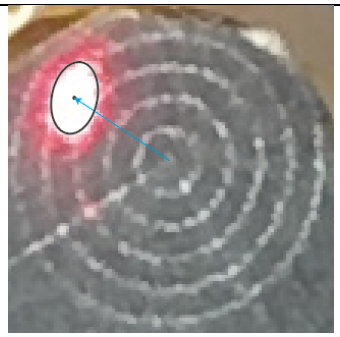
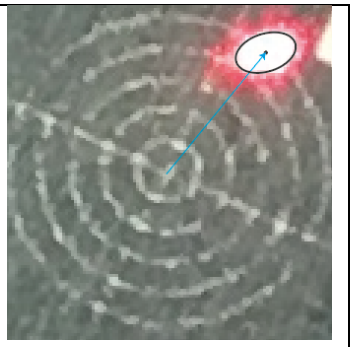
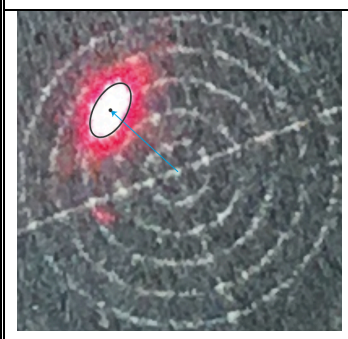
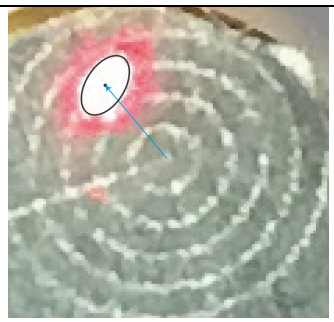
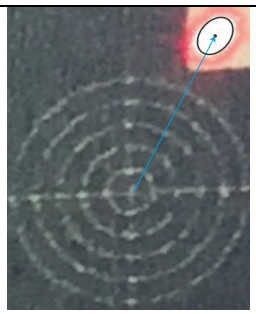
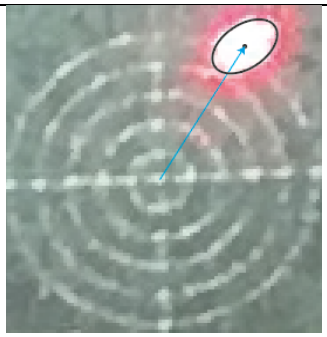
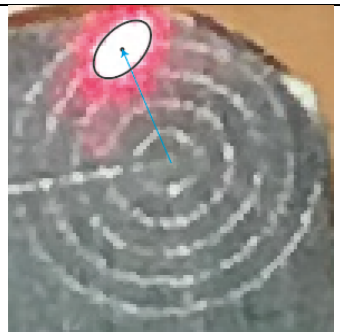
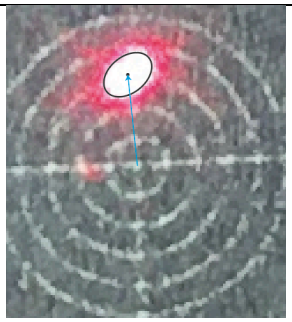
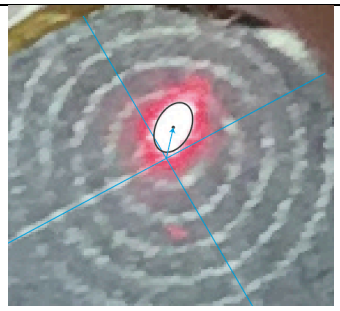
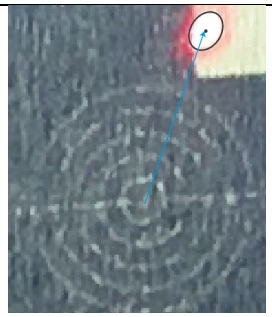
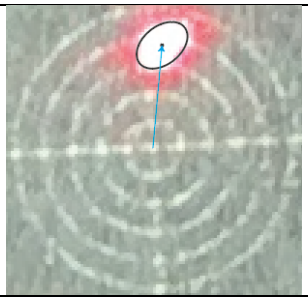
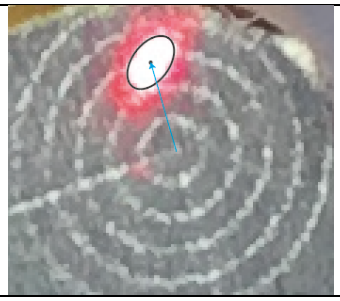
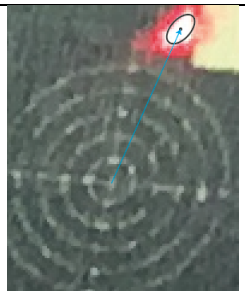
Slika 56. Skice postupka mjerenja pogreške

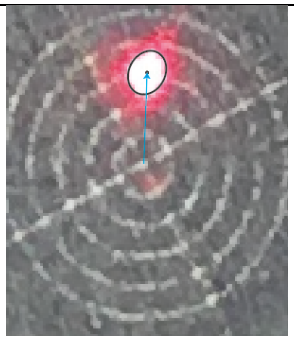
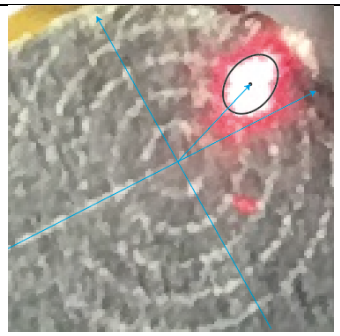
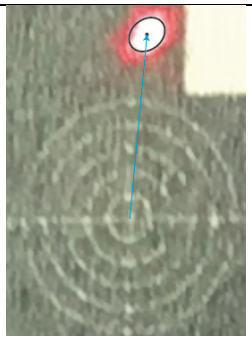
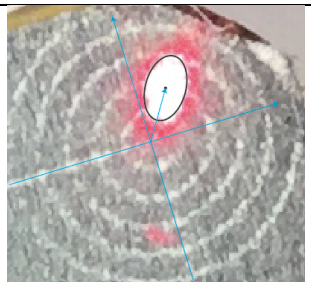
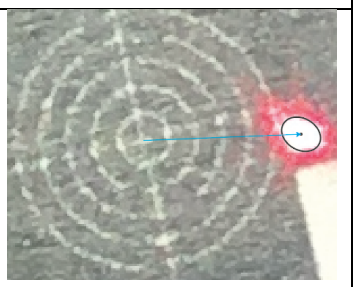
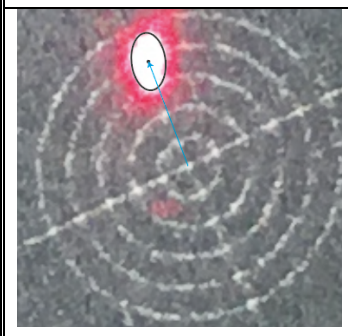
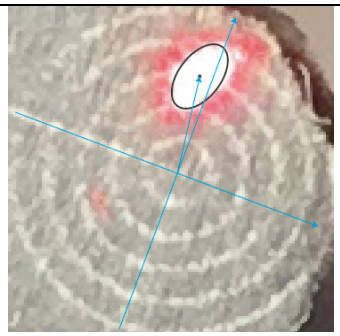
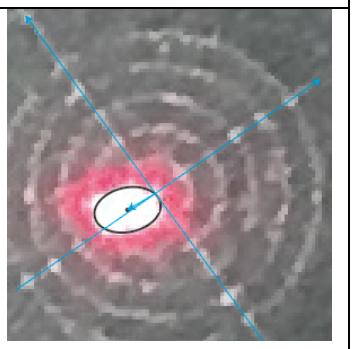


Slika 57. Mjerenje ravninske greške

Tablica 12. Fotografije rezultata skeniranja kocke, uz mijenjanje orijentacije alata robota tijekom gibanja

Kut α:	Marker 1	Marker 2	Marker 3
0°			
10°			
20°			
30°			

40°			
50°			
60°			
70°			
80°			

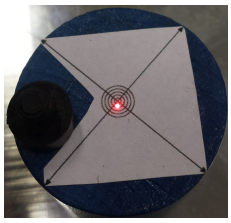
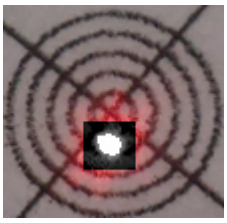
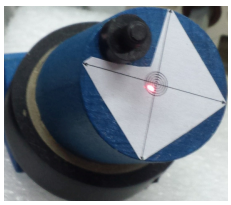
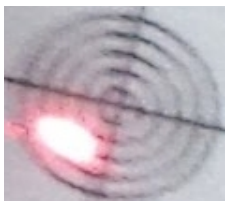
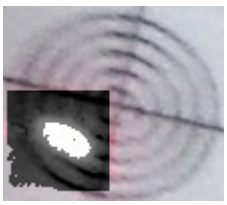
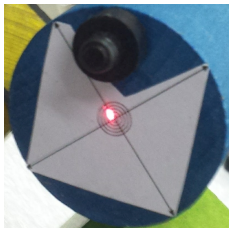
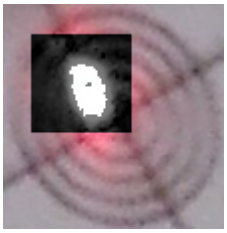
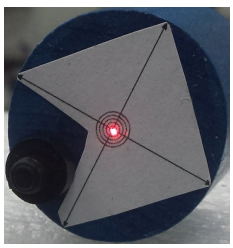
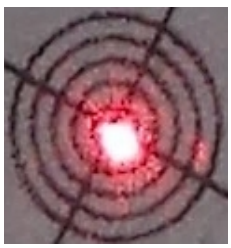
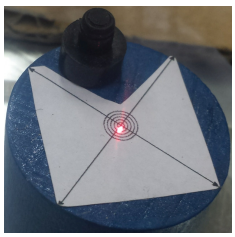
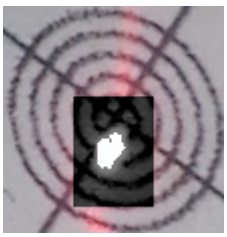
90°			
45° Kružno gibanje			
15° Kružno gibanje			

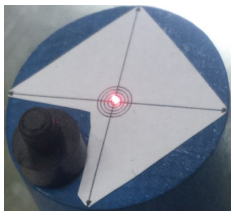
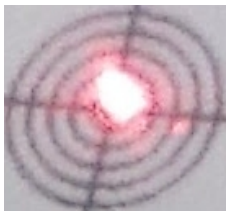
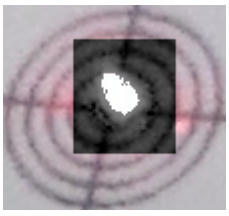
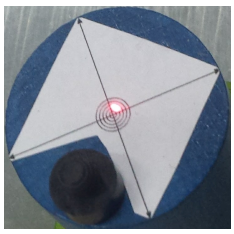
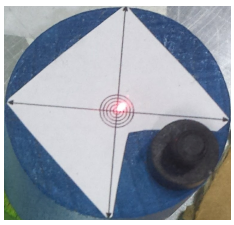
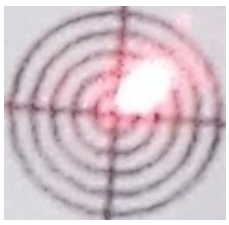
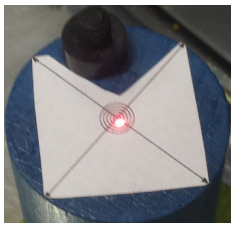
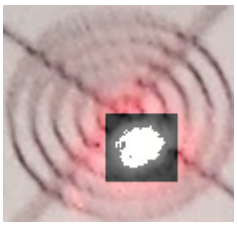
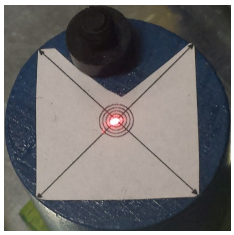
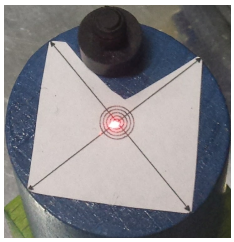
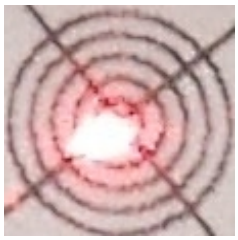
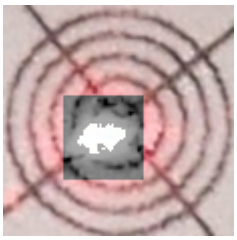
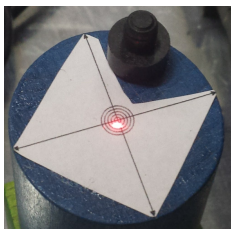
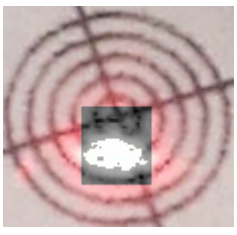
Tablica 13. Rezultati skeniranja kocke, uz mijenjanje orijentacije alata robota tijekom gibanja

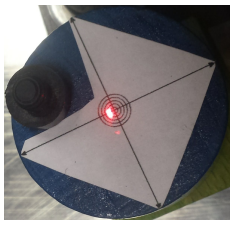
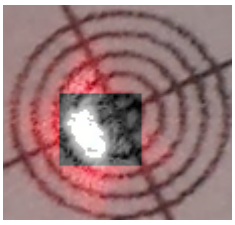
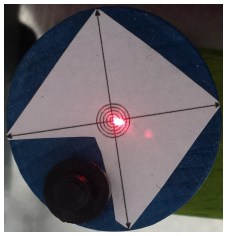
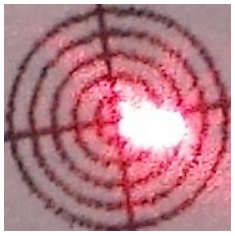
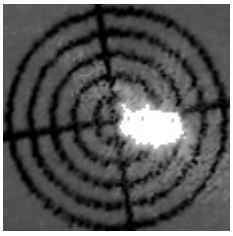
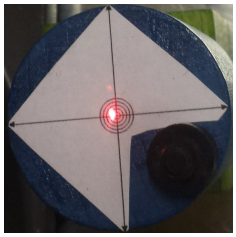
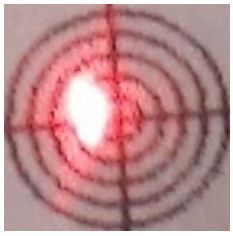
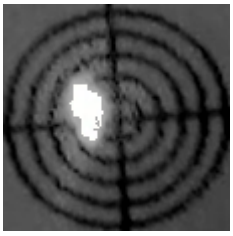
Kut α: [Slika 54]	Marker 1			Marker 2			Marker 3		
	X[mm]	**Y**[mm]	**d**[mm]	**X**[mm]	**Y**[mm]	**d**[mm]	**X**[mm]	**Y**[mm]	**d**[mm]
0°	0,58	2,02	1,49	0,14	1,25	0,86	0,28	0,47	0,03
10°	0,98	0,87	0,74	2,23	1,21	0,90	1,19	1,31	0,69
20°	0,06	0,58	0,50	0,13	1,38	0,91	0,07	1,43	0,97
30°	0,89	1,32	1,10	0,06	0,28	0,06	0,55	2,68	1,74
40°	0,42	1,74	1,21	0,79	1,70	1,15	0,58	2,47	1,65
50°	0,62	1,25	1,10	0,53	1,44	1,00	1,77	3,39	2,55
60°	1,45	2,30	1,95	0,44	2,17	1,70	0,10	2,91	2,16
70°	0,11	1,51	1,40	0,33	0,36	0,26	1,46	3,49	2,98
80°	0,25	1,79	1,25	0,01	1,50	1,34	0,95	3,40	2,83
90°	0,76	1,26	1,26	1,46	0,52	0,54	0,34	4,07	3,58
90° Kružno gibanje	0,27	1,73	1,12	0,51	0,81	0,58	3,08	0,64	2,01
30° Kružno gibanje	0,14	1,84	1,15	0,19	1,49	0,99	0,04	0,50	0,43

Puno faktora ujače na konačnu grešku, preciznost lijepljenja markera, gustoća točaka skeniranja (jer tražimo najbližu točku, a ako su točke vrlo rijetke moguće je da je njena udaljenost velika), mjerenje prije konstrukcije modela u CAD-u, otklon zrake od smjera normale plohe tijekom skeniranja, nesavršenost preklapanja (usporedbe) oblaka, točnost pozicioniranja robota, ali daleko najviše utjecaja na grešku ima točnost kalibracije alata što se može potvrditi s dobivenim rezultatima [Tablica 13] koji su sve lošiji s većim zakretom alata, dok u rezultatima [Tablica 14] kod koje nije bilo promjene orijentacije alata kod skeniranja, vidimo da su rezultati približno isti (svi unutar 2 mm) bez obzira na položaj i orijentaciju objekta.

Tablica 14. Fotografije rezultata skeniranja jedne plohe objekta na različitim položajima i s različitim orijentacijama. Tijekom pojedinog skeniranja orijentacija robota se nije mijenjala

			Okomito valjak, laser u smjeru normale plohe
			Koso valjak, laser u smjeru normale plohe
			Valjak polegnut (horizontalno), laser u smjeru normale plohe
			Valjak polegnut (zakrenut za 180° oko okomice od prethodnog slučaja), laser u smjeru normale plohe
			Valjak okomito, laser zakrenut za 15° od normale plohe

			Valjak okomito, laser zakrenut za 20° od normale plohe
			Valjak okomito, laser zakrenut za 25° od normale plohe
			Valjak okomito, laser zakrenut za 30° od normale plohe
			Valjak okomito, laser zakrenut za 35° od normale plohe
			Valjak okomito, laser zakrenut za 40° od normale plohe
			Valjak okomito, laser zakrenut za 45° od normale plohe
			Valjak okomito, laser zakrenut za 50° od normale plohe

			Valjak okomito, laser zakrenut za 55° od normale plohe
			Valjak okomito, laser zakrenut za 60° od normale plohe
			Valjak okomito, laser zakrenut za 65° od normale plohe

Tablica 15. Sumirani rezultati testiranja kocke kod kojih se mijenjala orijentacija alata robota tijekom gibanja

Kut α:	Prostorna greška [mm]			
	Marker 1	Marker 2	Marker 3	Avg
0°	2,57	1,52	0,55	1,55
10°	1,50	2,69	1,90	2,03
20°	0,77	1,66	1,73	1,39
30°	1,94	0,29	3,24	1,82
40°	2,16	2,20	3,03	2,46
50°	1,78	1,83	4,60	2,73
60°	3,34	2,79	3,63	3,25
70°	2,06	0,55	4,82	2,48
80°	2,20	2,01	4,52	2,91
90°	1,94	1,64	5,43	3,00
90° Kružno gibanje	2,08	1,12	3,73	2,31
30° Kružno gibanje	2,17	1,80	0,66	1,54
avg	2,04	1,68	3,15	2,29
max	3,35	2,79	5,43	3,25
min	0,77	0,29	0,55	1,39

7. ZAKLJUČAK

Laserski mjerač udaljenosti AccuRange AR700 je kvalitetan uređaj velikih mogućnosti i preciznosti, no s manom što se ne može priključiti na moderne uređaje zbog zastarjele vrste komunikacije te mu se stoga moralo omogućiti komuniciranje modernijim načinom - TCP/IP protokolom, što je bio preduvjet nužan za daljnje ostvarenje sustava za 3D skeniranje, glavnog cilja ovog rada koji je nastao povezivanjem mjerača udaljenosti, Arduina, robota i računala. Za izradu tog sustava potrebno je poznavanje razmjene podataka na principu 'korisnik-poslužitelj', programiranja u Matlabu, Karelu i Arduinovom programskom jeziku kao i razumijevanje matematičkih algoritama za izračunavanje prostornih transformacija.

Cilj rada je ostvaren, razvijen je sustav koji će skenirati objekte u radnom prostoru robota, odnosno stvoriti oblak točaka, koji je osnovna za traženje značajki na objektu skeniranja odnosno usmjeravanja laserske zrake u nju. Također je omogućeno pronalaženje objekta u radnom prostoru robota.

No, nažalost nije sve idealno, te sigurno postoji prostor za unaprijeđenije sustava, ponajviše kod kalibracije alata koju bi se u budućim radovima moglo pokušati poboljšati nekim drugim metodama te tako smanjiti grešku sustava koja se u ovakvom postavu povećava sa sve većim zakretom alata. Također kod lokalizacije razvijene u ovom radu ima mnogo potencijalnog prostora za napredak, najviše kod njene brzine pronalaska početne točke objekta za skeniranje. Ujedno bi se moglo, da sustav bude pouzdaniji i sigurniji za korištenje, programski ograničiti izlazak robota iz radnog područja mjerača udaljenosti koji ima striktne granice koliko smije biti približen odnosno udaljen od objekta; i pokušati programski spriječiti da se senzor pozicionira u pozicije u kojima ne može obaviti triangulaciju (zaklonjen prolaz odbijenoj zraci na putu nazad u senzor). Zasigurno bi bilo poželjno i promijeniti sustav validacije rezultata, da se izbaci dio s ručnim lijepljenjem markera na objekte upitne točnosti oblika.

LITERATURA

[1] https://en.wikipedia.org/wiki/RS-232

[2] https://hr.wikipedia.org/wiki/TTL

[3] https://www.sparkfun.com/tutorials/215

[4] https://web.math.pmf.unizg.hr/nastava/rp2/pred10/pred10.html

[5] https://en.wikipedia.org/wiki/OSI_model - Layer_3:_Network_Layer

[6] http://titan.fsb.hr/~bosekora/nastava/ims/ims-Vjezba1.pdf

[7] https://www.arduino.cc/

[8] https://www.youtube.com/watch?v=vcOE2XAQHzY

[9] http://www.fanucrobotics.com/cmsmedia/datasheets/LR Mate 200iC Series_10.pdf

[10] ar700-data-sheet.pdf

[11] ar700-user-manual.pdf

[12] https://en.wikipedia.org/wiki/Iterative_closest_point

[13] https://www.fsb.unizg.hr/aero/images/books/01_Kinematika.pdf

[14] http://zemris.fer.hr/predmeti/rg/diplomski/04Neuhold/problem3.htm

[15] https://www.mathworks.com/matlabcentral/fileexchange/22409-stl-file-reader

[16] https://grabcad.com/questions/tutorial-excel-to-catia-how-to-create-import-points-in-catia-by-their-coordinate-existed-in-an-excel-file

[17] Matijević, H., Lapaine, M., Kapović, Z. (2006). Analiza pomaka i deformacija ravninskih ploha primjenom ravnine najmanjih kvadrata. *Kartografija i geoinformacije, 5(6)*, 27-38. Preuzeto s https://hrcak.srce.hr/6362

[18] M. Švaco, B. Šekoranja, F. Šuligoj, and B. Jerbić, "Calibration of an Industrial Robot Using a Stereo Vision System," Procedia Eng., vol. 69, pp. 459–463, 2014.

[19] https://www.mathworks.com/matlabcentral/fileexchange/22409-stl-file-reader

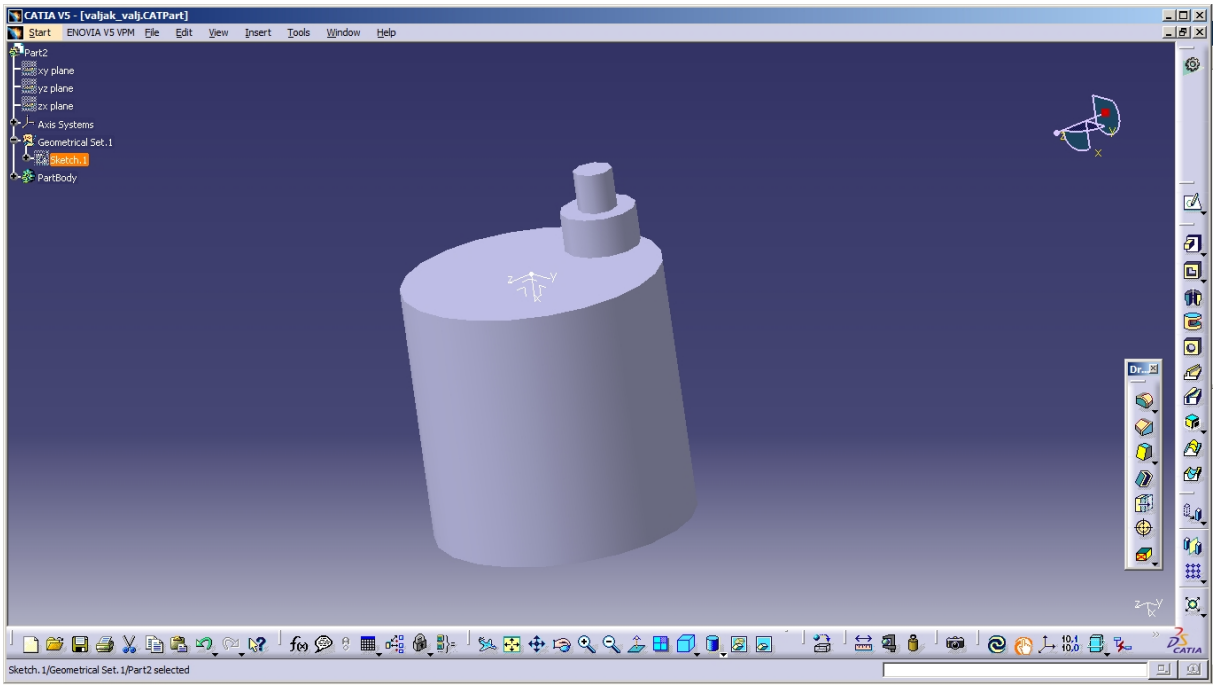
PRILOZI

I. Postupak pretvorbe CAD u oblak točaka
II. Postupak pretvorbe oblaka točaka u CAD
III. Programski kod KAREL
IV. MATLAB programski kod
V. MATLAB funkcija za lim
VI. Teach Pendant programski kod
VII. CD-R Disk

PRILOG I

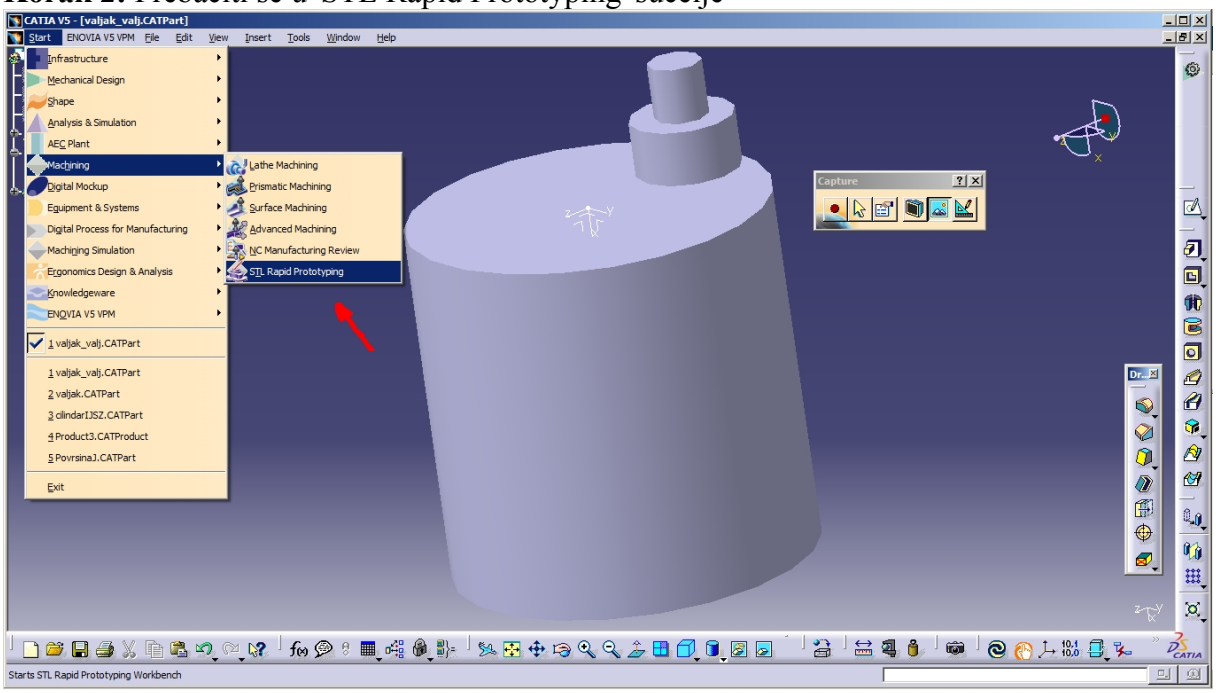
Postupak pretvorbe CAD-a u oblak točaka (CATIA → MATLAB)

Korak 1: Napraviti željni model u 'Part design'



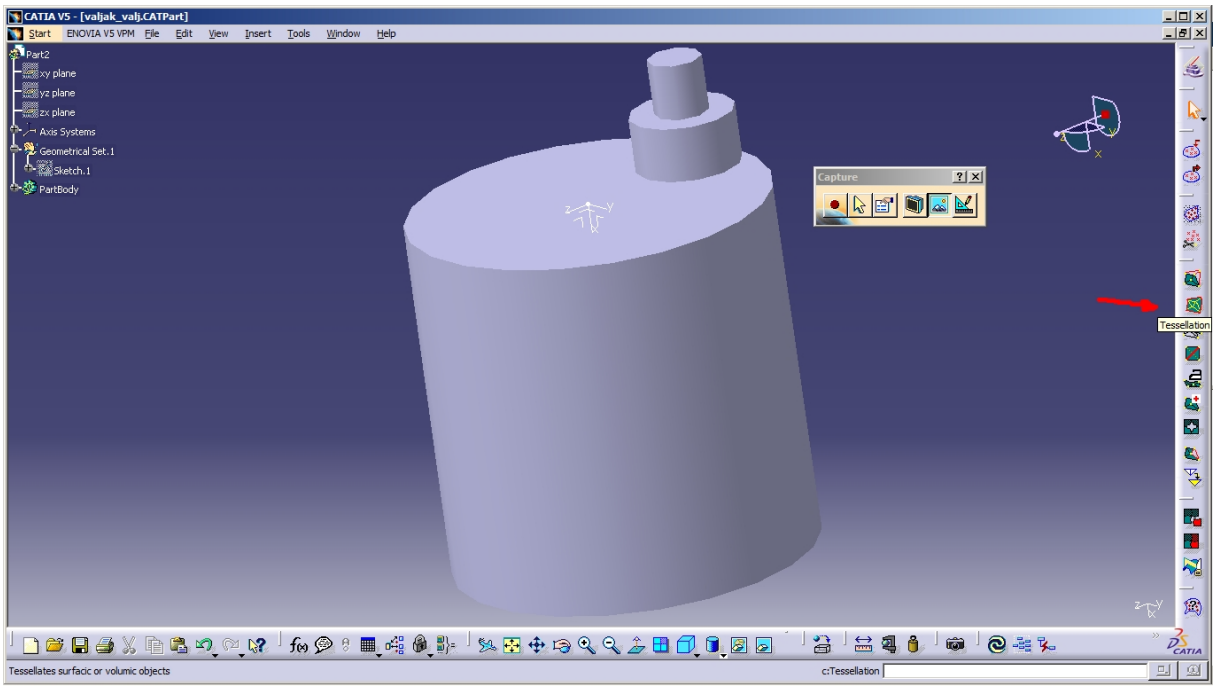
Slika 58. Model

Korak 2: Prebaciti se u 'STL Rapid Prototyping' sučelje



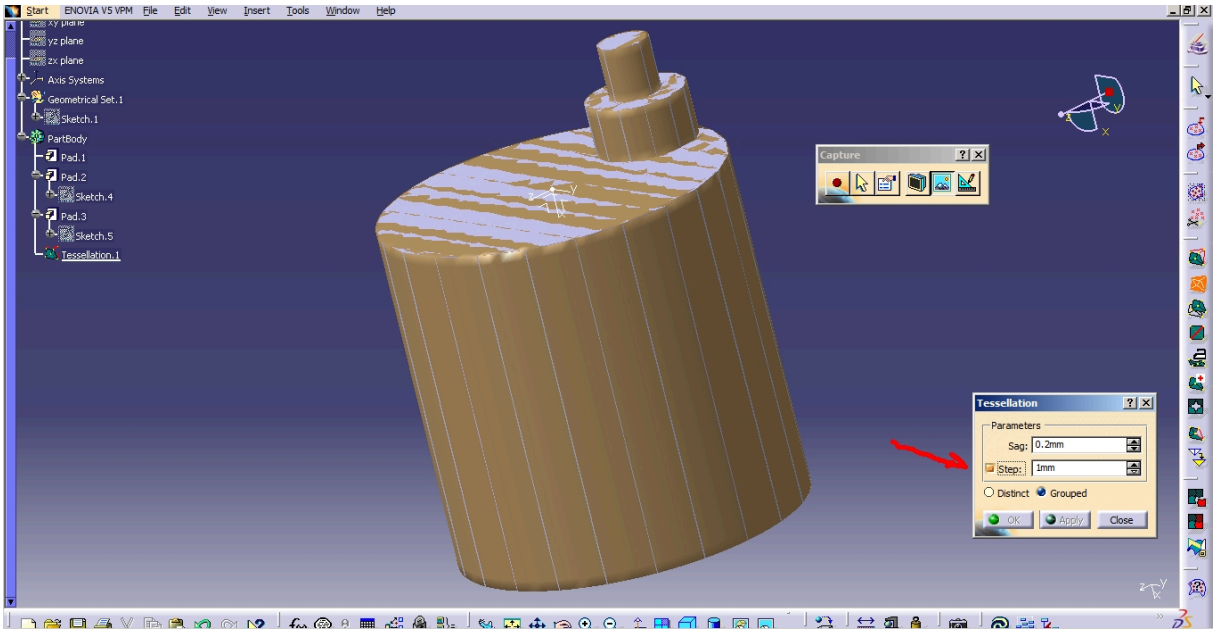
Slika 59. Promjena sučelja

Korak 3: Odabrati 'Tessellation'



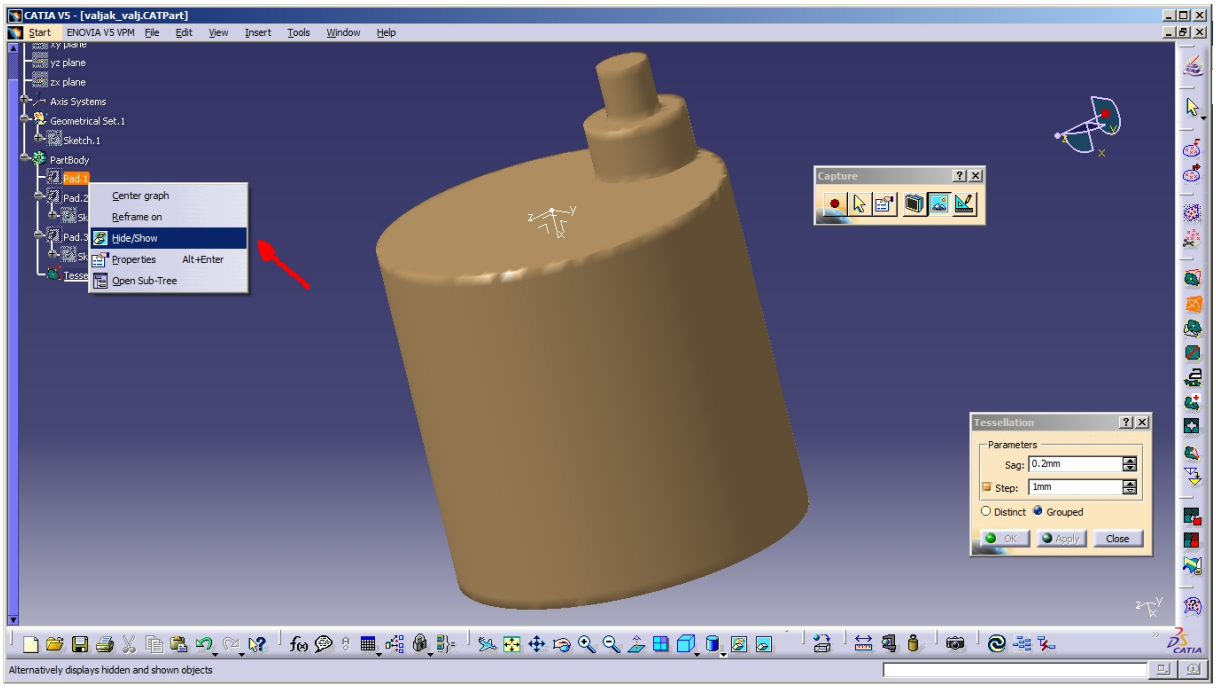
Slika 60. Odabir mreže

Korak 4: Odabrati 'step' (gustoću mreže)



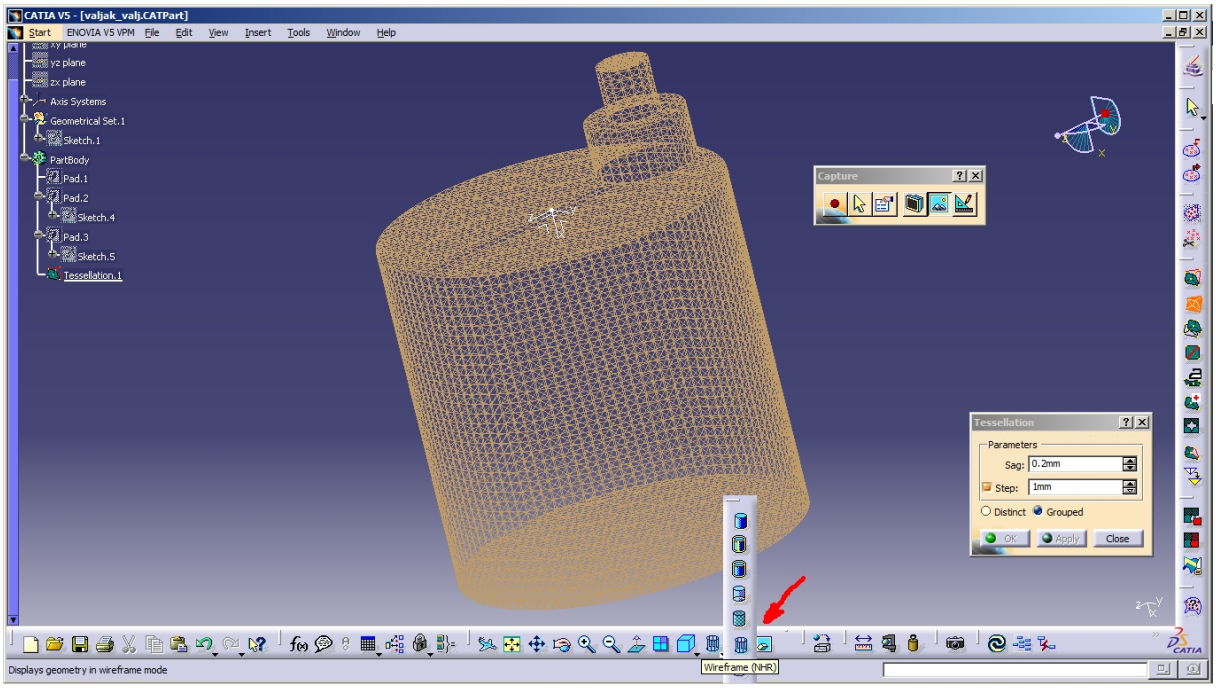
Slika 61. Podešavanje mreže

Korak 4.1: Za bolju preglednost sakriti model (napravljen prije)…



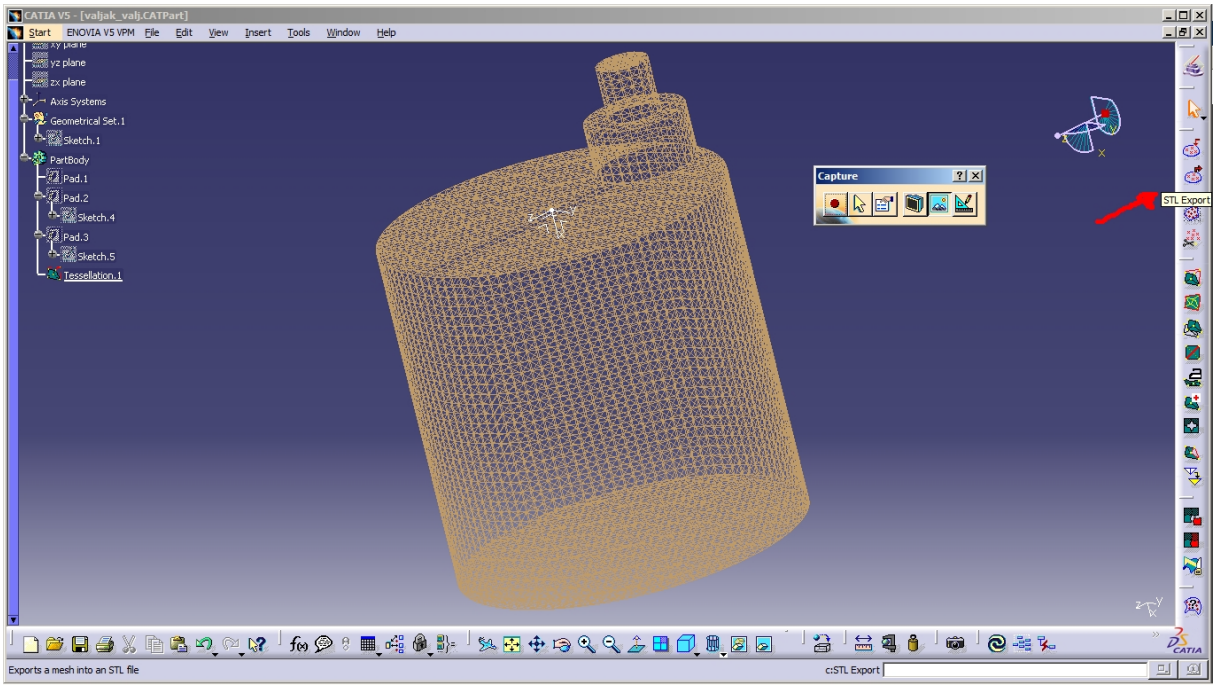
Slika 62. Poboljšavanje preglednosti

Korak 4.2: … i odabrati “Wireframe” pogled.



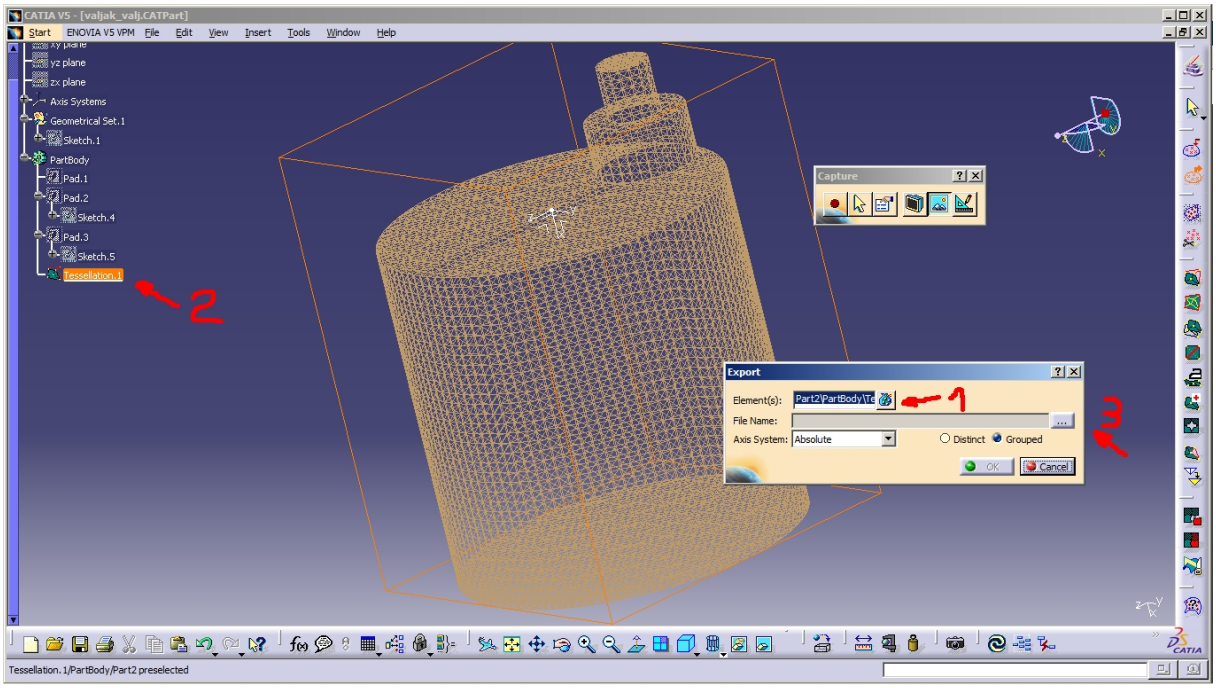
Slika 63. Mijenjanje načina prikaza

Korak 5: Odabrati 'STL Export' ikonu



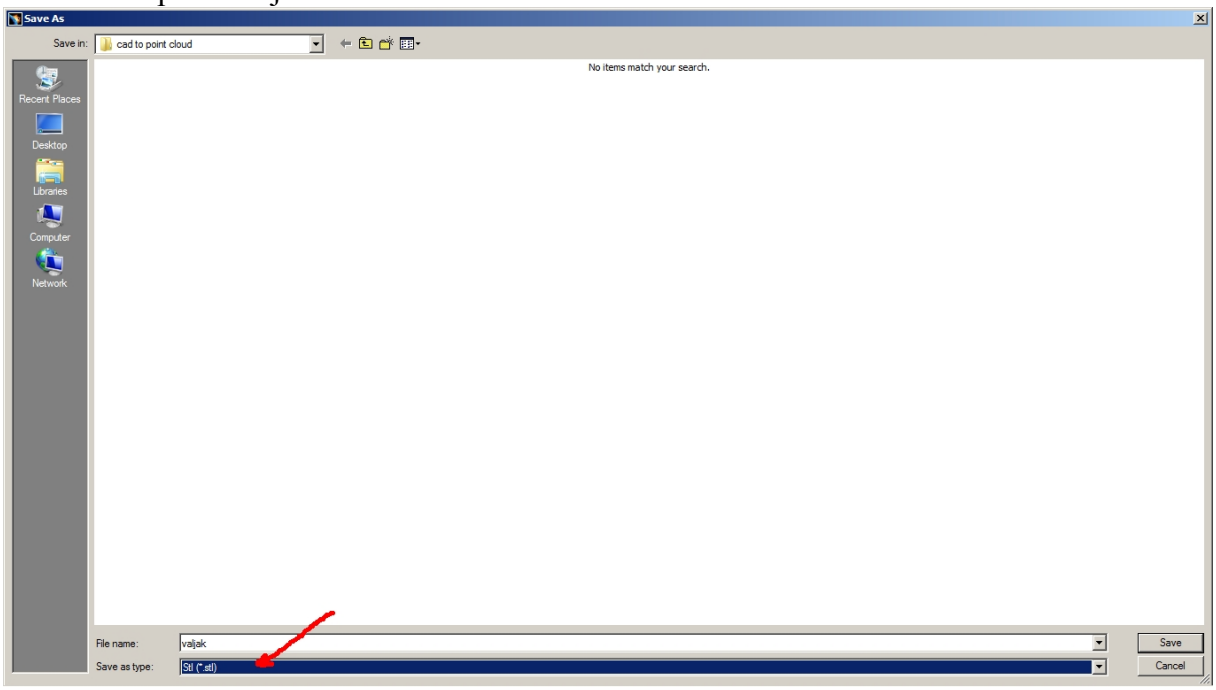
Slika 64. STL Export

Korak 6: Odabrati napravljenu mrežu (engl. tessellation) i zatim odabrati '3 točkice' označene brojem tri



Slika 65. Odabir mreže za spremanje

Korak 7: Upisati željeno ime i odabrati. stl format



Slika 66. Spremanje mreže

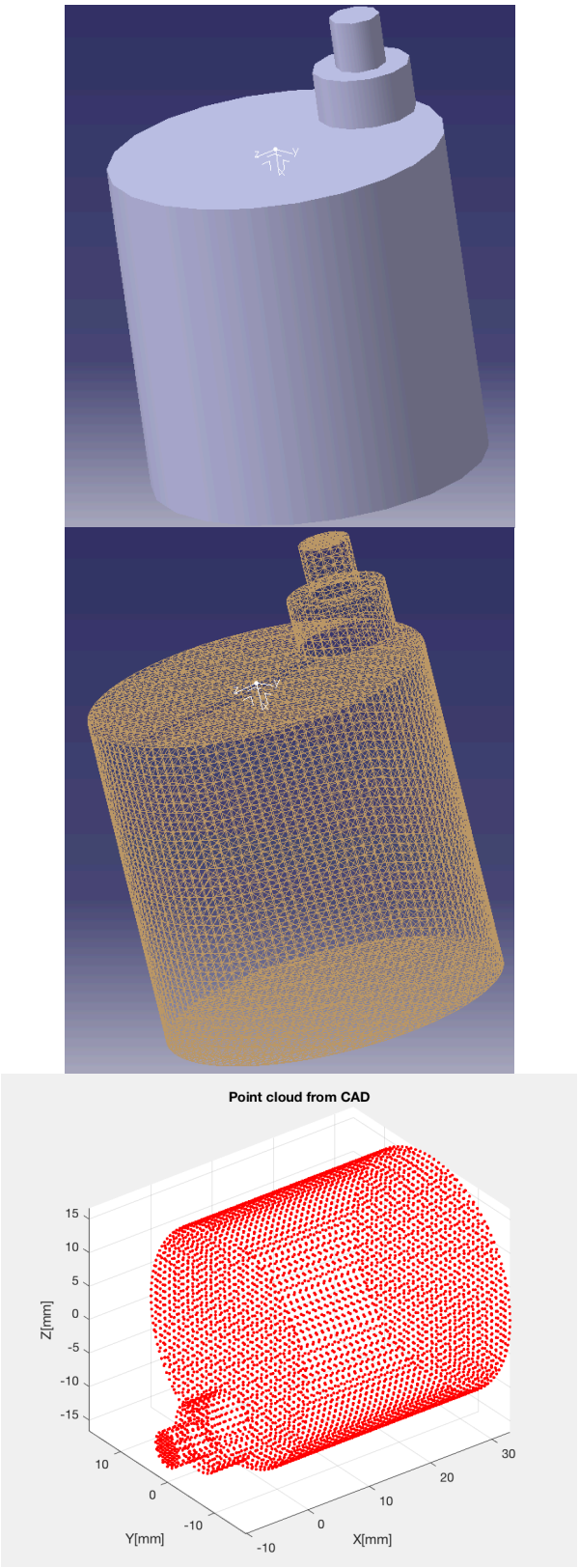
Korak 8: Ubaciti spremljenu .stl datoteku u MATAB

(Orginalno MATLAB nema tu opciju, ali ju je razvio Eric Johnson i dostupna je za besplatno preuzimanje sa stranice 'MathWorks':
https://www.mathworks.com/matlabcentral/fileexchange/22409-stl-file-reader)

'stlread' funkcija iz .stl datoteke dohvaća tri skupa podataka, plohe ('faces'), vrhove ('vertices') i normale ('normals'). Za korištenje MATLAB-ove funkcije 'pointCloud' (za koju je potrebno imati instaliran 'Computer Vision System Toolbox', koji je dostupan od verzije R2015a) trebamo imati 3 koordinate svake točke, tako da dalje koristimo set podatka 'vertices' koji ih sadrži.

```
% STL to MATLAB

[Faces,Vertices,Normals]=stlread('cilindarSTL.stl')
CLOUD=pointCloud (Vertices)



figure ('Name','Point cloud from CAD','NumberTitle','off')
pcshow (CLOUD.Location,'red')
title('Point cloud from CAD');
xlabel ('X[mm]')
ylabel ('Y[mm]')
zlabel ('Z[mm]')
```

Slika 67. Primjer korištenja 'stlread' funkcije

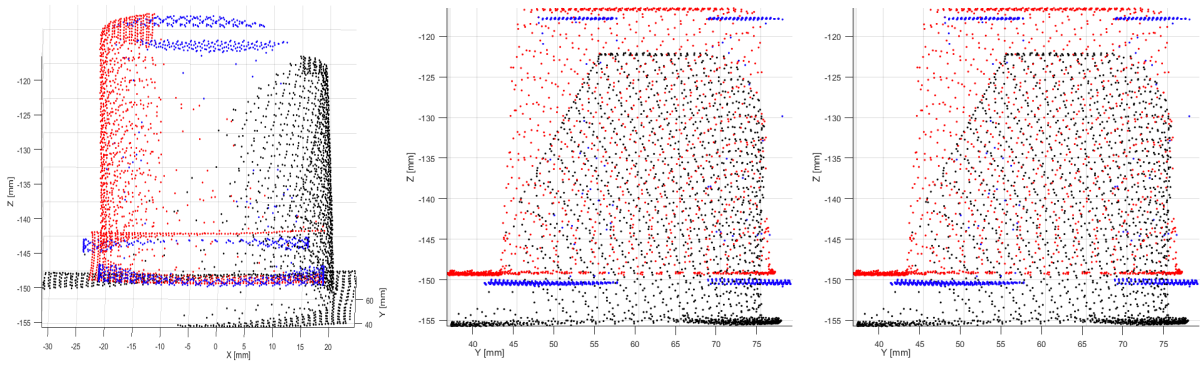


Slika 68. Od modela preko mreže do oblaka točaka objekta

PRILOG II

Postupak pretvorbe oblaka točaka u CAD (MATLAB → CATIA)

Korak 1: Dobiveni oblak točaka (koordinate točaka) iz MATLAB-a spremiti u .xsl datoteku



```
>> xlswrite ('ime_datoteke',matrica_s_koordinatama)
```

Slika 69. Oblaci točaka prikazani u Matlabu i funkcija za spremanje u .xls datoteku

Korak 2: Kopirati koordinate iz .xsl u .asc datoteku

XXls.xls [Compatibility Mode]

	A	B	C	D	E	F	G	H	I
1	-275.55	731.4476	-210.634	-270.278	722.838	84.513	179.891	-1.956	124.671
2	-275.112	730.3125	-186.119	-270.278	722.418	84.513	179.891	-1.956	124.671
3	-275.111	730.1863	-186.079	-270.278	722.293	84.513	179.891	-1.956	124.671
4	-275.113	730.0532	-186.179	-270.278	722.157	84.513	179.891	-1.956	124.671
5	-275.111	729.9307	-186.059	-270.278	722.038	84.513	179.891	-1.956	124.671
6	-275.11	729.8382	-186.009	-270.278	721.947	84.513	179.891	-1.956	124.671
7	-275.109	729.7419	-185.929	-270.278	721.853	84.513	179.891	-1.956	124.671
8	-275.107	729.6619	-185.859	-270.278	721.775	84.513	179.891	-1.956	124.671
9	-275.107	729.5939	-185.859	-270.278	721.707	84.513	179.891	-1.956	124.671
10	-275.107	729.4957	-185.819	-270.278	721.61	84.513	179.891	-1.956	124.671
11	-275.105	729.3917	-185.749	-270.278	721.508	84.513	179.891	-1.956	124.671
12	-275.104	729.2903	-185.669	-270.278	721.409	84.513	179.891	-1.956	124.671
13	-275.106	729.1543	-185.769	-270.278	721.27	84.513	179.891	-1.956	124.671
14	-275.11	728.7617	-185.989	-270.278	720.871	84.513	179.891	-1.956	124.671
15	-275.117	728.0042	-186.419	-270.278	720.101	84.513	179.891	-1.956	124.671
16	-275.154	727.0787	-188.457	-270.278	719.116	84.513	179.891	-1.956	124.671
17	-275.185	725.8847	-190.206	-270.278	717.871	84.513	179.891	-1.956	124.671
18	-275.242	724.52	-193.404	-270.278	716.413	84.513	179.891	-1.956	124.671
19	-275.318	723.0035	-197.672	-270.278	714.772	84.513	179.891	-1.956	124.671
20	-275.414	721.2407	-203.029	-270.278	712.853	84.513	179.891	-1.956	124.671
21	-275.535	719.3205	-209.775	-270.278	710.736	84.513	179.891	-1.956	124.671
22	-275.66	717.2002	-216.791	-270.278	708.411	84.513	179.891	-1.956	124.671
23	-275.76	715.0118	-222.368	-270.278	706.06	84.513	179.891	-1.956	124.671
24	-277.423	696.7959	-315.513	-270.278	685.127	84.513	179.891	-1.956	124.671

Sheet1

asc_datoteka.txt - Notepad

File Edit Format View Help

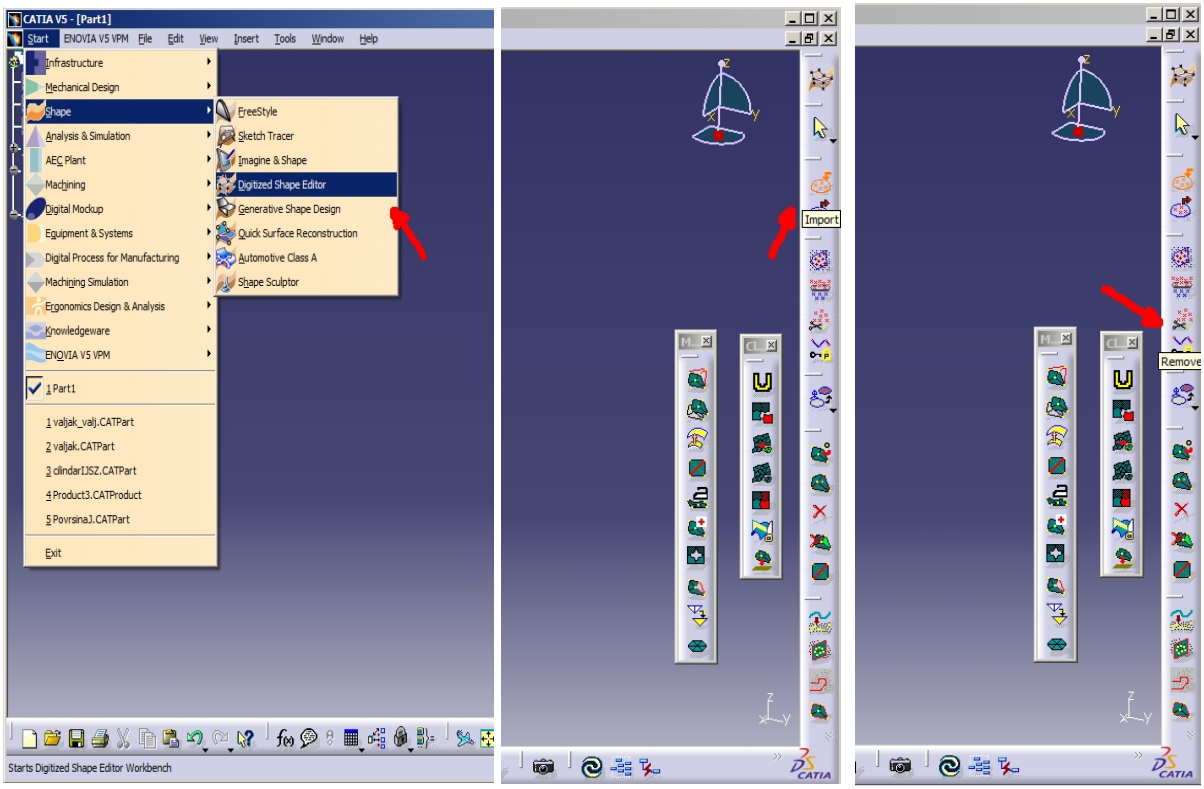
```
   87.8621  -22.7360 -282.7050
   87.8621  -22.7360 -282.7050
   87.8621  -22.7360 -282.7050
   87.8621  -22.7360 -282.7050
   87.8621  -22.7360 -282.7050
   87.8621  -22.7360 -282.7050
   87.8621  -22.7360 -282.7050
   87.8621  -22.7360 -282.7050
   87.8621  -22.7360 -282.7050
   87.8621  -22.7360 -282.7050
   87.8621  -22.7360 -282.7050
   87.8621  -22.7360 -282.7050
   87.8621  -22.7360 -282.7050
   87.8621  -22.7360 -282.7050
   87.8621  -22.7360 -282.7050
   87.8621  -22.7360 -282.7050
   86.3071  -22.7360 -282.6150
   85.4721  -22.7361 -283.2050
   86.2501  -22.7361 -283.2650
   88.6111  -22.7361 -282.9450
   91.0301  -22.7360 -282.2450
   93.4780  -22.7358 -281.2950
   95.8679  -22.7356 -280.1350
   98.3158  -22.7354 -278.8150
  100.7067  -22.7352 -277.3050
  103.1255  -22.7349 -275.5250
  105.6024  -22.7346 -273.5450
  107.9922  -22.7342 -271.5550
  110.4110  -22.7339 -269.2550
  112.8018  -22.7335 -266.7950
  115.2206  -22.7330 -264.1150
  117.6394  -22.7326 -261.4350
  120.0582  -22.7322 -258.7750
  122.5640  -22.7317 -256.0550
  124.9248  -22.7313 -253.4350
  127.3146  -22.7309 -251.0450
  129.7344  -22.7306 -248.7250
  132.1532  -22.7302 -246.5250
  134.6011  -22.7299 -244.4950
  136.9909  -22.7296 -242.6950
  139.4098  -22.7293 -240.9850
  141.8287  -22.7291 -239.3850
  144.3056  -22.7288 -237.9850
  146.6955  -22.7287 -236.8650
  149.0864  -22.7285 -235.9950
  151.5054  -22.7284 -235.4150
  153.9533  -22.7284 -235.2650
```

Slika 70. .xls i .asc datoteka

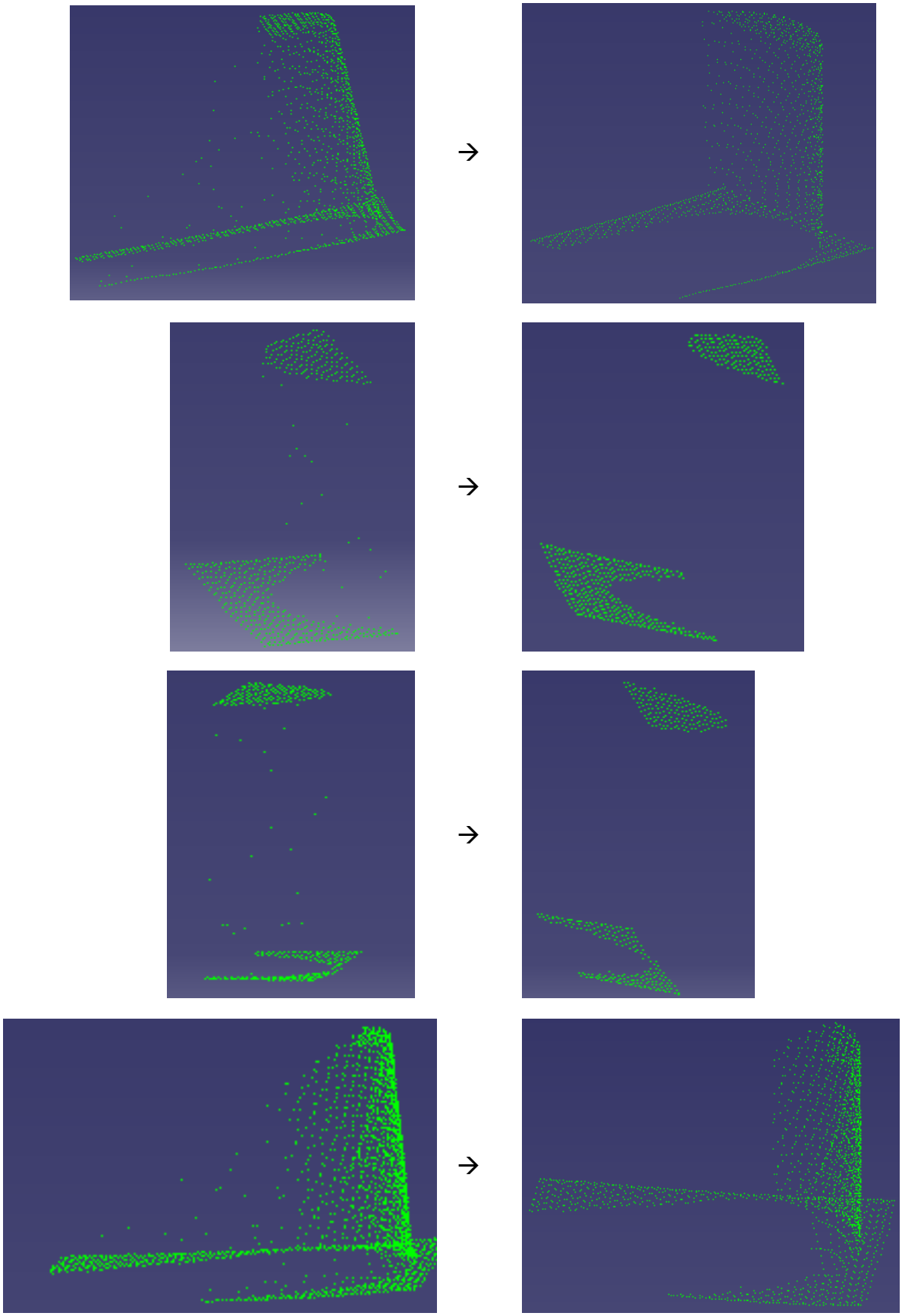
.asc datoteku možemo stvoriti na identičan način kao što smo stvorili .stl datoteku u prethodnom prilogu, samo u koraku 7, datoteku ne spremimo u .stl nego u .asc. (.asc datoteku možemo otvoriti i editirati tako da joj promijenimo nastavak iz asc u txt. Nakon što smo kopirali koordinate datoteku preimenujemo ponovno u izvorno stanje)

Korak 3: Ubacivanje oblaka točaka u CATIA-u i uklanjanje nepotrebnih točaka

Stvorenu .asc datoteku ubacujemo u CATIA-u u 'Digital Shape Editor' sučelju klikom na Import ikonu, a neželjene točke izbacujemo naredbom 'Remove.'

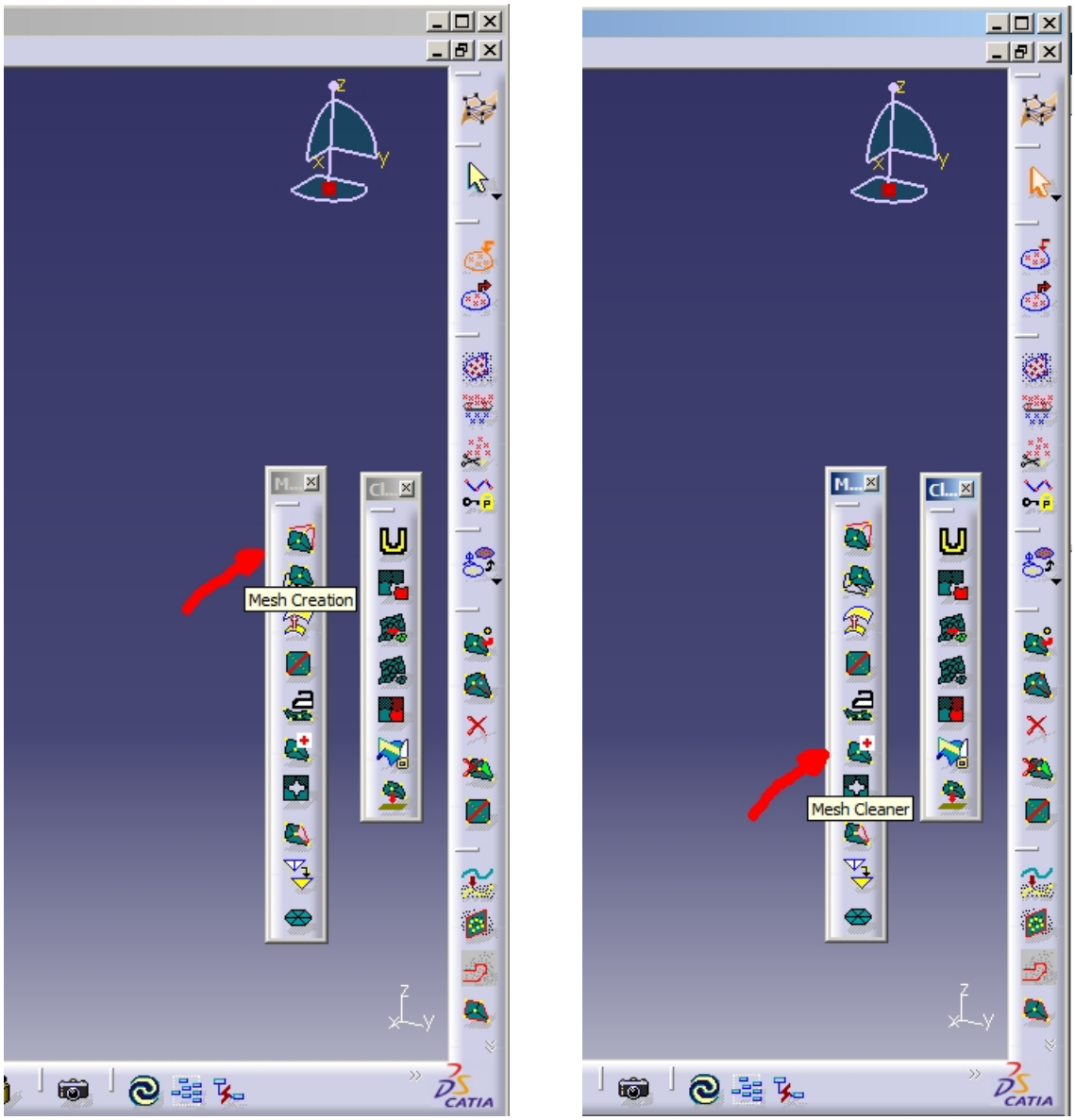


Slika 71. Ubacivanje oblaka točaka u CATIA-u i uklanjanje nepotrebnih točaka

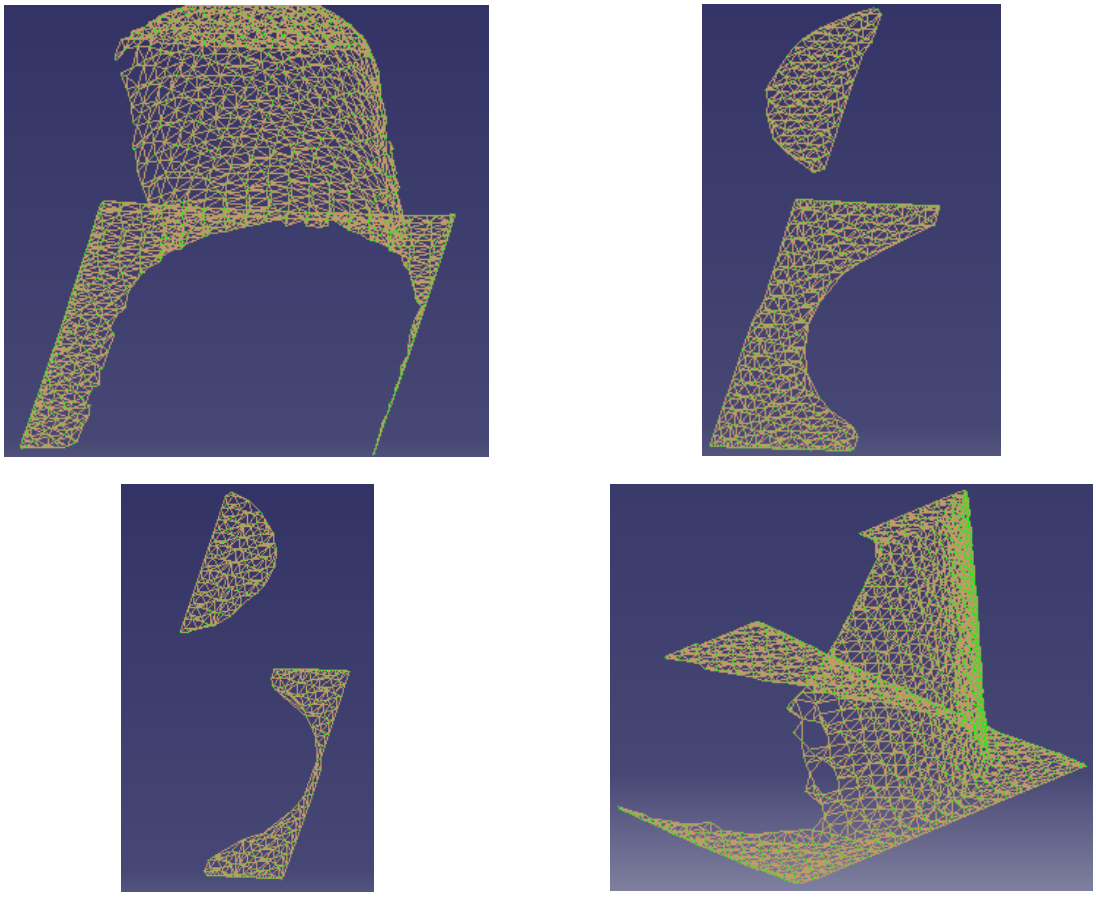


Slika 72. Oblaci točaka prije i nakon uklanjanja neželjenih točaka

Korak 4: Stvaranje mreže naredbom 'Mesh Creation' i potom njeno analiziranje funkcijom 'Mesh Cleaner' u kojoj je ponuđeno više opcija za uklanjanje možebitnih pronađenih grešaka (npr. trokut mreže koji nema spojene stranice).



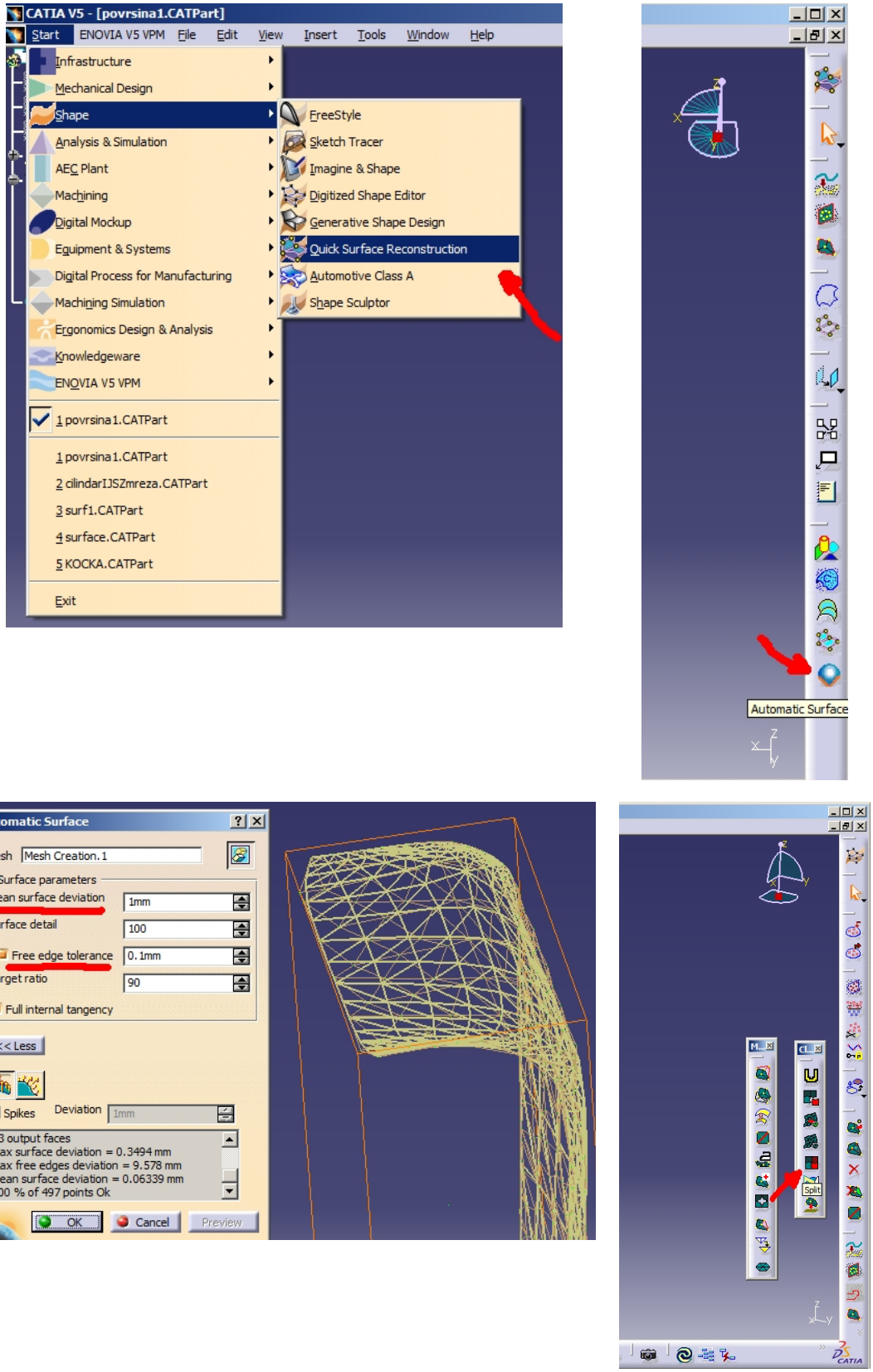
Slika 73. Naredbe za mrežu



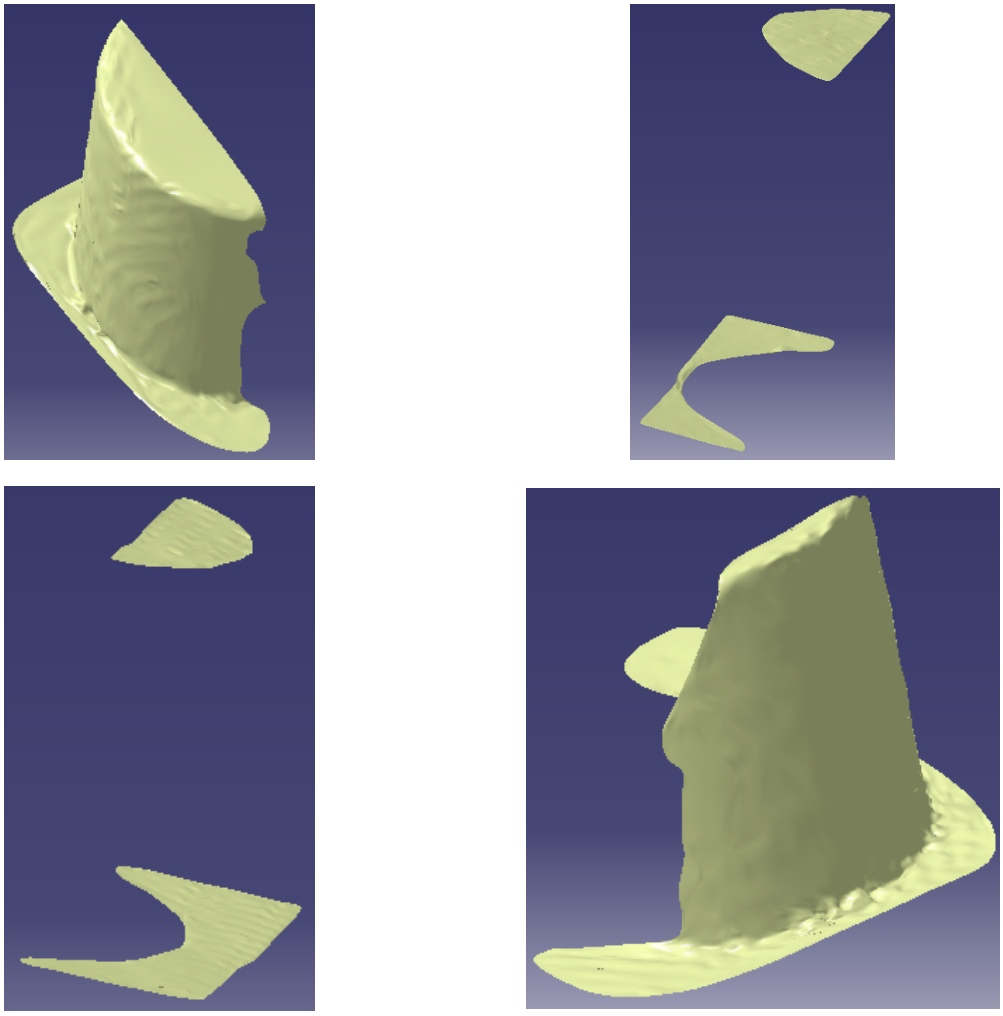
Slika 74. Mreže dobivene iz oblaka točka

Korak 5: Stvaranje površine se odvija u 'Quick Surface Reconstruction' sučelju, naredbu 'Automatic Surface' unutar koje podešavamo parametre s ciljem da površina što bolje prekrije mrežu.

Ako se mreža sastoji od više segmenata (kao npr. druga i treća sa Slika 74) prije stvaranje površine potrebno ih je prije podijeliti na dvije zasebne mreže (u 'Digital Shape Editoru') naredbom 'Split'.

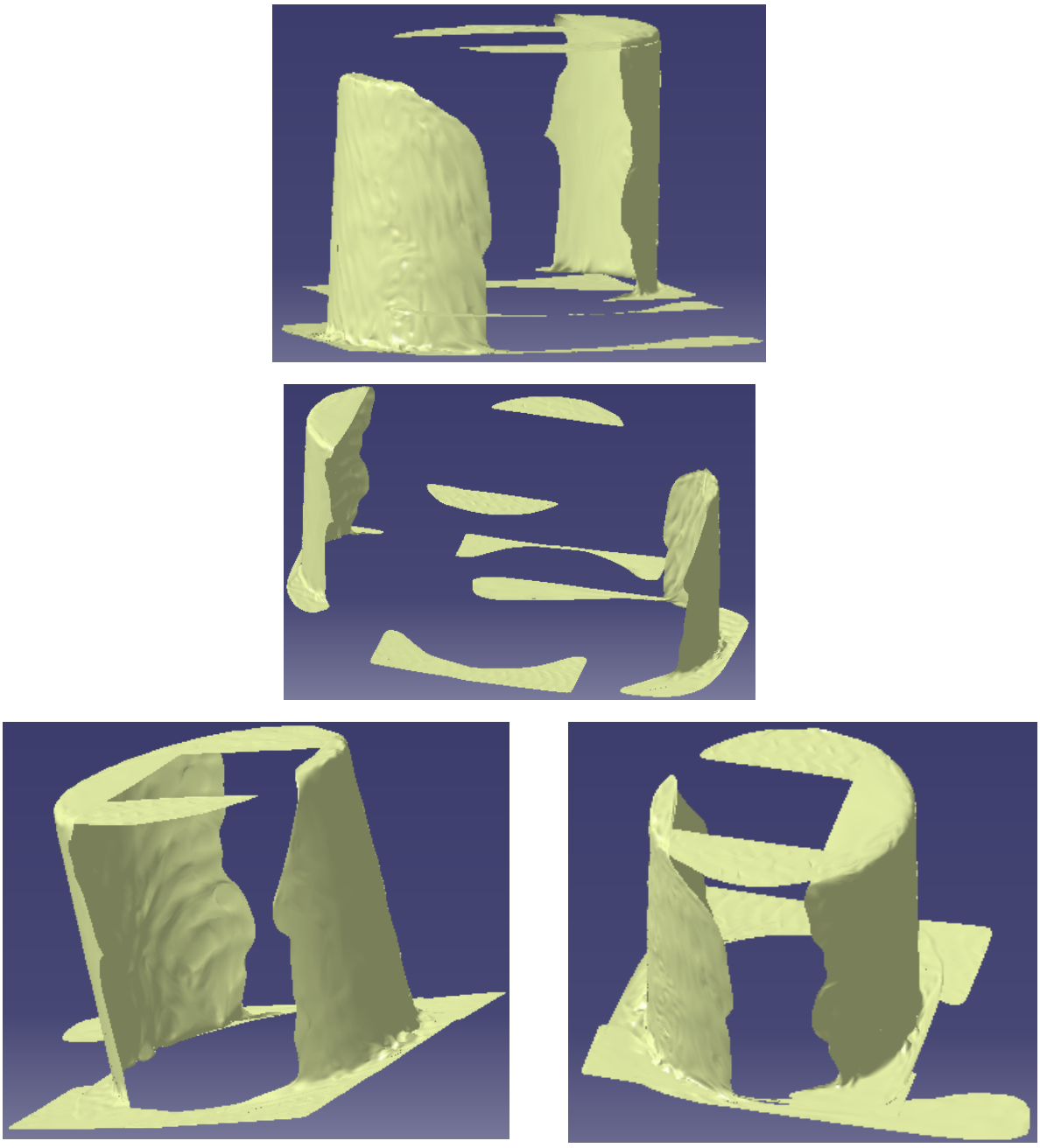


Slika 75. Stvaranje površine iz mreže i naredba za razdvajanje mreže na više dijelova

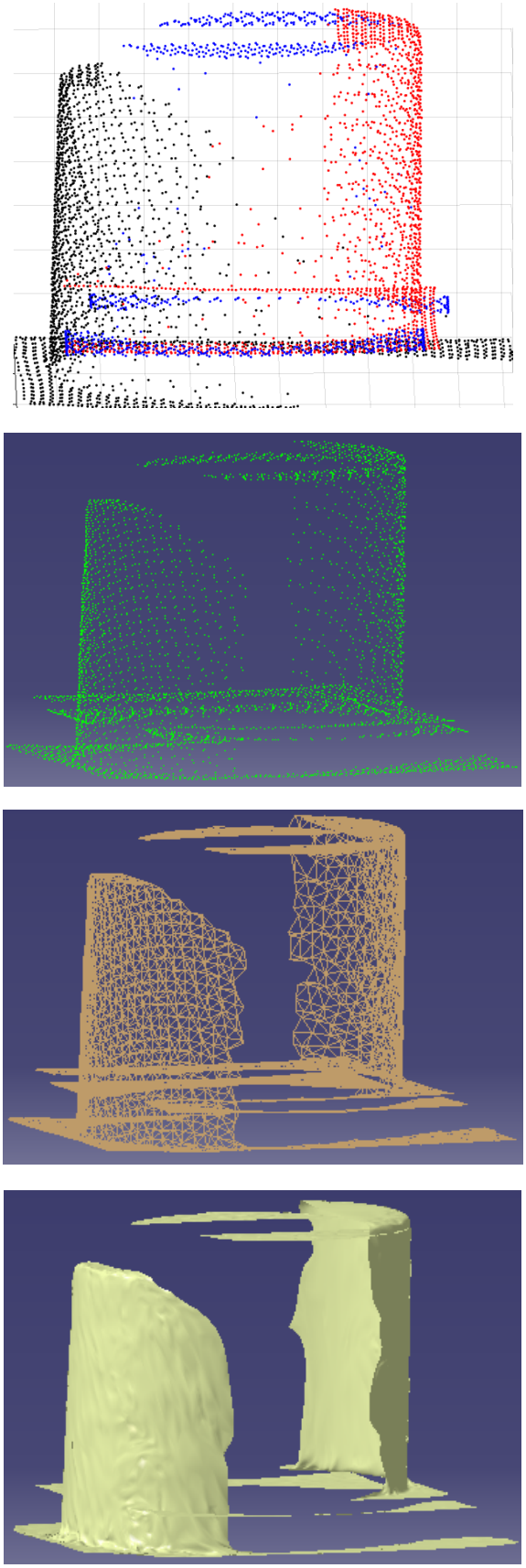


Slika 76. Površine dobivene iz mreža

Korak 6: Dobivene površine translatirati, rotirati i spojiti u cjelinu



Slika 77. Translatiranje, rotiranje i udruživanje oblaka točaka



Slika 78. Od oblaka točaka do modela

PRILOG III

KAREL programski kod

```
PROGRAM svee2
%NOLOCKGROUP
%NOPAUSE=ERROR + COMMAND + TPENABLE

VAR
conf: CONFIG
p_user,p_tool,pp,p: XYZWPR
target,priv,assemble,poda,pol_x,pol_y,pol_z,kut_w,kut_p,kut_r: STRING[20]
decimalni,decimalni1,decimalni2,laser,info,visina: REAL
s:STRING[10]

STATUS, i, j, k, int_value, a, b,c, entry,cijeli,cijeli1,cijeli2,n_bytes,CC,stat: INTEGER
real_flag, UVIJET_1, UVIJET_2, UVIJET,flag: BOOLEAN
indx IN DRAM: INTEGER
TCP1:FILE
TCP:FILE

----------
ROUTINE OPEN_FILE_ (FILE_ : FILE; TAG_ : STRING) FROM LIB_FIL2
ROUTINE CLOSE_FILE_(FILE_ : FILE; TAG_ : STRING) FROM LIB_FIL2
BEGIN

$GROUP[1].$UFRAME = $MNUFRAME[1,2];
$GROUP[1].$UTOOL = $MNUTOOL[1,10]       ;

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$OPER',0,STATUS) ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$STATE',0,STATUS) ; DELAY 20
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$COMMENT','MO',STATUS) ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$PROTOCOL','SM',STATUS) ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$REPERRS','FALSE',STATUS) ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$TIMEOUT',9999,STATUS) ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$PWRD_TIMEOUT',0,STATUS) ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$SERVER_PORT',5556,STATUS) ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$STRT_PATH','192.168.123.51',STA
TUS) ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$STRT_REMOTE','192.168.123.51',
STATUS);
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$PATH','192.168.123.51',STATUS)
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$remote','192.168.123.51',STATUS)

DELAY 10 ; SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$OPER',3,STATUS);
DELAY 10 ;
```

```
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[7].$STATE',3,STATUS) ;



SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$OPER',0,STATUS) ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$STATE',0,STATUS) ; DELAY 20
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$COMMENT','MO',STATUS) ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$PROTOCOL','SM',STATUS) ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$REPERRS','FALSE',STATUS) ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$TIMEOUT',9999,STATUS) ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$PWRD_TIMEOUT',0,STATUS) ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$SERVER_PORT',5555,STATUS) ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$STRT_PATH','192.168.123.50',STA
TUS) ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$STRT_REMOTE','192.168.123.50',
STATUS);
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$PATH','192.168.123.50',STATUS)
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$remote','192.168.123.50',STATUS)

DELAY 10 ; SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$OPER',3,STATUS);
DELAY 10 ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$STATE',3,STATUS) ;

DELAY 10 ;
reconn_:: --
DELAY 100; CLOSE_FILE_(TCP1,'S7:') ; CLOSE_FILE_(TCP,'S8:') ; DELAY 100;
OPEN_FILE_(TCP1,'S7:'); OPEN_FILE_(TCP,'S8:');



SET_REAL_REG(11,0,STATUS)
SET_REAL_REG(15,0,STATUS)
SET_REAL_REG(14,0,STATUS)
SET_REAL_REG(13,0,STATUS)
SET_REAL_REG(4,0,STATUS)
SET_INT_REG(23,0,STATUS)

DELAY 2000

c=0


WHILE 1=1 DO
        GET_REG(6,flag,cijeli,decimalni,STATUS)
        DELAY cijeli
                CC=0

        WRITE TCP('PsingleK',CR)
        DELAY 85
```

```
p=CURPOS(0,0,1)
WHILE CC=0 DO

BYTES_AHEAD (TCP, n_bytes, STATUS)

IF n_bytes = 0 THEN
        DELAY 1
        WRITE ('NO:',CR);
        SET_INT_REG(2,99,STATUS)
ENDIF
--



        IF n_bytes>0 THEN

        READ TCP (poda::1)
        assemble = ''
        WHILE poda <> ';' DO
                assemble = assemble + poda
                READ TCP (poda::1)
        ENDWHILE
        CNV_STR_REAL(assemble,laser)
        IF c<12 THEN
                visina=laser-2
                c=13
                ENDIF

        SET_REAL_REG(3,laser,STATUS)
        IF laser<visina THEN
                pp=CURPOS(0,0,1)
                SET_REAL_REG(4,1,STATUS)
                SET_POS_REG(59, pp, STATUS)
        ENDIF

        CNV_REAL_STR(p.x, 7, 3, pol_x)
        CNV_REAL_STR(p.y, 7, 3, pol_y)
        CNV_REAL_STR(p.z, 7, 3, pol_z)
        CNV_REAL_STR(p.w, 7, 3, kut_w)
        CNV_REAL_STR(p.p, 7, 3, kut_p)
        CNV_REAL_STR(p.r, 7, 3, kut_r)
        WRITE
TCP1('P',pol_x,'@',pol_x,'@',pol_y,'@',pol_z,'@',kut_w,'@',kut_p,'@',kut_r,'@',assem
ble,'@',CR);

        CC=1
```

```
            ENDIF


        ENDWHILE



IF TPIN[132] THEN;GOTO van_;ENDIF -- F3
IF TPIN[133] THEN;GOTO nova_;ENDIF -- F4
ENDWHILE

nova_::
SET_REAL_REG(23,1,STATUS)
        DELAY 100
        SET_INT_REG(23,99,STATUS)
WRITE TCP1('prom',CR);
WHILE 1=1 DO


        BYTES_AHEAD (TCP1, n_bytes, STATUS)

        IF n_bytes = 0 THEN
                DELAY 1
                WRITE ('NO:',CR);
                SET_INT_REG(2,99,STATUS)
        ENDIF
        --



                IF n_bytes>0 THEN
                READ TCP1 (poda::1)
                assemble = ''
                WHILE poda <> ';' DO
                        assemble = assemble + poda
                        READ TCP1 (poda::1) --X
                ENDWHILE
                CNV_STR_REAL(assemble,info)
                SET_REAL_REG(13,info,STATUS)
                READ TCP1 (poda::1)
                assemble = ''
                WHILE poda <> ';' DO
                        assemble = assemble + poda
                        READ TCP1 (poda::1) --Y
                ENDWHILE
                CNV_STR_REAL(assemble,info)
                SET_REAL_REG(14,info,STATUS)
                READ TCP1 (poda::1)
                assemble = ''
```

```
WHILE poda <> ';' DO
        assemble = assemble + poda
        READ TCP1 (poda::1) --Z
ENDWHILE
CNV_STR_REAL(assemble,info)
SET_REAL_REG(15,info,STATUS)
READ TCP1 (poda::1)
assemble = ''
WHILE poda <> ';' DO
        assemble = assemble + poda
        READ TCP1 (poda::1) --w
ENDWHILE
CNV_STR_REAL(assemble,info)
SET_REAL_REG(20,info,STATUS)
READ TCP1 (poda::1)
assemble = ''
WHILE poda <> ';' DO
        assemble = assemble + poda
        READ TCP1 (poda::1) --p
ENDWHILE
CNV_STR_REAL(assemble,info)
SET_REAL_REG(21,info,STATUS)
READ TCP1 (poda::1)
assemble = ''
WHILE poda <> ';' DO
        assemble = assemble + poda
        READ TCP1 (poda::1) --r
ENDWHILE
CNV_STR_REAL(assemble,info)
SET_REAL_REG(22,info,STATUS)



READ TCP1 (poda::1)
assemble = ''
WHILE poda <> ';' DO
        assemble = assemble + poda
        READ TCP1 (poda::1) --X
ENDWHILE
CNV_STR_REAL(assemble,info)
SET_REAL_REG(30,info,STATUS)
READ TCP1 (poda::1)
assemble = ''
WHILE poda <> ';' DO
        assemble = assemble + poda
        READ TCP1 (poda::1) --Y
ENDWHILE
CNV_STR_REAL(assemble,info)
SET_REAL_REG(31,info,STATUS)
READ TCP1 (poda::1)
```

```
assemble = ''
WHILE poda <> ';' DO
        assemble = assemble + poda
        READ TCP1 (poda::1) --Z
ENDWHILE
CNV_STR_REAL(assemble,info)
SET_REAL_REG(32,info,STATUS)
READ TCP1 (poda::1)
assemble = ''
WHILE poda <> ';' DO
        assemble = assemble + poda
        READ TCP1 (poda::1) --w
ENDWHILE
CNV_STR_REAL(assemble,info)
SET_REAL_REG(33,info,STATUS)
READ TCP1 (poda::1)
assemble = ''
WHILE poda <> ';' DO
        assemble = assemble + poda
        READ TCP1 (poda::1) --p
ENDWHILE
CNV_STR_REAL(assemble,info)
SET_REAL_REG(34,info,STATUS)
READ TCP1 (poda::1)
assemble = ''
WHILE poda <> ';' DO
        assemble = assemble + poda
        READ TCP1 (poda::1) --r
ENDWHILE
CNV_STR_REAL(assemble,info)
SET_REAL_REG(35,info,STATUS)


READ TCP1 (poda::1)
assemble = ''
WHILE poda <> ';' DO
        assemble = assemble + poda
        READ TCP1 (poda::1) --X
ENDWHILE
CNV_STR_REAL(assemble,info)
SET_REAL_REG(36,info,STATUS)
READ TCP1 (poda::1)
assemble = ''
WHILE poda <> ';' DO
        assemble = assemble + poda
        READ TCP1 (poda::1) --Y
ENDWHILE
CNV_STR_REAL(assemble,info)
SET_REAL_REG(37,info,STATUS)
READ TCP1 (poda::1)
```

```
                assemble = ''
                WHILE poda <> ';' DO
                        assemble = assemble + poda
                        READ TCP1 (poda::1) --Z
                ENDWHILE
                CNV_STR_REAL(assemble,info)
                SET_REAL_REG(38,info,STATUS)
                READ TCP1 (poda::1)
                assemble = ''
                WHILE poda <> ';' DO
                        assemble = assemble + poda
                        READ TCP1 (poda::1) --w
                ENDWHILE
                CNV_STR_REAL(assemble,info)
                SET_REAL_REG(39,info,STATUS)
                READ TCP1 (poda::1)
                assemble = ''
                WHILE poda <> ';' DO
                        assemble = assemble + poda
                        READ TCP1 (poda::1) --p
                ENDWHILE
                CNV_STR_REAL(assemble,info)
                SET_REAL_REG(40,info,STATUS)
                READ TCP1 (poda::1)
                assemble = ''
                WHILE poda <> ';' DO
                        assemble = assemble + poda
                        READ TCP1 (poda::1) --r
                ENDWHILE
                CNV_STR_REAL(assemble,info)
                SET_REAL_REG(41,info,STATUS)
                SET_REAL_REG(11,111.111,STATUS)

                ENDIF
ENDWHILE

van_::
DELAY 100
WRITE TCP1('kraj',CR);
DELAY 100
DELAY 100; CLOSE_FILE_(TCP1,'S7:') ; CLOSE_FILE_(TCP,'S8:') ;DELAY 100;
END svee2
```

PRILOG IV

MATLAB programski kod

```
clear
clc
close all
global Y_TCP
global X_TCP
global Z_TCP
global W_TCP
global P_TCP
global R_TCP
global Y_TCP2
global X_TCP2
global Z_TCP2
global W_TCP2
global P_TCP2
global R_TCP2
global Y_TCP3
global X_TCP3
global Z_TCP3
global W_TCP3
global P_TCP3
global R_TCP3
global message_c
global message_p
global in
while true
message_c='';
message_p='';
ip='192.168.123.25';port=5556;
X_client_open_rw(ip,port,100000);
fprintf('Veza uspostavljena\n')
% komunikacija glavna petlja

fprintf('ULAZAK U PETLJU\n');
j=0;
figure(1)
i=0;
while true
kraj=0;
X_client_rcv_rw();

if size(message_c,2)==5
if strcmp(message_c(1:4),'kraj')
```

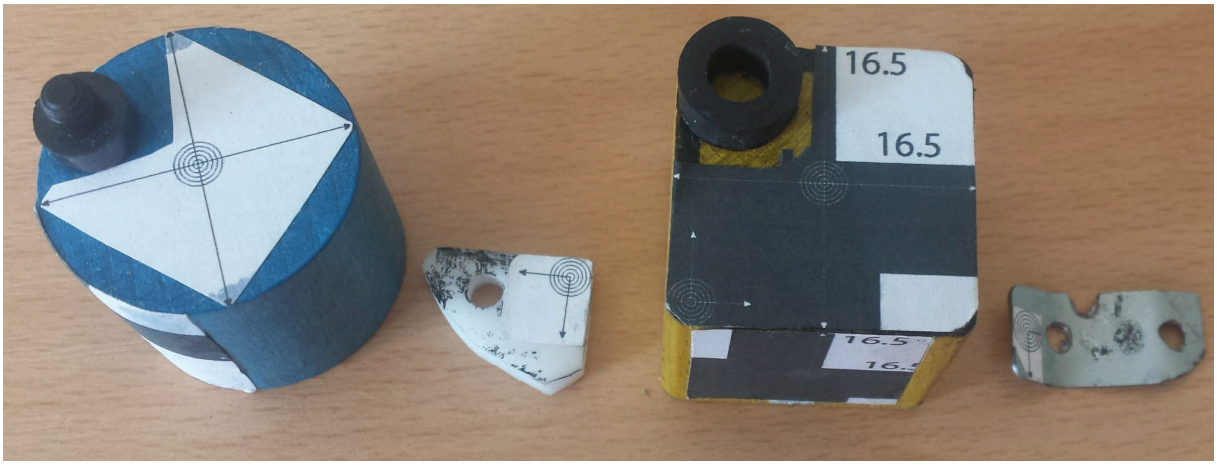
```
%  message_c
X_client_close_rw();
pause (0.1)
kraj=1; break
end
end

if size(message_c,2)==5
if strcmp(message_c(1:4),'prom')
%  message_c
kraj=1; break
end
end


if isempty(message_c);a=1;
s='';
else
poruka=message_c;
i=i+1;
if size(poruka,2)<100;
broj_str = strrep (poruka, ',' , '.') ;
koordinata=str2num(broj_str);
disp(koordinata);
message_p;
in;
X=in(2);
Y=in(3)
Z=in(4);
W=in(5);
P=in(6);
R=in(7)
laser=in(8)-10

if (laser~=0 && laser~=9999 && laser~=9998 && laser~=9997 &&
laser~=9996 && laser~=9989)
    brojac=brojac+1;
Rx=[1 0 0 ;0 cosd(W) -sind(W) ;0 sind(W) cosd(W) ];

Ry=[cosd(P) 0 sind(P)  ;0 1 0 ; -sind(P) 0 cosd(P) ];

Rz=[cosd(R) -sind(R) 0 ;sind(R) cosd(R) 0 ;0 0 1 ];



RR=Rz*Ry*Rx;

RR(:,4)=[X Y Z];
RR(4,:)=[0 0 0 1];
```

```
las=[0 0 laser 1]'
tocka=RR*las;
tockaK(1,1)=tocka(1,1);
tockaK(2,1)=tocka(2,1);
tockaK(3,1)=tocka(3,1);

koordinate(brojac,1)=tockaK(1,1);
koordinate(brojac,2)=tockaK(2,1);
koordinate(brojac,3)=tockaK(3,1);
koordinate(brojac,4)=X;
koordinate(brojac,5)=Y;
koordinate(brojac,6)=Z;
koordinate(brojac,7)=W;
koordinate(brojac,8)=P;
koordinate(brojac,9)=R;

tockaK
hold on
% view([tockaK(1),tockaK(2),tockaK(3)]);
view(3)
plot3 (tockaK(1),tockaK(2),tockaK(3),'r.')
laser=0;
end




end
log2(i)=size(poruka,2);
end
pause(0.01);
end

koordinateR=koordinate;

% match3(koordinate) %za cilindar
% veliki(koordinate) %Za pleksi
% kockaciljnik(koordinate) %Za kocku

lim(koordinate)

X_client_send_rw(X_TCP)
X_client_send_rw(';')
X_client_send_rw(Y_TCP)
X_client_send_rw(';')
X_client_send_rw(Z_TCP)
X_client_send_rw(';')

X_client_send_rw(W_TCP)
```

```
X_client_send_rw(';')
X_client_send_rw(P_TCP)
X_client_send_rw(';')
X_client_send_rw(R_TCP)
X_client_send_rw(';')


X_client_send_rw(X_TCP2)
X_client_send_rw(';')
X_client_send_rw(Y_TCP2)
X_client_send_rw(';')
X_client_send_rw(Z_TCP2)
X_client_send_rw(';')

X_client_send_rw(W_TCP2)
X_client_send_rw(';')
X_client_send_rw(P_TCP2)
X_client_send_rw(';')
X_client_send_rw(R_TCP2)
X_client_send_rw(';')

X_client_send_rw(X_TCP3)
X_client_send_rw(';')
X_client_send_rw(Y_TCP3)
X_client_send_rw(';')
X_client_send_rw(Z_TCP3)
X_client_send_rw(';')

X_client_send_rw(W_TCP3)
X_client_send_rw(';')
X_client_send_rw(P_TCP3)
X_client_send_rw(';')
X_client_send_rw(R_TCP3)
X_client_send_rw(';')

break
% X_client_close_rw();
% pause (0.1)

end
```

PRILOG V

MATLAB funkcija za lim



Slika 79. Elementi za koje je napravljen program za traženje značajki

(Funkcije 'match3', 'veliki', 'kockaciljnik' za cilindar, komadić pleksiglasa, odnosno kocku rade na isti način kao i u nastavku pokazana funkcija 'lim'.)

```
function lim (koordinate)
clc
close all
global Y_TCP
global X_TCP
global Z_TCP
global R_TCP
global W_TCP
global P_TCP

global Y_TCP2
global X_TCP2
global Z_TCP2
global R_TCP2
global W_TCP2
global P_TCP2

global Y_TCP3
global X_TCP3
global Z_TCP3
global R_TCP3
global W_TCP3
global P_TCP3
```

```
komplet2=koordinate
k2=koordinate(:,[1 2 3])
krnje2=pointCloud (k2) %novi, koji će mirovat

load('lim01.mat')
komplet3=koordinate
k3=koordinate(:,[1 2 3]) %zadani njega pomicemo, neočišćeni
krnje3=pointCloud (k3)

oguljen = pcread('ciscenje.ply') %očišćeni

figure('Name','Scanovi','NumberTitle','off');
subplot(2,2,2)
pcshow (krnje2.Location,'black')
xlabel('X [mm]');
ylabel('Y [mm]');
zlabel('Z [mm]');
title('Random');
subplot(2,2,1)
pcshow (krnje3.Location,'black')
xlabel('X [mm]');
ylabel('Y [mm]');
zlabel('Z [mm]');
title('Etalonac');
subplot(2,2,3)
pcshow (oguljen.Location,'black')
title('ocisceni');
xlabel('X [mm]');
ylabel('Y [mm]');
zlabel('Z [mm]');
;

% TOCKA1 OD INTERESA NA PRVI OBLAK
figure('Name','tockica1 na prvotnom','NumberTitle','off');
pcshow (oguljen.Location,'black')
hold on
[indices,dists] = findNeighborsInRadius(oguljen,[-95 598 -
285],10)
[M,I] = min(dists(:))
II=indices(I)
tocka = select(oguljen,II)
pcshow (tocka.Location,'green','MarkerSize',20) % tocka na
prvom
xlabel('X [mm]');
ylabel('Y [mm]');
zlabel('Z [mm]');
hold off


% TOCKA2 OD INTERESA NA PRVotni OBLAK, etalonac
```

```
figure('Name','tockica2 na prvotnom','NumberTitle','off');
pcshow (oguljen.Location,'black')
hold on
[indices,dists] = findNeighborsInRadius(oguljen,[-109 606 -
285],10)
[M,I] = min(dists(:))
II=indices(I)
tocka2 = select(oguljen,II)
pcshow (tocka2.Location,'yel','MarkerSize',20) % tocka na
prvom
xlabel('X [mm]');
ylabel('Y [mm]');
zlabel('Z [mm]');
hold off


% TOCKA2 OD INTERESA NA PRVI OBLAK
figure('Name','tockica3 na prvotnom','NumberTitle','off');
pcshow (oguljen.Location,'black')
hold on
[indices,dists] = findNeighborsInRadius(oguljen,[-103 595 -
285],10)
[M,I] = min(dists(:))
II=indices(I)
tocka3 = select(oguljen,II)
pcshow (tocka3.Location,'blu','MarkerSize',20) % tocka na
prvom
xlabel('X [mm]');
ylabel('Y [mm]');
zlabel('Z [mm]');
hold off

% TOCKA2 OD INTERESA NA PRVI OBLAK
figure('Name','sve tri na prvotnom','NumberTitle','off');
pcshow (oguljen.Location,'black')
hold on
[indices,dists] = findNeighborsInRadius(oguljen,[-103 595 -
285],10)
[M,I] = min(dists(:))
II=indices(I)
tocka3 = select(oguljen,II)
pcshow (tocka.Location,'gre','MarkerSize',20) % tocka na
prvotnom
pcshow (tocka2.Location,'yel','MarkerSize',20) % tocka na
prvotnom
pcshow (tocka3.Location,'blu','MarkerSize',20) % tocka na
prvotnom

xlabel('X [mm]');
ylabel('Y [mm]');
```

```
zlabel('Z [mm]');
hold off

A = [cosd(0) sind(0) 0 0; ...
    -sind(0) cosd(0) 0 0; ...
      0         0  1 0; ...
      0         0 0 1];
tform1 = affine3d(A);

rmse=10
rmse1=11;
a=0
while (rmse>0)
    if a<359
    YY = [cosd(0+a) 0 sind(0+a) 0; ...
       0      1   0     0; ...
      -sind(0+a) 0  cosd(0+a) 0; ...
      0        0    0    1];
tform1 = affine3d(YY);
[tformX,cat_pom,rmse] =
pcregrigid(oguljen,krnje2,'MaxIterations',300,'InitialTransfor
m',tform1,'Extrapolate',true)
    end
    if a<719 && a>359
    XX = [1   0       0     0; ...
      0 cosd(0+a) -sind(0+a) 0; ...
      0 sind(0+a)  cosd(0+a)  0; ...
      0   0       0    1];
tform1 = affine3d(XX);
[tformX,cat_pom,rmse] =
pcregrigid(oguljen,krnje2,'MaxIterations',300,'InitialTransfor
m',tform1,'Extrapolate',true)
    end
    if a>720   && a<1080
    ZZ = [cosd(0+a) -sind(0+a) 0 0; ...
            sind(0+a) cosd(0+a) 0 0; ...
             0         0  1 0; ...
             0         0 0 1];

tform1 = affine3d(ZZ);
[tformX,cat_pom,rmse] =
pcregrigid(oguljen,krnje2,'MaxIterations',300,'InitialTransfor
m',tform1,'Extrapolate',true)
    end
a=a+15
if (rmse<rmse1)
  rmse1=rmse ;
  cat_pom1=cat_pom;
  tformXX=tformX;
```

```
end
if a>1081
    break
end


end

figure ('Name','pomaknut etalonac oguljeni na random scan' )
pcshow (cat_pom1.Location,'red')
hold on
pcshow (krnje2.Location,'blue')
hold off
xlabel('X [mm]');
ylabel('Y [mm]');
zlabel('Z [mm]');
rmse


ptCloudTformed = pctransform(krnje3,tformXX);
figure ('Name','Preklopljeni neoguljeni etalonac i random')
pcshow (ptCloudTformed.Location,'red')
hold on
pcshow (krnje2.Location,'black')

tockaT = pctransform(tocka,tformXX);
tockaT2 = pctransform(tocka2,tformXX);
tockaT3 = pctransform(tocka3,tformXX);
figure ('Name','tocka i random i njegova tocka')
pcshow (tockaT.Location,'green','MarkerSize',20)%tocka od
prvotnog zarotirana na random
hold on
pcshow (tockaT2.Location,'yel','MarkerSize',20)
hold on
pcshow (tockaT3.Location,'blu','MarkerSize',20)
hold on
pcshow (krnje2.Location,'black')


[indices1,dists1] =
findNeighborsInRadius(krnje2,tockaT.Location,10)
[M,I] = min(dists1(:))
II=indices1(I)
tocka_mirujucem1 = select(krnje2,II)
pcshow (tocka_mirujucem1.Location,'red','MarkerSize',20) %
tocka na mirujucem najbliza zarotiranoj tocki s prvotnog
(random oblak)
hold on
```

```
[indices2,dists2] =
findNeighborsInRadius(krnje2,tockaT2.Location,10)
[M,I] = min(dists2(:))
II=indices2(I)
tocka_mirujucem2 = select(krnje2,II)
pcshow (tocka_mirujucem2.Location,'red','MarkerSize',20) %
tocka na mirujucem najbliza zarotiranoj tocki s prvotnog
(random oblak)
hold on

[indices3,dists3] =
findNeighborsInRadius(krnje2,tockaT3.Location,10)
[M,I] = min(dists3(:))
II=indices3(I)
tocka_mirujucem3 = select(krnje2,II)
pcshow (tocka_mirujucem3.Location,'red','MarkerSize',20) %
tocka na mirujucem najbliza zarotiranoj tocki s prvotnog
(random oblak)
xlabel('X [mm]');
ylabel('Y [mm]');
zlabel('Z [mm]');
hold off

tocka_mir=tocka_mirujucem1.Location*10000;
k2x=k2*10000;
[tf, index]=ismember(k2x,tocka_mir,'rows');
i=1;
while (tf(i)==false)
i=i+1;
end
komplet2(i,:)
X_TCP=num2str(komplet2(i,4))
Y_TCP=num2str(komplet2(i,5))
Z_TCP=num2str(komplet2(i,6))
W_TCP=num2str(komplet2(i,7))
P_TCP=num2str(komplet2(i,8))
R_TCP=num2str(komplet2(i,9))

tocka_mir=tocka_mirujucem2.Location*10000;
k2x=k2*10000;
[tf, index]=ismember(k2x,tocka_mir,'rows');
i=1;
while (tf(i)==false)
i=i+1;
end
komplet2(i,:)
X_TCP2=num2str(komplet2(i,4))
Y_TCP2=num2str(komplet2(i,5))
Z_TCP2=num2str(komplet2(i,6))
W_TCP2=num2str(komplet2(i,7))
```

```
P_TCP2=num2str(komplet2(i,8))
R_TCP2=num2str(komplet2(i,9))


tocka_mir=tocka_mirujucem3.Location*10000;
k2x=k2*10000;
[tf, index]=ismember(k2x,tocka_mir,'rows');
i=1;
while (tf(i)==false)
i=i+1;
end
komplet2(i,:)
X_TCP3=num2str(komplet2(i,4))
Y_TCP3=num2str(komplet2(i,5))
Z_TCP3=num2str(komplet2(i,6))
W_TCP3=num2str(komplet2(i,7))
P_TCP3=num2str(komplet2(i,8))
R_TCP3=num2str(komplet2(i,9))
```

PRILOG VI

Teach Pendant programski kod

```
  1:   ;
  2:  UTOOL_NUM=10 ;
  3:  R[7]=0    ;
  4:  R[8]=0    ;
  5:  R[16]=0    ;
  6:  R[17]=0    ;
  7:  R[18]=0    ;
  8:  R[19]=0    ;
  9:  R[23]=0    ;
 10:  R[11]=0    ;
 11:   ;
 12:   ;
 13:   ;
 14:  PR[10]=PR[60]    ;
 15:L PR[10] 100mm/sec FINE    ;
 16:   ;
 17:  RUN SVEE2 ;
 18:  WAIT  12.00(sec) ;
 19:  LBL[5] ;
 20:  PR[10,1]=PR[10,1]+140    ;
 21:L PR[10] 35mm/sec FINE    ;
 22:  IF R[4]=1,JMP LBL[7] ;
 23:  LBL[7] ;
 24:  R[7]=1    ;
 25:  IF R[4]=1,JMP LBL[6] ;
 26:   ;
 27:   ;
 28:  PR[10,2]=PR[10,2]+13    ;
 29:L PR[10] 55mm/sec FINE    ;
 30:   ;
 31:   ;
 32:   ;
 33:   ;
 34:  PR[10,1]=PR[10,1]-140    ;
 35:L PR[10] 35mm/sec FINE    ;
 36:  IF R[4]=1,JMP LBL[8] ;
 37:  LBL[8] ;
 38:  R[8]=1    ;
 39:  IF R[4]=1,JMP LBL[6] ;
 40:   ;
 41:   ;
 42:   ;
```

```
 43:  PR[10,2]=PR[10,2]+13    ;
 44:L PR[10] 55mm/sec FINE    ;
 45:   ;
 46:   ;
 47:   ;
 48:   ;
 49:  JMP LBL[5] ;
 50:  LBL[6] ;
 51:  PR[57]=PR[59]    ;
 52:  R[4]=0    ;
 53:  WAIT   0.00(sec) ;
 54:   ;
 55:  PR[58,1]=PR[59,1]-25    ;
 56:  PR[58,2]=PR[59,2]-25    ;
 57:  PR[58,3]=PR[59,3]    ;
 58:  PR[58,4]=PR[59,4]    ;
 59:  PR[58,5]=PR[59,5]    ;
 60:  PR[58,6]=PR[59,6]    ;
 61:L PR[58] 100mm/sec FINE    ;
 62:  PR[10]=PR[58]    ;
 63:  R[4]=0    ;
 64:   ;
 65:  LBL[9] ;
 66:  PR[10,1]=PR[10,1]+50    ;
 67:L PR[10] 20mm/sec FINE    ;
 68:  IF R[4]=1,JMP LBL[10] ;
 69:  LBL[10] ;
 70:  R[16]=1    ;
 71:  IF R[4]=1,JMP LBL[20] ;
 72:   ;
 73:   ;
 74:  PR[10,2]=PR[10,2]+5    ;
 75:L PR[10] 20mm/sec FINE    ;
 76:   ;
 77:   ;
 78:   ;
 79:   ;
 80:  PR[10,1]=PR[10,1]-50    ;
 81:L PR[10] 20mm/sec FINE    ;
 82:  IF R[4]=1,JMP LBL[17] ;
 83:  LBL[17] ;
 84:  R[17]=1    ;
 85:  IF R[4]=1,JMP LBL[20] ;
 86:   ;
 87:   ;
 88:   ;
 89:  PR[10,2]=PR[10,2]+5    ;
 90:L PR[10] 20mm/sec FINE    ;
 91:   ;
 92:   ;
```

```
 93:   ;
 94:   ;
 95:  JMP LBL[9] ;
 96:  LBL[20] ;
 97:  R[4]=0    ;
 98:  PR[56]=PR[59]    ;
 99:  WAIT   0.00(sec) ;
100:   ;
101:   ;
102:   ;
103:  PR[55,1]=PR[59,1]-30    ;
104:  PR[55,2]=PR[59,2]-5    ;
105:  PR[55,3]=PR[59,3]    ;
106:  PR[55,4]=PR[59,4]    ;
107:  PR[55,5]=PR[59,5]    ;
108:  PR[55,6]=PR[59,6]    ;
109:L PR[55] 100mm/sec FINE    ;
110:  PR[10]=PR[55]    ;
111:  R[4]=0    ;
112:   ;
113:  LBL[11] ;
114:  PR[10,1]=PR[10,1]+5    ;
115:L PR[10] 55mm/sec FINE    ;
116:  IF R[4]=1,JMP LBL[12] ;
117:  LBL[12] ;
118:  R[18]=1    ;
119:  IF R[4]=1,JMP LBL[30] ;
120:   ;
121:   ;
122:  PR[10,2]=PR[10,2]+30    ;
123:L PR[10] 20mm/sec FINE    ;
124:   ;
125:   ;
126:   ;
127:   ;
128:  PR[10,1]=PR[10,1]+5    ;
129:L PR[10] 55mm/sec FINE    ;
130:  IF R[4]=1,JMP LBL[13] ;
131:  LBL[13] ;
132:  R[19]=1    ;
133:  IF R[4]=1,JMP LBL[30] ;
134:   ;
135:   ;
136:   ;
137:  PR[10,2]=PR[10,2]-30    ;
138:L PR[10] 20mm/sec FINE    ;
139:   ;
140:   ;
141:   ;
142:   ;
```

```
143:  JMP LBL[11] ;
144:  LBL[30] ;
145:  R[4]=0    ;
146:  PR[54]=PR[59]    ;
147:  WAIT   0.00(sec) ;
148:   ;
149:  PR[53,1]=PR[54,1]-8    ;
150:  PR[53,2]=PR[56,2]-5    ;
151:  PR[53,3]=PR[56,3]    ;
152:  PR[53,4]=PR[56,4]    ;
153:  PR[53,5]=PR[56,5]    ;
154:  PR[53,6]=PR[56,6]    ;
155:L PR[53] 55mm/sec FINE    ;
156:  PR[10]=PR[53]    ;
157:  R[4]=0    ;
158:   ;
159:   ;
160:   ;
161:   ;
162:  LBL[1] ;
163:  IF R[11]=111.111,JMP LBL[2] ;
164:  IF R[23]=99,JMP LBL[2] ;
165:  PR[10,1]=PR[10,1]+30    ;
166:L PR[10] 7mm/sec FINE    ;
167:   ;
168:  PR[10,2]=PR[10,2]+.5    ;
169:L PR[10] 7mm/sec FINE    ;
170:   ;
171:  PR[10,1]=PR[10,1]-30    ;
172:L PR[10] 7mm/sec FINE    ;
173:   ;
174:  PR[10,2]=PR[10,2]+.5    ;
175:L PR[10] 7mm/sec FINE    ;
176:   ;
177:  JMP LBL[1] ;
178:  LBL[2] ;
179:  IF R[11]=111.111,JMP LBL[14] ;
180:  JMP LBL[2] ;
181:  LBL[14] ;
182:   ;
183:   ;
184:  PR[2,1]=R[13]    ;
185:  PR[2,2]=R[14]    ;
186:  PR[2,3]=R[15]    ;
187:  PR[2,4]=R[20]    ;
188:  PR[2,5]=R[21]    ;
189:  PR[2,6]=R[22]    ;
190:L PR[2] 15mm/sec FINE    ;
```