Simulacija primjene digitalizatora u korekciji putanje alata prema odstupanju oblika pripremka

Legin, Mihovil

Master's thesis / Diplomski rad

2023

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:186385

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-18**

Repository / Repozitorij:

Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mihovil Legin

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor: Student:

Dr. sc. Tomislav Staroveški, dipl. ing. Mihovil Legin

Zagreb, 2023.

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

Zahvaljujem se mentoru dr. sc. Tomislavu Staroveškom na stručnoj pomoći, pristupačnosti i savjetima koji su mi uvelike pomogli u izradi ovoga rada.

Zahvaljujem se svim zaposlenicima katedre za alatne strojeve, posebno asistentu Luki Drobilu, mag.ing.mech. na pruženoj stručnoj pomoći, asistentici Dori Bagarić, mag.ing.mech. te kolegi Luki Repalustu na suradnji i pomoći tijekom izrade ovoga rada.

Posebno hvala mojoj obitelji i prijateljima na pruženoj podršci tijekom studiranja.

Mihovil Legin

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Proizvodno inženjerstvo, inženjerstvo materijala, industrijsko inženjerstvo i menadžment, mehatronika i robotika, autonomni sustavi i računalna inteligencija

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 23 - 6 / 1**	
Ur.broj: **15 - 23 -**	

DIPLOMSKI ZADATAK

Student: **Mihovil Legin** JMBAG: 0035219899

Naslov rada na hrvatskom jeziku: **Simulacija primjene digitalizatora u korekciji putanje alata prema odstupanju oblika pripremka**

Naslov rada na engleskom jeziku: **Simulation of 3D scanner-based tool path correction for workpiece shape deviation**

Opis zadatka:

Na Katedri za alatne strojeve Fakulteta strojarstva i brodogradnje u tijeku je provedba projekta ARCOPS, čiji je cilj razvoj robotskog obradnog sustava (ćelije) za brušenje tankostjenih pozicija (ispitnih uzoraka) većih dimenzija. Predmetni ispitni uzorci mogu značajno odstupati oblikom, a sile koje nastaju kao posljedica djelovanja stezne naprave mogu ih dodatno elastično deformirati. Navedena odstupanja mogu imati negativan utjecaj na kvalitetu obrađene površine ukoliko se putanja alata ne korigira prema odstupanju razmatrane površine ispitnog uzorka. Stoga se u sklopu projektnih aktivnosti planira istražiti utjecaj odstupanja oblika i elastične deformacije ispitnih uzoraka na kvalitetu obrađene površine.
U radu je potrebno sljedeće:

1. Na osnovi 3D modela zadanog ispitnog uzorka izraditi 3D modele nekoliko varijanti ispitnih uzoraka s različitim stupnjevima odstupanja oblika. 3D modele je potrebno pripremiti primjenom CAD/CAM sustava Catia V5.
2. Izraditi putanje alata za brušenje zadanog ispitnog uzorka i generirati NC program za obradu industrijskim robotom ugrađenim u ćeliju, tipa ABB IRB 6660-205/1.9.
3. Primjenom softvera za analizu rezultata digitalizacije GOM Inspect formirati postav u kojemu će se zadani ispitni uzorak koristiti kao idealni (CAD) model, a pripremljeni ispitni uzorci s odstupanjima kao simulirani rezultati digitalizacije. Postav treba sadržavati niz kontrolnih točaka čije koordinate odgovaraju koordinatama točaka putanje alata prema NC programu iz prethodnog koraka. Rezultati analize trebaju prikazati odstupanja koordinata i vektora normala razmatranih kontrolnih točaka na uzorcima s odstupanjima.
4. Izraditi NC program koji sadrži putanju alata korigiranu prema rezultatima analize iz prethodnog koraka.
5. Simulirati nekorigirani i korigirani NC program koristeći programski paket ABB *Robot Studio*.
6. Donijeti zaključke rada.

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

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
28. rujna 2023.	30. studenoga 2023.	4. – 8 . prosinca 2023.

Zadatak zadao: Predsjednik Povjerenstva:

Izv. prof. dr. sc. Tomislav Staroveški Prof. dr. sc. Ivica Garašić

SADRŽAJ

SADRŽAJ I
POPIS SLIKA III
POPIS TABLICA V
SAŽETAK VI
SUMMARY VII
1. UVOD 1
1.1. Greške odstupanja oblika pripremka 1
1.2. Idejno rješenje problema 4
2. DIGITALIZACIJA I 3D SKENIRANJE 7
2.1. Karakteristike i primjena 3D skenera 7
2.2. Metode 3D skeniranja 8
2.2.1. Kontaktni 3D skeneri 8
2.2.2. Beskontaktni 3D skeneri 8
2.2.2.1. 3D skeniranje strukturiranim svjetlom 8
2.2.2.2. Lasersko 3D skeniranje 9
2.2.2.3. 3D skeniranje pulsirajućim laserskim zrakama 10
2.2.2.4. Fotogrametrija 10
3. ROBOTSKA ĆELIJA 12
3.1. Robot ABB IRB 6660-205/1.9 12
3.2. ABB IRBP A500 14
3.3. Robot ABB IRB 4600-40/2.55 15
3.3.1. ATOS 5X 16
3.4. Stezna naprava 17
4. IZRADA REFERENTNOG CAD MODELA, PUTANJE ALATA I ISPITNIH UZORAKA 18
4.1. Referentni CAD model 18
4.2. Putanja alata 19
4.3. Ispitni uzorci (simulirani rezultati digitalizacije) 21
5. GENERIRANJE POČETNOG RAPID KODA 24
5.1. RoboDK 24
5.2. RobotStudio 26
6. FORMIRANJE MJERNOG POSTAVA 29
6.1. Ručno postavljenje 29
6.1.1. Učitavanje stezne naprave, referentne ploče i ispitnog uzorka u GOM Inspect Pro 29
6.1.2. Poravnanje 30
6.1.3. Konstruiranje točaka na površinu CAD modela i uzorka 32
6.2. Automatizirano postavljanje mjernog sustava 33
6.2.1. Automatiziranje poravnanja idealnog i deformiranog modela uzorka 34
6.2.2. Automatizirano stvaranje točaka na površini CAD modela 34

6.2.3. Automatizirano stvaranje točaka na površini uzorka 37
6.2.4. Korekcija putanje alata 37
7. SIMULACIJA KORIGIRANE PUTANJE ALATA 42
7.1. Uzorak s maksimalnim odstupanjem od 5 mm 42
7.2. Uzorak s maksimalnim odstupanjem od 15 mm 43
8. ZAKLJUČAK 45
LITERATURA 46
PRILOZI 47

POPIS SLIKA

Slika 1 Pravilno i nepravilno stegnut pripremak 2
Slika 2 Nepravilno stegnut pripremak 2
Slika 3 Putanja alata kod a) idealnog i b) deformiranog pripremka 4
Slika 4 Faze u procesu obrade materijala pomoću robota 5
Slika 5 Modifikacija faza u procesu obrade materijala pomoću robota 5
Slika 6 a) nekorigirana i b) korigirana putanja alata 6
Slika 7 3D skeniranje strukturiranim svjetlom [3] 9
Slika 8 Lasersko 3D skeniranje [4] 9
Slika 9 3D skeniranje pulsirajućim laserskim zrakama [6] 10
Slika 10 Fotogrametrija pomoću više fotoaparata [7] 11
Slika 11 Robotska ćelija 12
Slika 12 ABB IRB 6660-205/1.9 [9] 13
Slika 13 radni prostor robota ABB IRB 6660-205/1.9 [8] 14
Slika 14 ABB IRBP A500 [11] 14
Slika 15 ABB IRB 4600-40/2.55 [12] 15
Slika 16 ATOS 5X [14] 16
Slika 17 CAD model stezne naprave 17
Slika 18 Referentni CAD model 18
Slika 19 Putanja alata 19
Slika 20 Smjer gibanja alata 20
Slika 21 Dimenzije alata 20
Slika 22 Generiranje APT koda 21
Slika 23 Crtež ispitnog uzorka 21
Slika 24 Uzorci s različitim odstupanjima 22
Slika 25 Primjer uzorka korištenog u mjerenju odstupanja 22
Slika 26 Pogrešno izmjereno odstupanje 23
Slika 27 Pravilno orijentirana stezna naprava s referentnom pločom i definirana nul-točka obrade 25
Slika 28 Promjena orijentacije stezne naprave s referentnom pločom 26
Slika 29 Prikaz putanje alata u *RobotStudio*-u 27
Slika 30 Simulacija obrade referentne ploče 27
Slika 31 Izdvajanje točaka u RAPID kodu 28
Slika 32 Opcija koja omogućava da se učitana datoteka prikazuje kao stezna naprava 29
Slika 33 Stezna naprava i referentna ploča 30
Slika 34 Koordinatni sustav deformiranog ispitnog uzorka 31
Slika 35 Provjera poravnanja 32
Slika 36 Točka s pripadajućim vektorom normale na idealnoj ploči 32
Slika 37 Točka na deformiranom ispitnom uzorku 33
Slika 38 Povezivanje podataka iz CAM i GOM sustava s robotom 33
Slika 39 Skripta za automatiziranje poravnanja 34
Slika 40 Dio koda koji čita RAPID putanje od komentara "! Pocetak" do "! Kraj" 35
Slika 41 Dio koda koji izdvaja koordinate točaka iz putanje i postavlja ih na referentnu ploču 35
Slika 42 *parse_rapid_points* 35
Slika 43 *parse_rapid_points* na primjeru jedne linije RAPID koda 36
Slika 44 *quat2jv* 36
Slika 45 Skripta za automatizirano stvaranje točaka na površini uzorka 37

Slika 46 Dio koda za iščitavanje koordinata točaka i projekcija vektora normala te njihovo spremanje u tablicu 38
Slika 47 Dio koda koji iščitava linije iz originalnog RAPID koda 38
Slika 48 Dodatni vektor za definiciju orijentacije 38
Slika 49 Zapis Z vektora dobiven pomoću podataka iz GOM Inspect Pro-a 39
Slika 50 Dio glavnog koda za izračun korigiranih vrijednosti kvaterniona 40
Slika 51 kod funkcije *jv2rotmtx* 40
Slika 52 kod funkcije *rotmtx2quat* 40
Slika 53 ažuriranje podataka o koordinatama točaka i kvaterniona u RAPID kodu 40
Slika 54 Izgled korigiranog RAPID koda (dio koji opisuje gibanje alata) 41
Slika 55 Položaj točaka na referentnoj (plavo) i deformiranoj (zeleno) ploči te pripadajući vektori normala 42
Slika 56 Korigirana putanja alata prikazana u RobotStudio-u 42
Slika 57 Položaj točaka i pripadajući vektori normala na referentnoj (plavo) i deformiranoj (zeleno) ploči 43
Slika 58 Nekorigirana putanja alata u RobotStudio-u 43
Slika 59 Korigirana putanja alata u RobotStudio-u 44

POPIS TABLICA

Tablica 1 Specifikacije robota ABB IRB 6660-205/1.9 [10] 13
Tablica 2 Tablica 2 Specifikacije robota ABB IRB 4600-40/2.55 [12] 15

SAŽETAK

Tema ovog diplomskog rada je simulacija primjene digitalizatora u korekciji putanje alata prema odstupanju oblika pripremka. U programskom paketu GOM Inspect Pro formiran je mjerni postav u kojem je sklop stezne naprave i idealnog ispitnog uzorka korišten kao referentni CAD model, a sklop s deformiranim uzorkom kao simulirani rezultat digitalizacije. Pomoću formiranog mjernog postava došlo se do podataka o koordinatama točaka (vrha alata) i orijentacije alata u odnosu na uzorak. Na temelju dobivenih podataka upotrebom programskog koda izvršila se korekcija putanje alata. Na kraju rada korigirana i nekorigirana putanja alata simulirane su u programskom paketu RobotStudio gdje je verificirana ispravnost korigirane putanje a time i programskog koda.

Ključne riječi: GOM Inspect Pro, RobotStudio, korekcija putanje alata

SUMMARY

The theme of this master's thesis is the simulation of 3D scanner-based tool path correction for workpiece shape deviation. In the program package GOM Inspect Pro, a measuring setup was created in which the assembly of the clamping device and the ideal test sample was used as a reference CAD model, and the assembly with the deformed sample served as a simulated digitization result. Using the formed measuring setup, the data on the coordinates of the points (tip of the tool) and the orientation of the tool in relation to the sample was obtained. Based on the obtained data, using a program code the correction of the tool path was performed. In order to validate the correctness of the corrected path and thus the program code, both the corrected and uncorrected tool paths were simulated in the RobotStudio software package

Key words: GOM Inspect Pro, RobotStudio, tool path correction

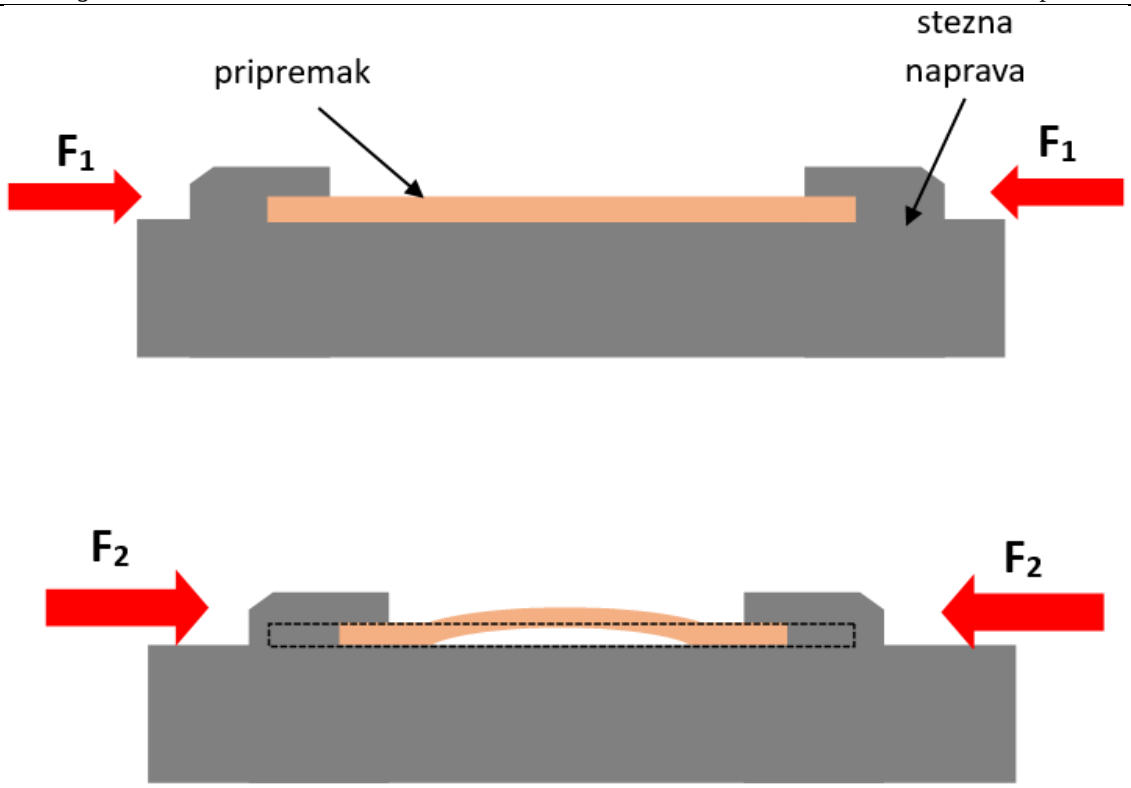
1. UVOD

Kod kompozitnih tankostijenih pozicija moguće su devijacije oblika koje mogu nastati kao posljedica prethodnih operacija u proizvodnom procesu (glodanje, tokarenje…). Uz navedeno, izvedba stezne naprave, kao i sile koje nastaju kao posljedica stezanja mogu unijeti dodatne elastične deformacije te na taj način uzrokovati daljnje greške prilikom obrade. Tankostijene pozicije, kakvima se ovaj rad bavi, sklone su savijanju te je stoga potreban oprez prilikom njihovog stezanja. U slučaju obrade deformiranih pozicija, vođenje alata po idealnom modelu može prouzrokovati neujednačenu kvalitetu obrađene površine, ukoliko se putanja ne prilagodi odstupanjima od idealnog oblik.

Integracijom beskontaktnog 3D skeniranja u početnoj fazi procesa brušenja ove bi se greške mogle detektirati prije početka same obrade te bi se one mogle popraviti (npr. ponovnim namještanjem pripremka u steznu napravu) ili bi se mogla korigirati putanja alata prema odstupanju oblika pripremka.

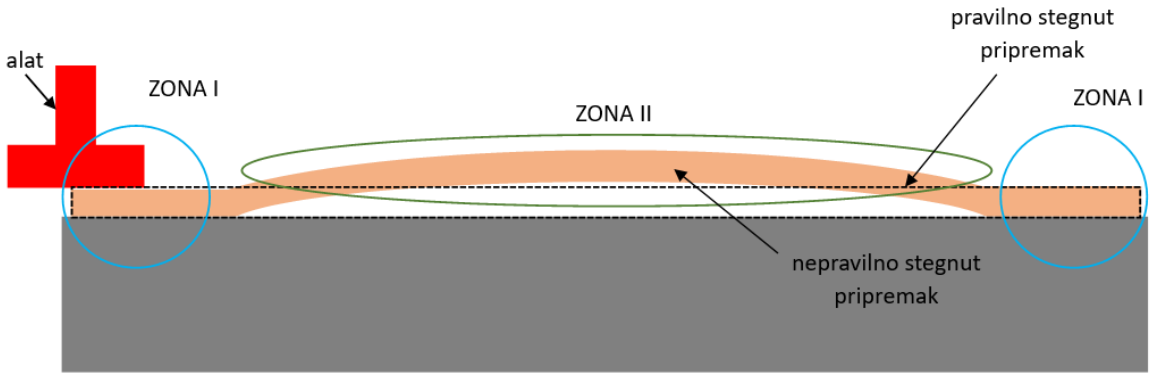
1.1. Greške odstupanja oblika pripremka

Osim što nedostaci koje se mogu pronaći na proizvodu nakon procesa brušenja mogu biti uzrokovani uslijed samog procesa brušenja (npr. pogrešnim odabirom parametara obrade, korištenjem neodgovarajućeg ili istrošenog brusnog papira) oni mogu biti uzrokovani i greškama koje su se unijele u proces netom prije samog brušenja. Slika 1 gore prikazuje pravilno stegnut pripremak. U slučaju kada bi se sila stezanja povećala iznad neke kritične vrijednosti ($F_2 > F_{kritično}$), ovisno o materijalu i obliku pripremka, postoji mogućnost da se pripremak elastično deformira (slika 1 dolje).



Slika 1 Pravilno i nepravilno stegnut pripremak

Zbog prevelike sile stezanja dolazi do elastične deformacije pripremka te se na njemu mogu uočiti dvije zone (slika 2). U prvoj zoni ne dolazi do velikog odstupanja pripremka (u odnosu na slučaj pravilnog stezanja) jer stezne čeljusti u tom području drže pripremak na istoj razini (postoji mogućnost da se čeljusti blago zakrive prilikom stezanja, međutim za ovu situaciju to je zanemarivo).

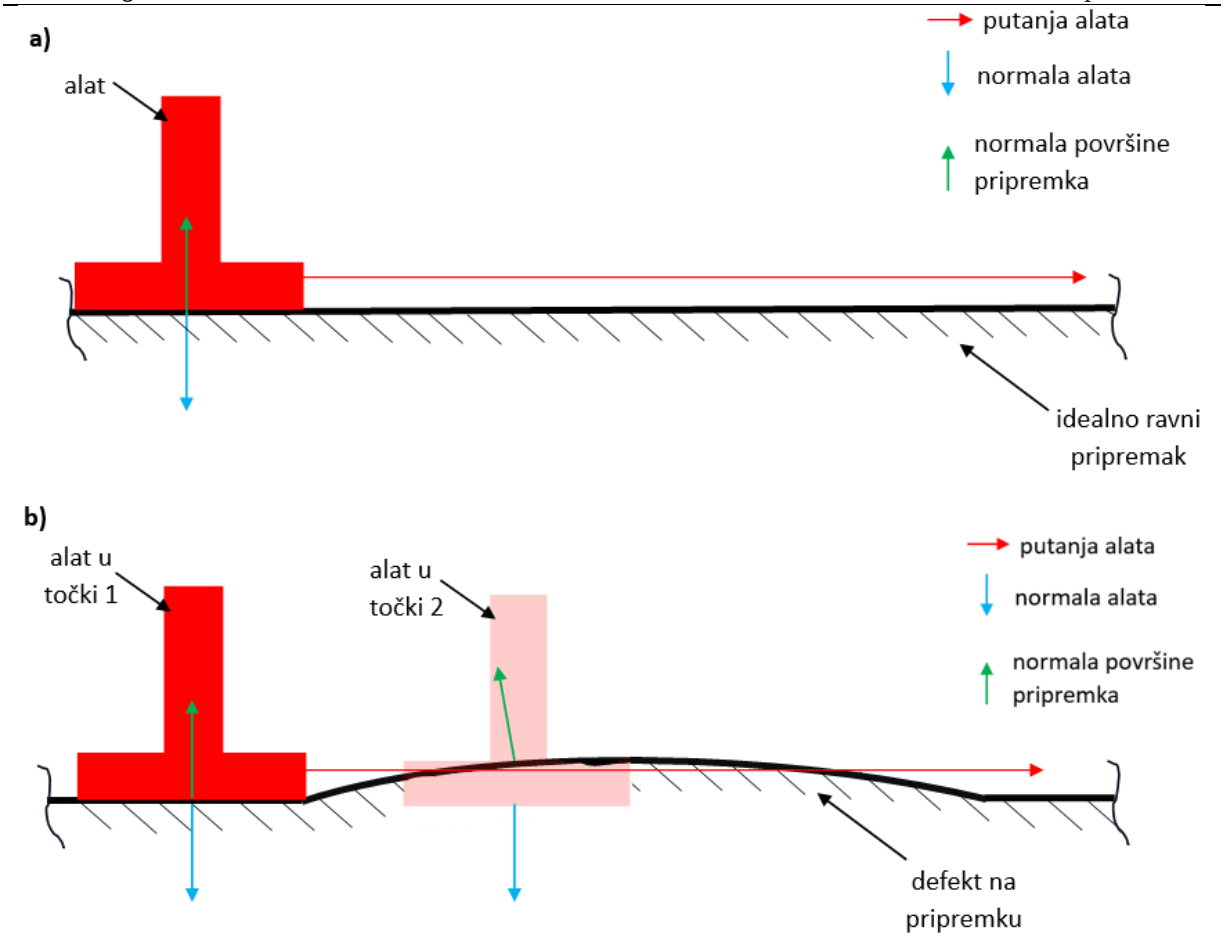


Slika 2 Nepravilno stegnut pripremak

U drugoj zoni dolazi do elastične deformacije, odnosno u toj je zoni stvarna površina pripremka iznad pretpostavljene. U ovoj zoni alat ulazi u zahvat s materijalom te započinje proces odvajanja čestica materijala.

Kod obradnih sustava sličnih korištenom za potrebe ovoga rada, glavni prigon najčešće nije ugrađen izravno na prihvatnicu robota (ili neki drugi posmični prigon ako se radi o klasičnim alatnim strojevima), već se između posmičnog i glavnog prigona najčešće ugrađuje i prigon za regulaciju aksijalne sile. Također, brusni papir nije kruto vezan uz glavni prigon, već je to sučelje izvedeno spužvastim materijalom. Sučelje između alata (brusnog papira) i glavnog prigona, kao i prigon za regulaciju sile mogu kompenzirati promjene u sili koje se mogu pojaviti u ovoj zoni međutim ne mogu korigirati orijentaciju alata u odnosu na obrađivanu površinu. Zbog toga se kontaktna površina (zona zahvata između brusnog papira i obrađivane površine) smanjuje što može uzrokovati značajno povećanje naprezanja u zoni zahvata (čak i ako je aksijalna sila konstantna), a time i neujednačenu kvalitetu površine.

Kao što je ranije napomenuto, osim grešaka koje nastaju pri stezanju pripremka, moguće su i greške na samom pripremku koje su uzrokovane nekom od prethodnih obrada. Neovisno o kojem se tipu greške radi, putanja alata programirana je za slučaj da je pripremak idealan (nedeformiran, pravilno stegnut). Iako se ovaj rad bavi slučajem u kojem normala alata ima isti smjer kao i normala površine pripremka (slika 3 a) to ne mora uvijek biti tako. Normala alata ne mora uvijek biti okomita na površinu koja se obrađuje (npr. obrada kutnom brusilicom) pa time i normala alata ne mora imati isti smjer kao i normala površine pripremka.



Slika 3 Putanja alata kod a) idealnog i b) deformiranog pripremka

Slika 3 b) prikazuje slučaj u kojem se na pripremku nalazi neki defekt. Budući da je putanja alata programirana za idealan pripremak, alat se nastavlja gibati pravocrtno. U području defekta, normala alata nema isti smjer kao i normala površine pripremka. Na slici 3 prikazan je dvodimenzionalni slučaj iz kojeg se može vidjeti da normala površine pripremka odstupa u odnosu na normalu alata. Također, položaj točke 2 je iznad očekivane. U dvodimenzionalnom slučaju, položaj točke može odstupati u dva smjera (X, Y osi), a normala može odstupati za određeni kut u jednoj ravnini (XY ravnina). U trodimenzionalnom prostoru, postoji mogućnost da neka točka na pripremku odstupa od idealne točke u sva tri smjera (X, Y, Z osi) te da normala na površinu pripremka u toj točki odstupa u sve tri ravnine (XY, XZ, YZ). Ovakva obrada pripremka (slika 3) može negativno utjecati na konačnu kvalitetu obrađene površine.

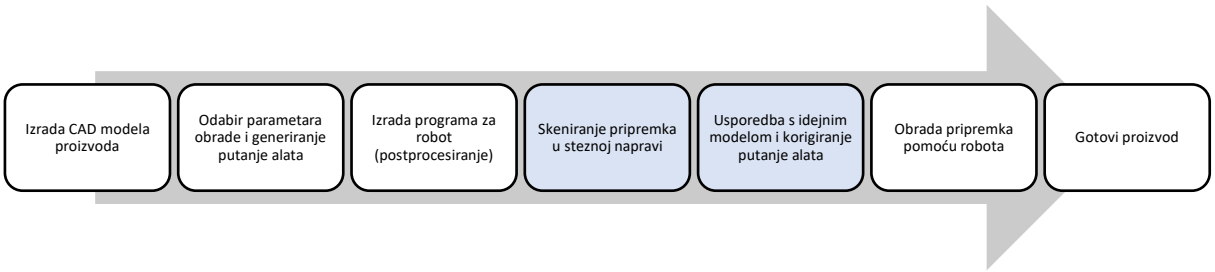
1.2. Idejno rješenje problema

Slika 4 prikazuje faze prilikom obrade nekog predmeta pomoću robota (ili nekog višeosnog stroja). Najprije je potrebno izraditi CAD model pripremka i gotovog proizvoda. Zatim se pomoću CAD modela generira putanja alata te se ovisno o odabranom alatu i materijalu pripremka odabiru parametri obrade. U slučaju korištenja robota, generiranu putanju

alata potrebno je pretvoriti u jezik koji će robot razumjeti (postprocesiranje). Nakon toga slijedi obrada pomoću robota.

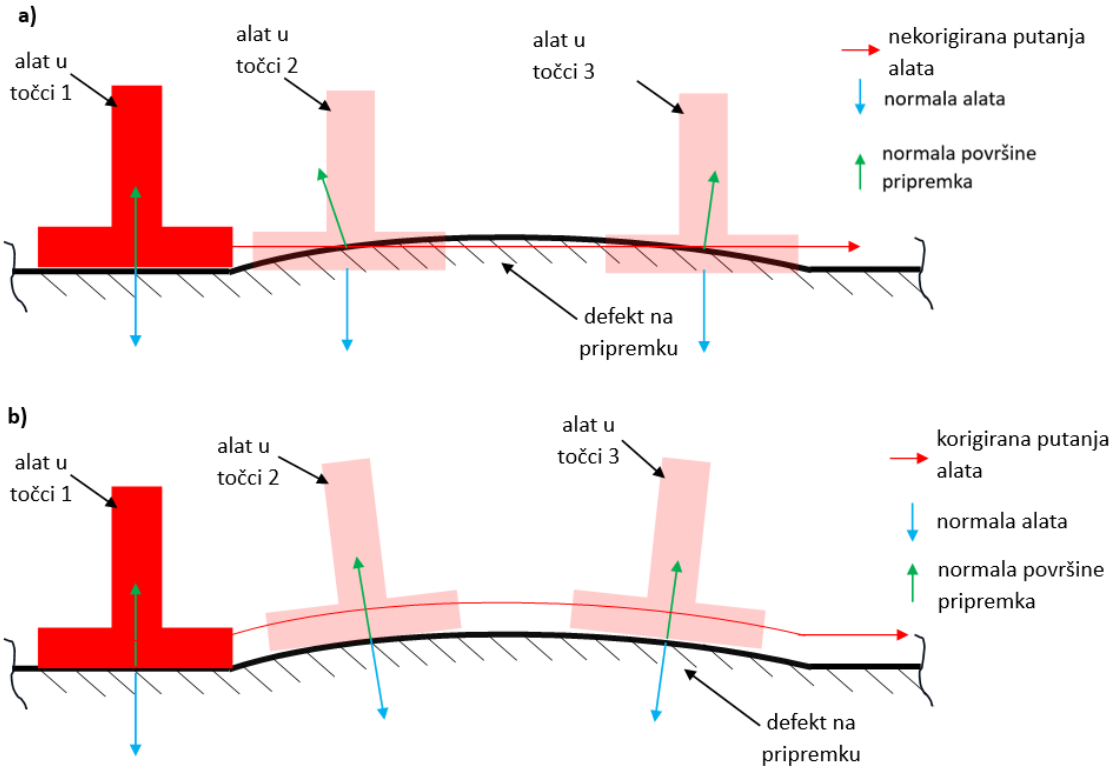
Slika 4 Faze u procesu obrade materijala pomoću robota

Način na koji će se robot (a time i alat) gibati, a samim time i odvijati proces obrade definiran je u CAM sustavu. Prije ili tijekom rada robot nema nikakvu povratnu vezu o tome je li pripremak pravilno pozicioniran i stegnut u steznu napravu. Robot samo izvršava naredbe zadane u početnoj fazi procesa. Upravo zbog toga i postoji mogućnost da se na kraju procesa brušenja dobije površina koja ne zadovoljava traženu kvalitetu. Slika 5 prikazuje moguću modifikaciju ranije spomenutog procesa.



Slika 5 Modifikacija faza u procesu obrade materijala pomoću robota

Implementacijom 3D skenera, sustav bi bio u mogućnosti detektirati defekte na pripremku te korigirati putanju alata s obzirom na izmjerena odstupanja (slika 6). U tom slučaju u svakoj točki obrade os alata imala bi isti smjer kao i normala na površinu pripremka.



Slika 6 a) nekorigirana i b) korigirana putanja alata

2. DIGITALIZACIJA I 3D SKENIRANJE

Digitalizacija je proces koji uključuje prikupljanje podataka o obliku i fizičkom izgledu razmatranog objekta, okoliša i osobe i potom korištenje tih podataka za izradu digitalnih 3D modela [1]. Digitalna kopija objekta može se koristiti za njegov trodimenzionalni ispis, kontrolu kvalitete ili za njegovo digitalno arhiviranje.

U sklopu robotske ćelije koja je korištena za simuliranje obrade, za digitalizaciju predmeta iz realnog svijeta koristi se uređaj ATOS 5X. U nastavku poglavlja dan je kratki pregled karakteristika 3D skenera kao i područja njihove primjene u strojarstvu te su nabrojene metode 3D skeniranja.

2.1. Karakteristike i primjena 3D skenera

Primjenom beskontaktnog 3D skeniranja minimizira se mogućnost ljudske greške prilikom nepravilnog korištenja mjernih instrumenata ili pogrešnog očitanja mjera (pod uvjetom da je mjerni sustav pravilno postavljen). U odnosu na konvencionalne mjerne metode, pravilnim postavljanjem mjernog sustava može se ubrzati postupak mjerenja objekata složene geometrije. Integracijom 3D skenera u ostalu opremu poduzeća (kao što su roboti) otvara se mogućnost automatizacije procesa kontrole kvalitete. Upotrebom beskontaktnih 3D skenera smanjuje se mogućnost oštećenja predmeta prilikom mjerenja. Kod skeniranja prozirnih predmeta ili predmeta s visoko reflektirajućom površinom moguće je otežano skeniranje. U tom slučaju takve je predmete potrebno pripremiti za skeniranje na način da se na njih nanese tanki sloj boje ili praha. 3D skeniranjem dobiva se znatno veća količina podataka čime se otvaraju nove mogućnosti. Jedan od nedostataka upotrebe 3D skenera jest njihova visoka cijena. Iako razvojem tehnologije to danas ne vrijedi u potpunosti, svejedno je potrebno uložiti više novca u kvalitetan 3D skener koji nudi veću preciznost od ostalih.

U strojarstvu 3D skeniranje se koristi kod kontrole kvalitete, za povratno inženjerstvo, izradu prototipa i drugo. Implementacijom 3D skeniranja u kontrolu kvalitete ubrzava se proces kontrole i otvara se mogućnost puno detaljnije provjere svojstava i mjera proizvoda koja klasičnim metodama nisu bila moguća ili su bila dugotrajna. Skenirani predmet može se usporediti sa CAD modelom te se na računalu omogućuje prikaz njegovog odstupanja od idealnog oblika. U slučaju da se neki gotovi predmet želi modificirati, pomoću 3D skenera moguće je brzo napravi precizan CAD model koji se zatim u nekom od softverskih paketa modificira [1].

2.2. Metode 3D skeniranja

Postoje razne izvedbe tehnologija skeniranja ovisno o tome za što su namijenjene. Naprimjer, u slučaju 3D skeniranja velikih površina uz pomoć zrakoplova, od skenera se očekuje da pokrivaju što veći volumen dok preciznost i nije toliko bitna budući da su skenirani predmeti (kuče, zgrade, šume) veliki, dok se prihvatljivost odstupanja može gledati čak i u metrima. S druge strane, kod skeniranja manjih predmeta, posebno za potrebe preciznog strojarstva, od skenera se očekuje izrazito velika preciznost i točnosti u stotinkama milimetra. Osnovna podjela 3D skenera je na kontaktne i beskontaktne.

2.2.1. Kontaktni 3D skeneri

Kod kontaktnog 3D skeniranja informacije o predmetu koji se skenira dobivaju se ostvarivanjem kontakta između razmatranog objekta i ticala koje najčešće ima oblik kugle. Sustav za digitalizaciju prati položaj ticala u prostoru te se na taj način dolazi do podataka o geometriji predmeta. Predmet je čvrsto stegnut kako bi se smanjile greške te se on ne smije micati tijekom skeniranja.

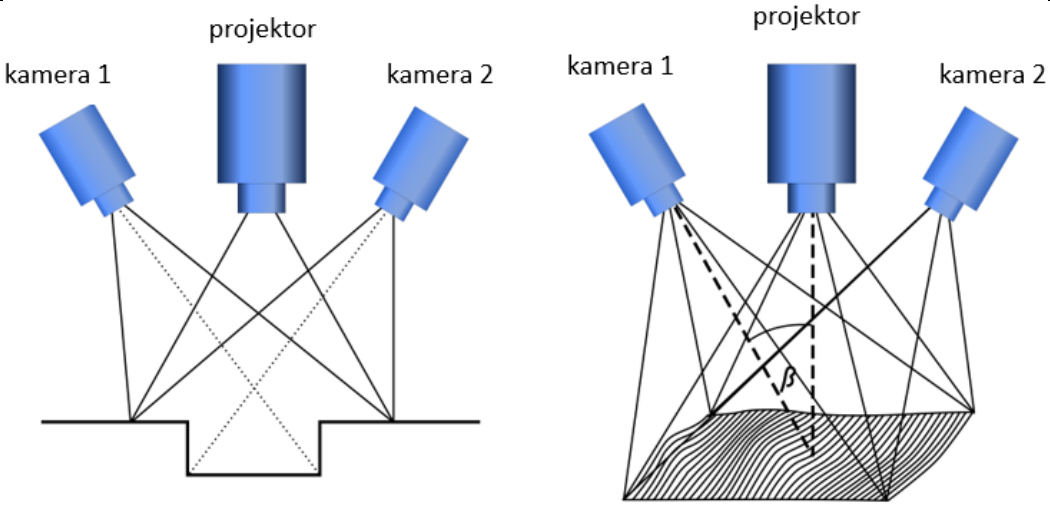
Najveća prednost ovih skenera u odnosu na beskontaktne je njihova preciznost. Također, ovim skenerima moguće je skenirati prozirne predmete i predmete koji imaju izuzetno reflektirajuću površinu bez potrebe za pripremom razmatranih površina. Glavni nedostatak je što su znatno sporiji od beskontaktnih skenera. Ticalo mora prijeći po čitavoj površini predmeta što oduzima puno vremena [2]. Također, kontaktni skeneri ne mogu mjeriti jako mekane predmete. Budući da dolazi do kontakta između ticala i predmeta, trošenje ticala može utjecati na točnost mjerenja.

2.2.2. Beskontaktni 3D skeneri

Ovisno o principu rada, beskontaktno 3D skeniranje se može podijeliti u nekoliko kategorija. To su 3D skeniranje strukturiranim svjetlom, lasersko 3D skeniranje, 3D skeniranje pulsirajućim laserskim zrakama, te fotogrametrija.

2.2.2.1. 3D skeniranje strukturiranim svjetlom

Uređaji za 3D skeniranje strukturiranim svjetlom sastoje se od projektora te jedne ili najčešće dvije kamere (slika 7) [3]. Izvor svjetla iz glave projektora projicira niz uzoraka na predmet koji se skenira. Sustav za digitalizaciju detektira značajke na slikama koje su nastale kao posljedica osvjetljavanja razmatrane površine strukturiranim izvorom svjetlosti, nakon čega se postupkom triangulacije dolazi do koordinata točaka na površini. Glavna prednost ove metode jest brzina skeniranja.

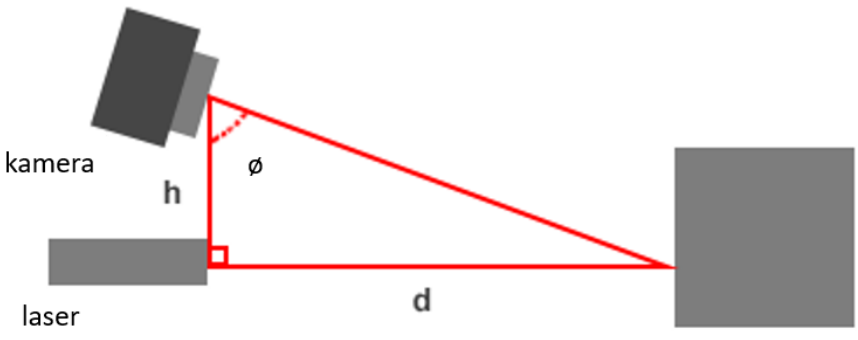


Slika 7 3D skeniranje strukturiranim svjetlom [3]

2.2.2.2. Lasersko 3D skeniranje

Lasersko 3D skeniranje se također zasniva na triangulaciji. Ovi skeneri sastoje se od lasera i kamere. Kamera, laser i točka u kojoj laserska zraka pogađa predmet formiraju trokut (slika 8). U ovom trokutu, udaljenost između kamere i lasera je poznata kao i njihov međusobni kut. Pomoću ovih poznatih vrijednosti 3D skener može izračunati udaljenosti između lasera i površine predmeta. Slika 8 prikazuje slučaj u kojem skener projicira točku na površinu. Većina modernih 3D skenera koji koriste ovu tehnologiju projicira liniju. To im omogućuje mjerenje čitavog presjeka predmeta odjednom [4].

Zbog jednostavne konstrukcije ovi su skeneri jeftiniji od ostalih [5]. Kao i kod gotovo svih optičkih skenera, nedostatak ove metode je taj što kvaliteta skena uvelike ovisi o svojstvima površine predmeta.

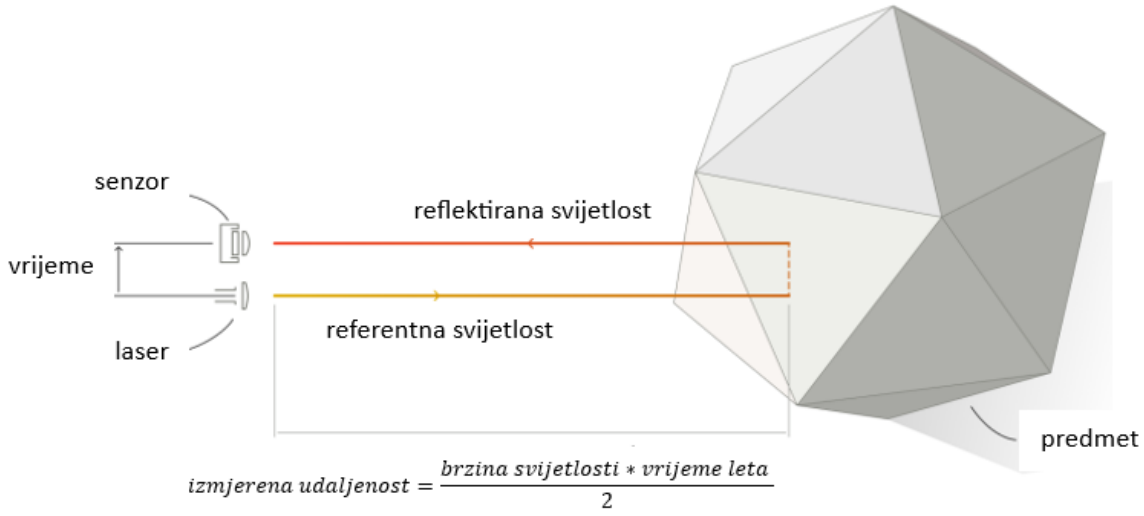


Slika 8 Lasersko 3D skeniranje [4]

2.2.2.3. 3D skeniranje pulsirajućim laserskim zrakama

Skeneri koji koriste ovu metodu određuju udaljenost od predmeta na način da emitiraju lasersku zraku na predmet i mjere koliko je vremena potrebno zraci da se odbije od površine i vrati do senzora (*Time-of-Flight* metoda). Kako je brzina svjetlosti poznata, vrijeme leta zrake može se koristiti za izračunavanje prijeđene udaljenosti (slika 9) [4].

Ovi skeneri mogu istovremeno izmjeriti udaljenost samo jedne točke dok skeneri koji koriste lasersko 3D skeniranje mogu istovremeno izmjeriti udaljenost više točaka koje se nalaze u liniji. Kako bi se skenirala čitava površina razmatranog objekta, ovi skeneri imaju ugrađena zrcala s odgovarajućim prigonima za horizontalni i vertikalni otklon zraka. Većina 3D skenera koji koriste *Time-of-Flight* metodu mogu izmjeriti na desetke tisuća točaka u sekundi. Prednost ovih skenera je ta što mogu skenirati velike volumene te se stoga oni koriste u geodeziji i za skeniranje okoliša [4].



Slika 9 3D skeniranje pulsirajućim laserskim zrakama [6]

2.2.2.4. Fotogrametrija

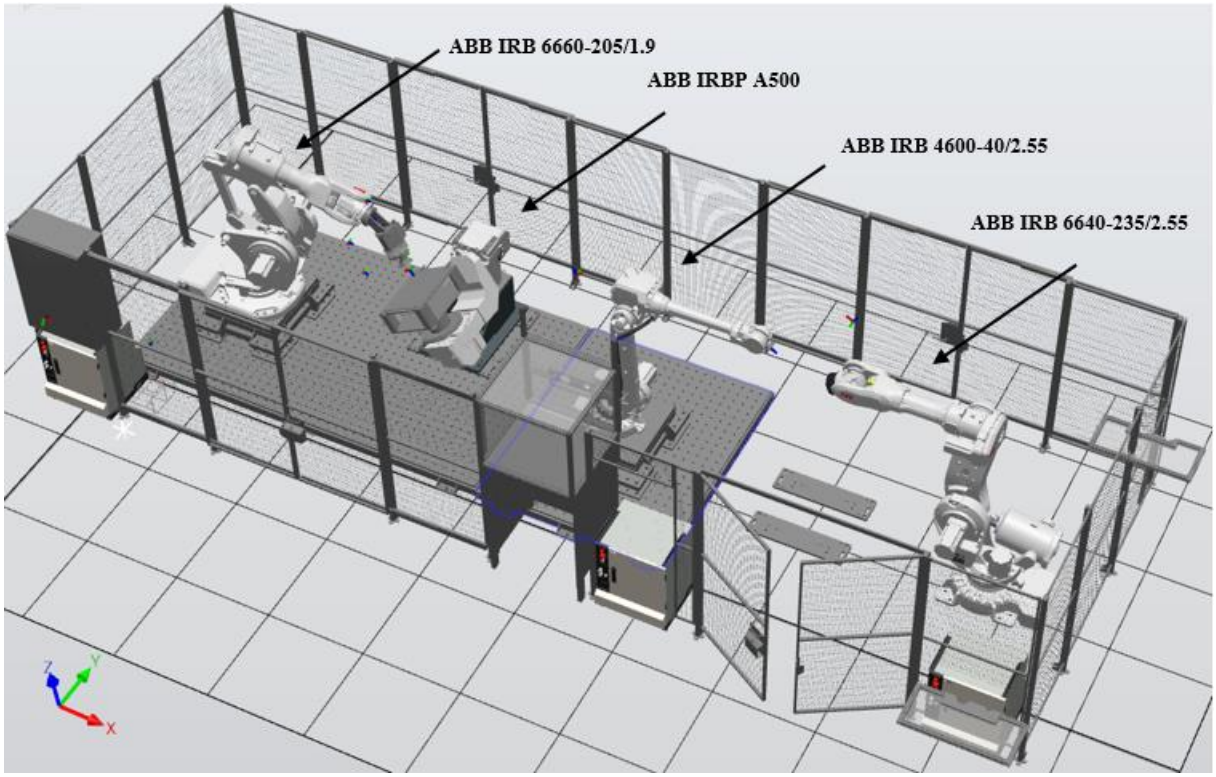
Fotogrametrija je metoda 3D skeniranja u kojoj se predmet fotografira tako da se dobije slika razmatranog objekta iz što većeg broja orijentacija te se pomoću tih fotografija stvara digitalizirani oblik stvarnog predmeta. Snima se veliki broj fotografija, obično s nekim preklapanjem, jer algoritam izračunava geometriju na temelju promjena s jedne fotografije na drugu. Tijekom fotografiranja predmet miruje.

Predmet se može fotografirati s više fotoaparata istovremeno (slika 10). Kvaliteta digitaliziranog predmeta uvelike ovisi o rezoluciji fotoaparata koji je korišten za fotografiranje te kvaliteti i optičkoj rezoluciji objektiva.

Slika 10 Fotogrametrija pomoću više fotoaparata [7]

3. ROBOTSKA ĆELIJA

Slika 11 prikazuje robotsku ćeliju koja je korištena za simulaciju obrade. Ćelija je napravljena u softveru *RobotStudio*. U njoj se nalaze tri industrijska robota i jedan okretno-nagibni stol. Za potrebe rada korišten je samo prvi robot (IRBP 6660-205/1.9) i okretno-nagibni stol (IRBP A500) na kojem je postavljena stezna naprava. U stvarnom procesu koristio bi se i robot ABB IRB 4600-40/2.55 za potrebe skeniranja jer je na njemu montiran mjerni uređaj ATOS 5X.



Slika 11 Robotska ćelija

3.1. Robot ABB IRB 6660-205/1.9

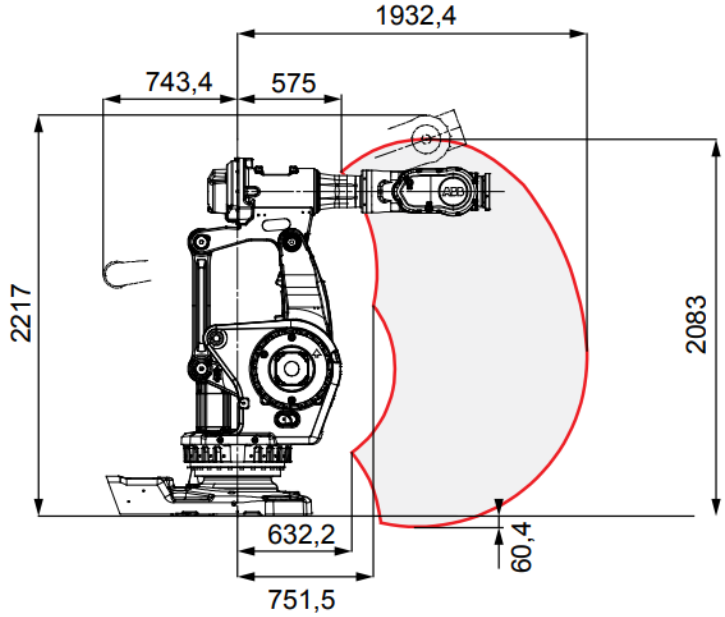
ABB je švedsko-švicarski proizvođač robota. Model 6660-205/1.9 (slika 12) je industrijski robot s šest stupnjeva slobode gibanja, nosivosti 205 kg i dohvata 1,9 m. Ostale specifikacije robota dane su u tablici 1. Ovaj je model namijenjen za procese kao što su glodanje, rezanje, piljenje, brušenje [8].

Slika 12 ABB IRB 6660-205/1.9 [9]

Tablica 1 Specifikacije robota ABB IRB 6660-205/1.9 [10]

Broj stupnjeva slobode gibanja	6
Nosivost robota	205 kg
Masa robota	1730 kg
Točnost ponavljanja	0,07 mm
Točnost pozicioniranja	0,18 mm
Dohvat	1,9 m

Slika 13 prikazuje radni prostor robota. Poznavanje radnog prostora robota je bitno kako bi se robot pravilno pozicionirao unutar robotske ćelije. Pravilnim pozicioniranjem robota unutar ćelije osigurava se da robot može obaviti sva gibanja potrebna kako bi se izvršile sve predviđene obrade na predmetu. Programiranje robota vrši se pomoću operatorskog panela ili preko programskog paketa *RobotStudio*. Za potrebe ovog rada, navedeni je robot korišten u simulaciji za obradu ispitnih uzoraka.



Slika 13 radni prostor robota ABB IRB 6660-205/1.9 [8]

3.2. ABB IRBP A500

ARCOPS robotska ćelija opremljena je okretno-nagibnim prigonom ABB IRBP A500 (slika 14). Ovaj uređaj namijenjen je za procese u kojima se obrađuju obratci velikih gabaritnih dimenzija. U navedenom slučaju, pomoću dvije rotacijske osi ovaj prigon omogućava željenu orijentaciju alata prema obratku. Maksimalna nosivost okretno-nagibnog prigona iznosi 500 kg. Uređaj koristi isti pogonski sustav i softver kao i roboti. Prilikom programiranja ili u toku rada sve se osi mogu koordinirano gibati s robotom [11].

Slika 14 ABB IRBP A500 [11]

3.3. Robot ABB IRB 4600-40/2.55

Za skeniranje uzoraka koristi se robot ABB IRB 4600-40/2.55 (slika 15) na kojem je montiran digitalizator ATOS 5X. Radi se robotu koji također ima šest stupnjeva slobode gibanja. Nosivost robota iznosi 40 kg, a dohvat 2.55 m. U tablici 2 dane su ostale specifikacije robota.

Tablica 2 Tablica 2 Specifikacije robota ABB IRB 4600-40/2.55 [12]

Broj stupnjeva slobode gibanja	6
Nosivost robota	40 kg
Masa robota	465 kg
Točnost ponavljanja	0,06 mm
Točnost pozicioniranja	0,02 mm
Dohvat	2,55 m

Slika 15 ABB IRB 4600-40/2.55 [12]

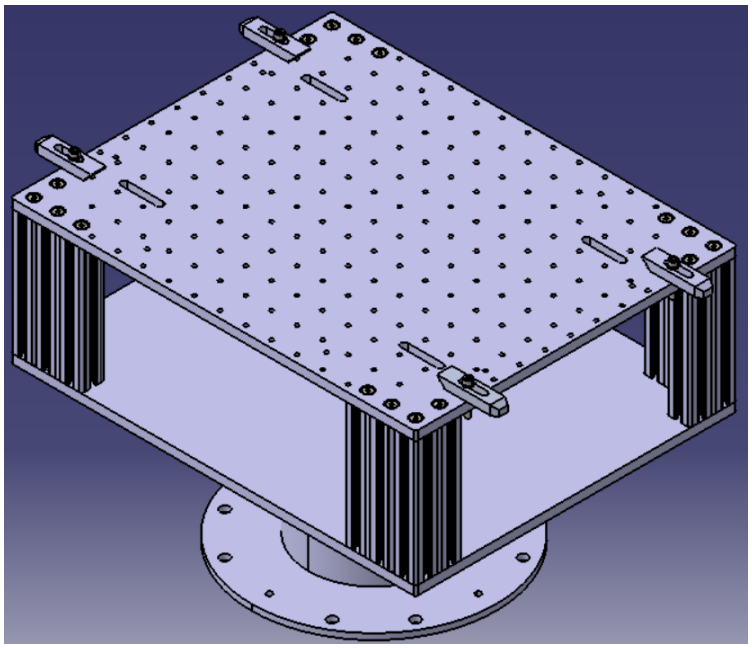
3.3.1. ATOS 5X

U ovoj robotskoj ćeliji, za potrebe skeniranja uzoraka koristi se mjerni sustav ATOS 5X (slika 16). ATOS 5X je beskontaktni 3D skener njemačkog proizvođača ZEISS koji se bazira na principu triangulacije. Uređaj koristi laserski izvor svjetla koji generira monokromatsku plavu svjetlost valne duljine 440 nm do 470 nm koja je rijetko prisutna u prirodi. Kako bi se smanjio utjecaj ambijentalne svjetlosti na kvalitetu skeniranja, uređaj ima ugrađene filtere koji propuštaju samo valne duljine monokromatske plave svjetlosti. Izmjenom odgovarajućih parova objektiva mjerni se volumen može mijenjati u rasponu od 320x240x240 mm^3 do 1000x750x750x mm^3 [13].

Slika 16 ATOS 5X [14]

3.4. Stezna naprava

Na slici 17 nalazi se stezna naprava za prihvat ispitnog uzorka. Na njoj se nalaze četiri čeljusti pomoću kojih se stežu uzorci. Kada se uzorak postavi na steznu napravu i stegne pomoću čeljusti, provrti s navojima na gornjoj ploči omogućuju deformiranje ispitnog uzorka pritezanjem vijaka s donje strane ploče.



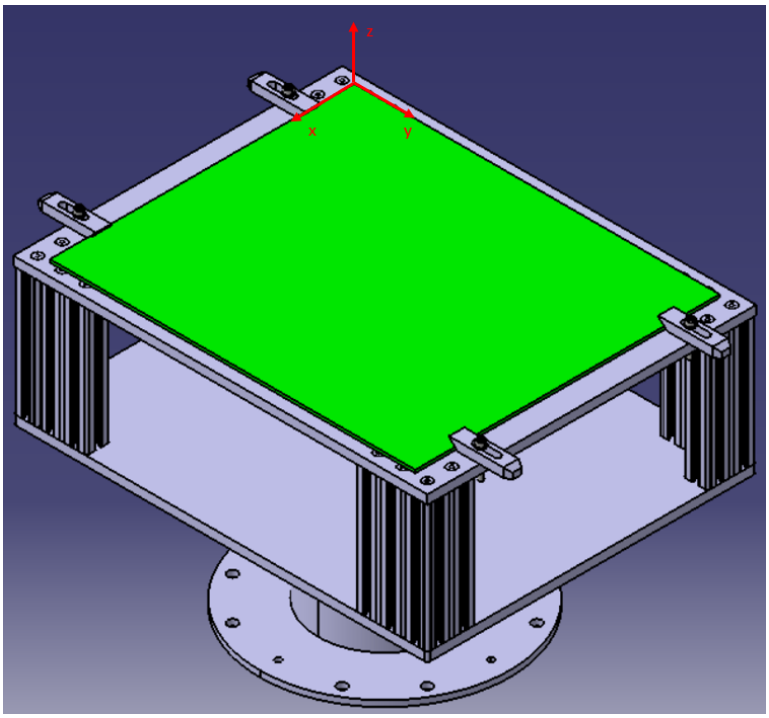
Slika 17 CAD model stezne naprave

4. IZRADA REFERENTNOG CAD MODELA, PUTANJE ALATA I ISPITNIH UZORAKA

Zbog trenutne izvanredne situacije na fakultetu nije bilo moguće napraviti stvarne obrade i mjerenja na deformiranim uzorcima. Iz tog razloga izvršena je simulacija procesa koristeći ciljano deformirane modele pripremljene pomoću programskog paketa Catia V5R21. Nakon što su uzorci pripremljeni, spremljeni su u obliku .stl datoteke i korišteni su umjesto rezultata skeniranja uzoraka.

4.1. Referentni CAD model

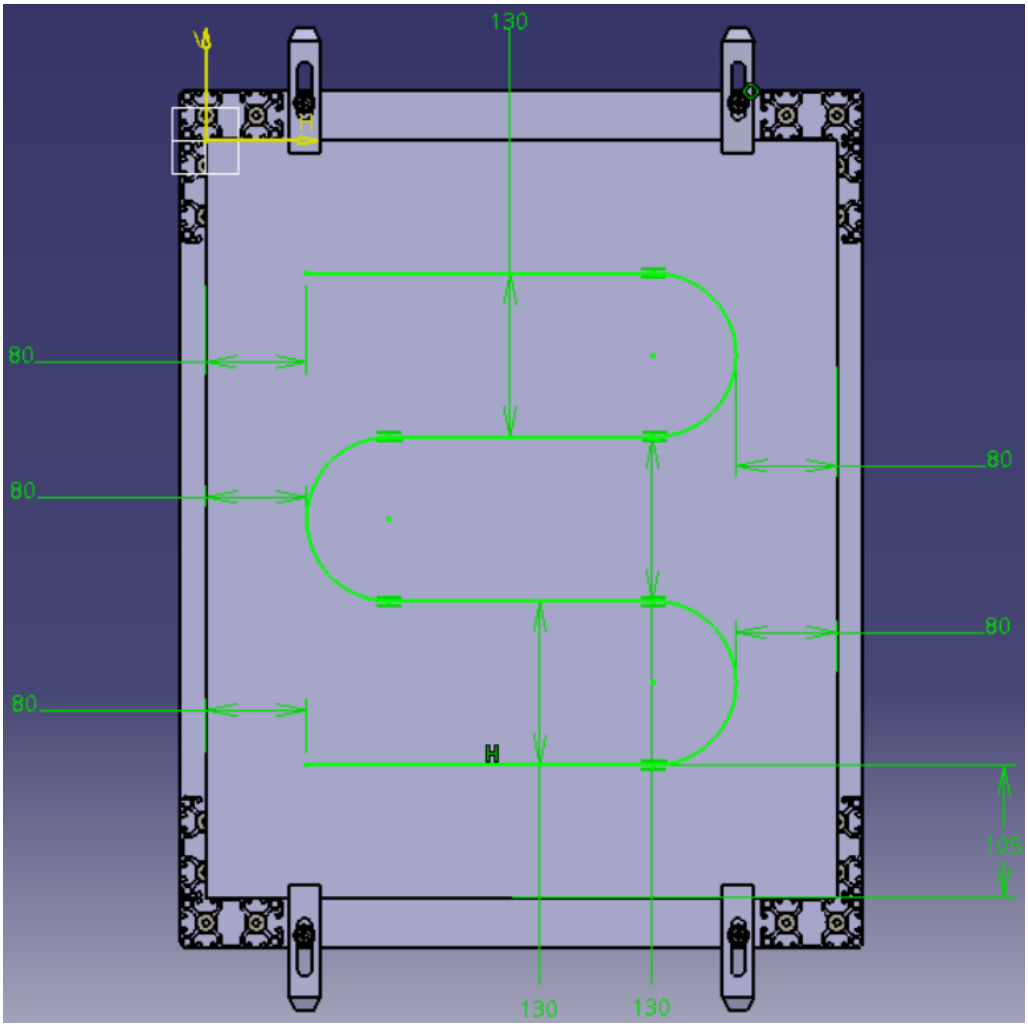
Slika 18 pokazuje referentni CAD model. CAD model se sastoji od stezne naprave i ploče dimenzija 500x600x5 mm. Navedeni CAD model napravljen je u softveru Catia V5R21. Središte koordinatnog sustava nalazi se na gornjoj strani referentne (nedeformirane) ploče. Koordinatni sustav je definiran na ovaj način kako bi se kasnije olakšala analiza deformacija u softveru *GOM Inspect Pro*. Kako bi se dobio koordinatni sustav kao na slici, prilikom stvaranja sklopa najprije je bilo potrebno učitati referentnu ploču i nju fiksirati. Nakon toga u sklop je dodana stezna naprava koja je poravnata u odnosu na referentnu ploču. Pomoću ovog sklopa kasnije je napravljena putanja alata (poglavlje 4.2.) kao i ispitni uzorci (poglavlje 4.3.). Sva odstupanja uzoraka mjeriti će se u odnosu na referentnu ploču (zeleno obojani predmet u sklopu).



Slika 18 Referentni CAD model

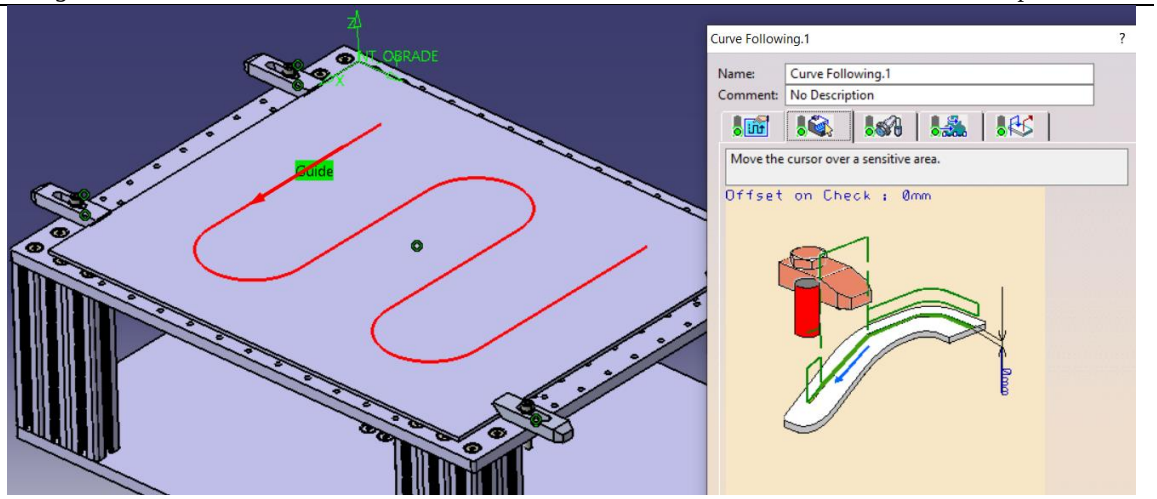
4.2. Putanja alata

Nakon što je pripremljen referentni CAD model slijedila je izrada putanje alata koja je također generirana u softveru Catia V5R21. U izborniku *Start* odabire se *Machining* i zatim *Prismatic Machining*. U otvorenom izborniku referentna ploča se odabire kao obradak. Kao nul-točka obrade odabire se središte koordinatnog sustava sklopa (slika 18). Ostale opcije nisu mijenjane.



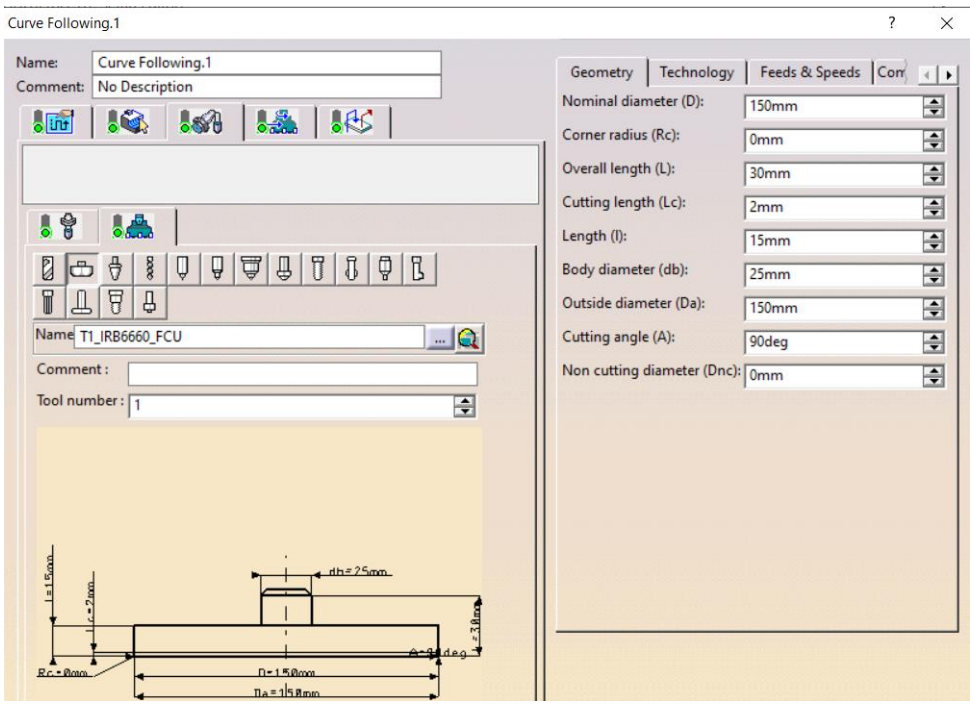
Slika 19 Putanja alata

Kako bi se putanja alata pojednostavila, odlučeno je da će se koristiti proizvoljna krivulja po kojoj će se alat kretati. Slika 19 pokazuje tu krivulju zajedno s pripadajućim kotama. Prilikom konstruiranja krivulje uzeto je u obzir da će za obradu biti korišten brusni papir promjera 150 mm, pri čemu će alat prilikom praćenja krivulje, čitavom površinom brusa biti u kontaktu s površinom ploče tijekom cijelog procesa brušenja. Važno je napomenuti da su prva i posljednja linija gibanja udaljene od ruba ploče za 105 mm, čime se osigurava da ne dođe do sudara alata sa steznim čeljustima.



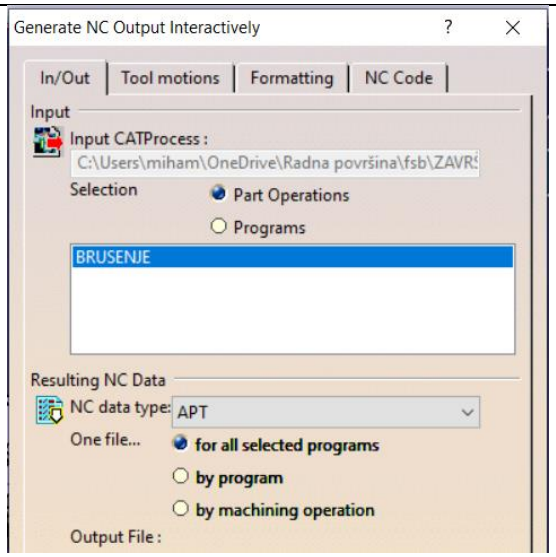
Slika 20 Smjer gibanja alata

Kada je krivulja nacrtana u izborniku se odabire *Insert*, zatim *Machining Operations*, te *Curve Following* i odabire se nacrtana krivulja. U ovom koraku odabran je smjer kojim će se alat gibati (slika 20). Alat ulazi u zahvat s one strane krivulje koja je najbliža nul-točki obrade. U otvorenom izborniku definiran je i alat. Slika 21 prikazuje alat s pripadajućim dimenzijama. Jedina vrijednost koja je promijenjena je promjer alata (150 mm).



Slika 21 Dimenzije alata

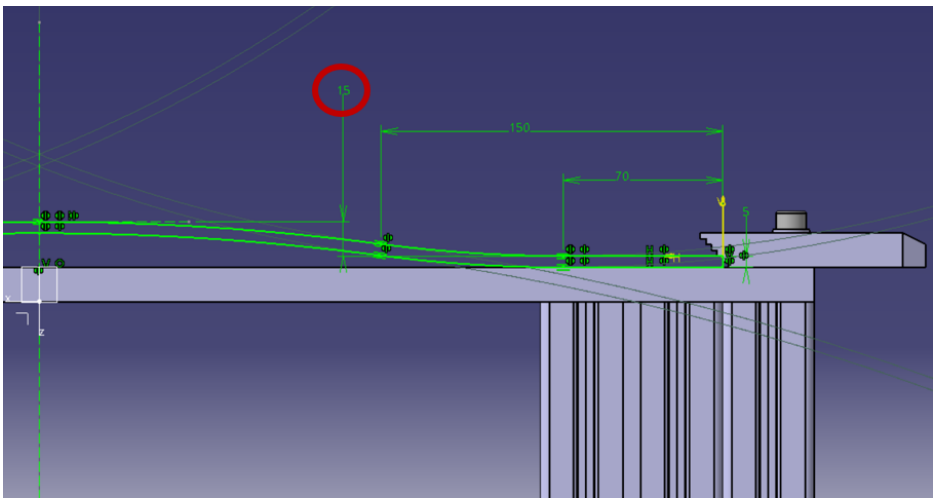
Nakon što je putanja alata simulirana, spremljena je kao APT kod. To je napravljeno na način da se u izborniku odabrala opcija *Generate NC Code Interactively*, te je kao *NC Data Type* odabran APT (slika 22). APT kod će se kasnije koristiti za generiranje RAPID koda (poglavlje 5).



Slika 22 Generiranje APT koda

4.3. Ispitni uzorci (simulirani rezultati digitalizacije)

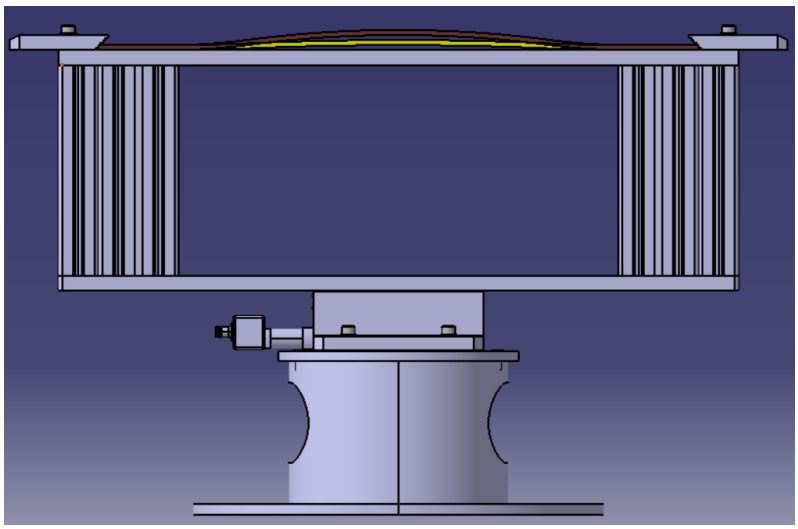
Za izradu uzoraka, kao baza korišten je referentni CAD model (poglavlje 4.1.) kako bi se na taj način poštivao koordinatni sustav stezne naprave, uzoraka i idealne ploče. Referentni CAD model koji je zapravo sklop stezne naprave i referentne ploče najprije je pretvoren u jedan element (*Part*). To je postignuto na način da se u stablu označio sklop, a zatim se u izborniku klikom na *Insert* odabrala opcija *Generate CATPart from Product*. Slika 23 prikazuje na koji način je definiran ispitni uzorak. Ispitni je uzorak definiran tako da se promjenom jedne dimenzije (zaokruženo) može definirati najveće odstupanje uzorka s obzirom na referentnu ploču. Vrijednosti 150 mm i 70 mm odabrane su proizvoljno, dok vrijednost 5 mm odgovara debljini stjenke ispitnog uzorka.



Slika 23 Crtež ispitnog uzorka

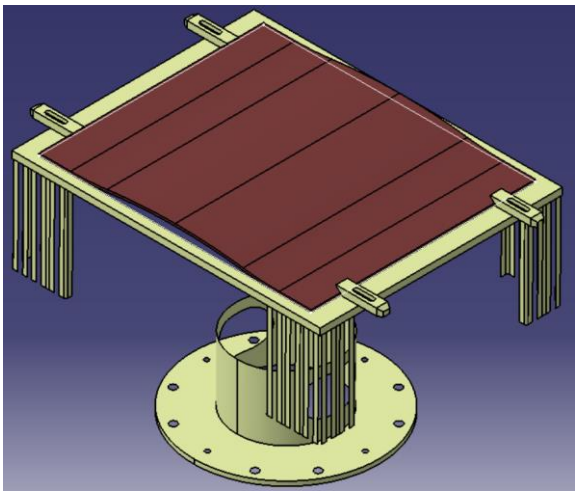
Iz slika 23 i 24 može se vidjeti kako ploča na lijevoj i desnoj strani nije zaobljena već ravna. Na svakoj ploči je ostavljeno (70 x 500) mm ravne plohe sa svake strane kako bi se kasnije lakše poravnali ispitni uzorci i referentna ploča u softveru *GOM Inspect Pro* (poglavlje 6.1.2.).

Na ovaj način omogućena je izrada uzoraka različitih odstupanja u vrlo kratkom roku. Slika 24 pokazuje dva ispitna uzorka s različitim odstupanjima (5 mm i 15 mm).



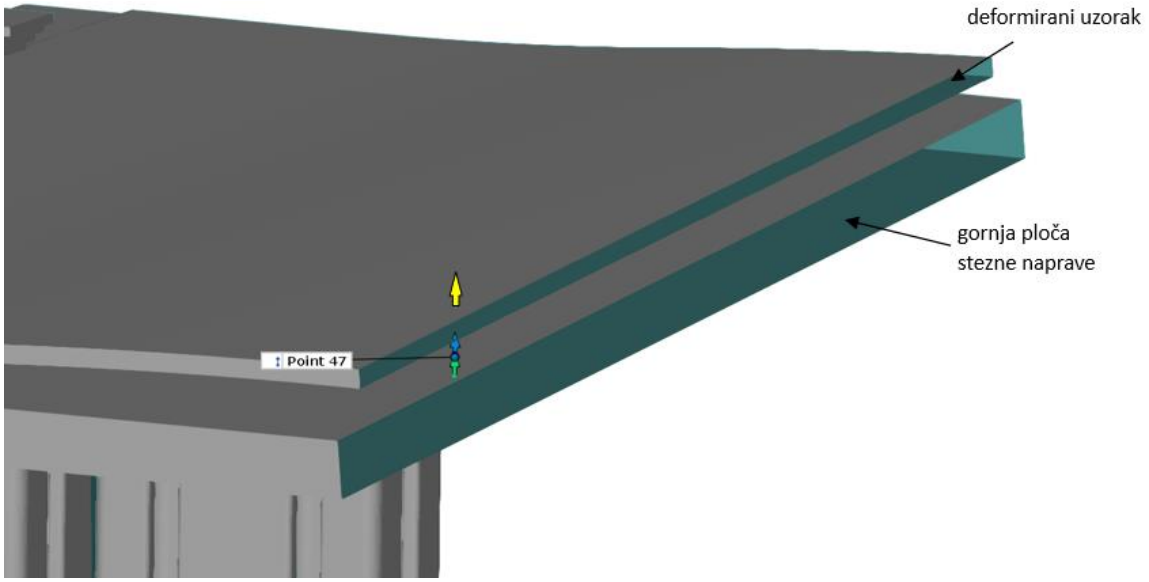
Slika 24 Uzorci s različitim odstupanjima

Kako bi uzorci formirani CAD softverom uistinu simulirali uzorke koji bi se dobili 3D skeniranjem predmeta, potrebno je izdvojiti njihove vanjske površine kao i površine stezne naprave koje bi 3D skener detektirao i skenirao. U tu svrhu korištena je naredba *Extract*. Također, pretpostavljeno je da skener neće moći vidjeti površinu stezne naprave koja se nalazi ispod uzorka te je stoga ona uklonjena pomoću naredbe *Split*. Svaki je izrađeni uzorak na kraju spremljen u obliku .stl datoteke, te je korišten za mjerenje odstupanja. Primjer uzorka koji se koristio za mjerenje odstupanja dan je na slici 25.



Slika 25 Primjer uzorka korištenog u mjerenju odstupanja

U slučaju kada bi se sklop stezne naprave s deformiranim uzorkom samo spremio kao .stl datoteka postoji mogućnost da dođe do pogrešnih očitanja odstupanja. Na slici 26 plava strelica predstavlja vektor normale površine na idealnom CAD modelu (koji je zbog preglednosti slike sakriven). Žuta strelica je položaj vektora normale koji se očekuje da će se dobiti prilikom analize, a zelena strelica je vektor normale koji se dobije korištenjem naredbe *Measuring Principle: Intersection With Mesh.* U tom slučaju softver stvara točku (s pripadajućim vektorom normale) na steznoj napravi, a ne na deformiranom uzorku.



Slika 26 Pogrešno izmjereno odstupanje

5. GENERIRANJE POČETNOG RAPID KODA

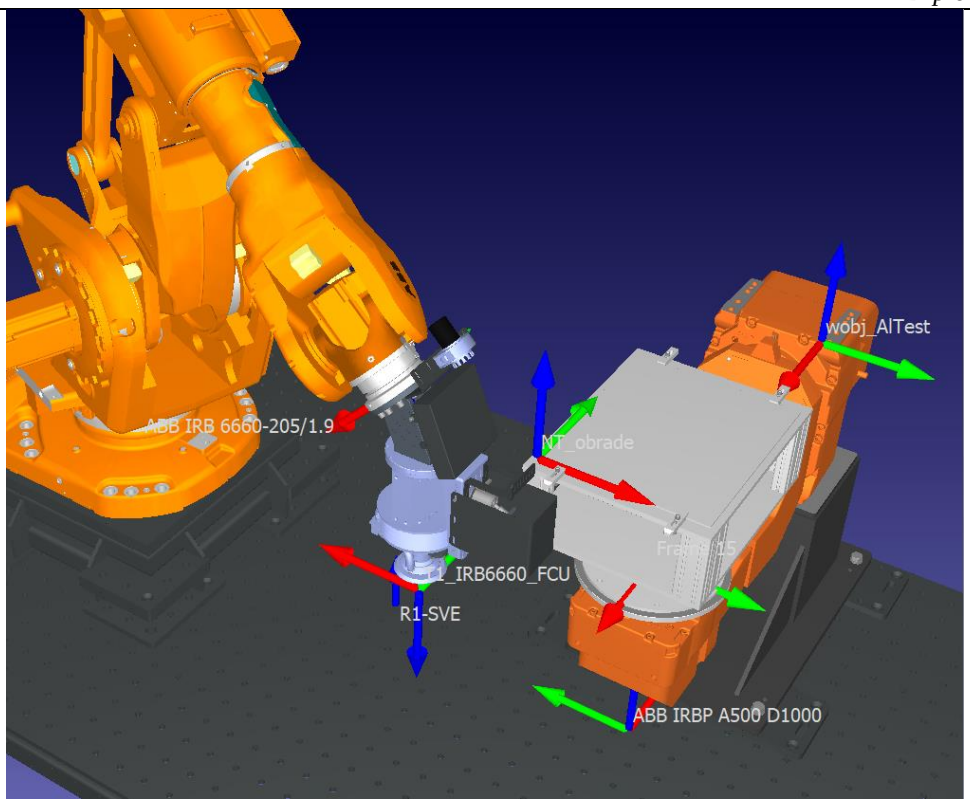
Nakon što je pripremljen referentni CAD model i uz njega vezana putanja alata, slijedilo je generiranje početnog RAPID koda. Kao što je navedeno u poglavlju 3.1., za simuliranje gibanja alata prilikom obrade korišten je ABB-ov robot IRB 6660-205/1.9., dok je za generiranje RAPID koda korišten programski paket *RoboDK*.

5.1. RoboDK

U radu je korišten model robotske ćelije koja je prethodno izrađena u sklopu projekta ARCOPS u programskom paketu *RoboDK*. *RoboDK* je programski paket koji služi za simulaciju gibanja robota te je neovisan o proizvođaču robota. Budući da *RobtStudio* nema mogućnost generiranja RAPID koda iz nekog drugog (npr. APT ili G), u ovome je radu *RoboDK* korišten kako bi se iz APT koda generiranog u CAD/CAM sustavu Catia V5R21 generirao RAPID kod. Zbog nemogućnosti izrade virtualnog kontrolera postoji mogućnost da kretanje robota u simulaciji neće u potpunosti odgovarati gibanju robota u stvarnom procesu. Iz tog razloga za konačno simuliranje obrade koristio se *RobotStudio*, a *RoboDK* je korišten samo za generiranje početnog RAPID koda.

U dobivenoj ćeliji na okretno-nagibnom stolu nije bilo stezne naprave. Odlučeno je da će se u robotsku ćeliju na okretno-nagibni stol postaviti referentni CAD model koji uz steznu napravu sadrži i ispitni uzorak. U programu Catia V5R21 navedeni CAD model je spremljen kao datoteka u STEP formatu te je ona učitana u *RoboDK*. Središte koordinatnog sustava učitanog modela je u središtu osi stezne naprave. Nakon učitavanja u *RoboDK*, stezna naprava s referentnim modelom postavljena je u odgovarajući koordinatni sustav (*Frame 15)* koji se nalazi na okretno nagibnom stolu. Steznu napravu je bilo potrebno rotirati oko vlastite Z osi za 90 stupnjeva kako bi se dobila njena pravilna orijentacija u odnosu na robot.

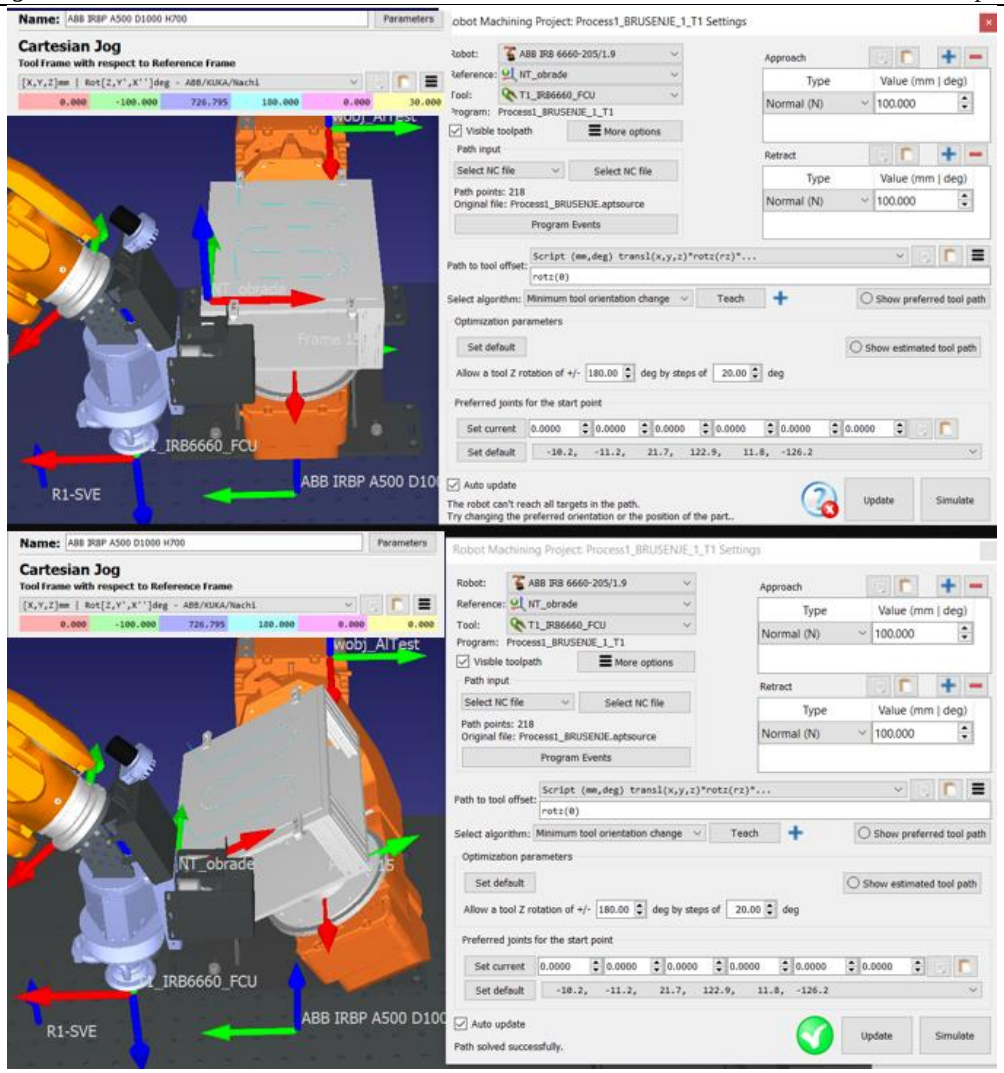
Idući korak bio je definirati nul-točku obrade. Najprije je potrebno odabrati *Frame 15* te unutar njega stvoriti novi koordinatni sustav. Na taj način novi koordinatni sustav (NT_obrade) mijenjati će se u ovisnosti o *Frame 15* (npr. ukoliko se stol na kojem se nalazi stezna naprava zarotira za 90 stupnjeva, zarotirat će se i koordinatni sustav NT_obrade). Pomoću CATIA-e izmjerene su vrijednosti koordinata na kojima se nalazi vrh ploče koji je odabran kao nul-točka obrade. Te vrijednosti su unesene u *RoboDK* i na taj način je postavljena nul-točka obrade (slika 27).



Slika 27 Pravilno orijentirana stezna naprava s referentnom pločom i definirana nul-točka obrade

Kada je definirana nul-točka obrade u program je učitan ranije generirani APT kod (poglavlje 4.2). Nakon što je kod učitan, otvara se izbornik u kojem je potrebno odrediti robot koji će izvršiti zadatak (ABB IRB 6660-205/1.9), nul-točku obrade (NT_obrade) te alat. Kada je sve definirano program javlja da robot ne može dohvatiti sve točke obrade (slika 28). Mijenjanjem nagiba stola uočeno je da pri nagibu od 14 stupnjeva robot može dohvatiti sve točke obrade. Budući da se rad bavi korekcijom putanje alata te se očekuje da će doći do promjena u položaju i nagibu alata, odlučeno je da će se nagib stola povećati na 30 stupnjeva. Na taj način ostavlja se mogućnost promjene nagiba alata a da pritom robot i dalje može dohvatiti svaku točku obrade.

Kada je pokrenuta simulacija i kada je verificirano da radi na željeni način, potrebno je APT kod pretvoriti u RAPID kako bi se simulacija mogla isprobati u *RobotStudio*-u. Desnim klikom miša na proces u stablu otvara se izbornik u kojem se odabire *Generate Robot Program* te se proces sprema u datoteku koja sadrži kod u programskom jeziku RAPID.

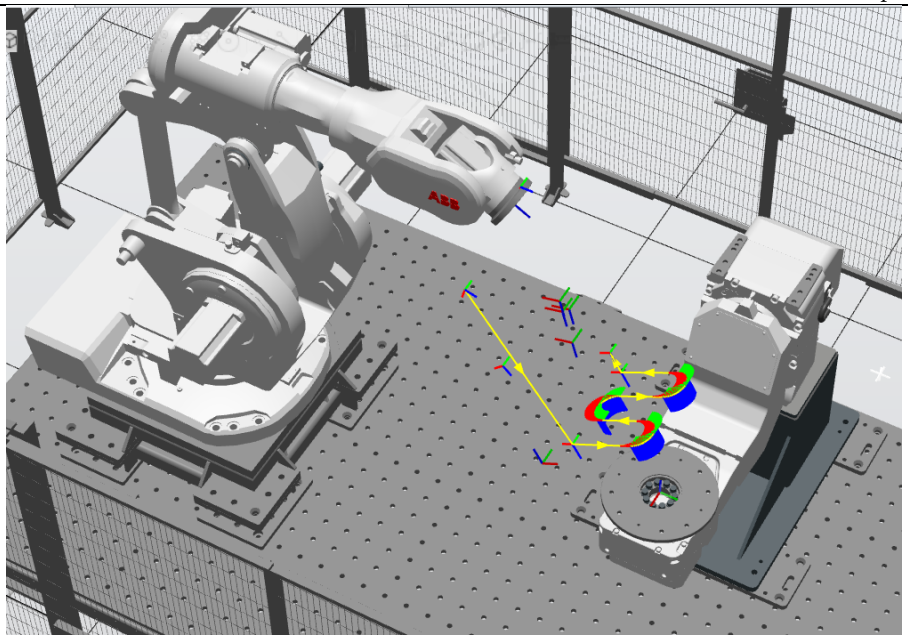


Slika 28 Promjena orijentacije stezne naprave s referentnom pločom

5.2. RobotStudio

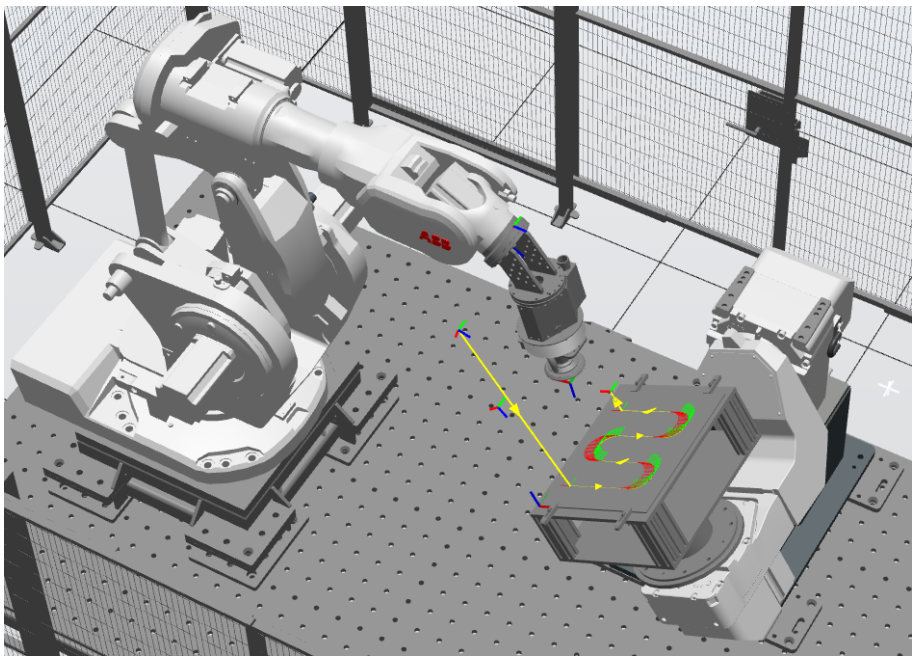
RobotStudio je softver za simuliranje i *off-line* programiranje ABB-ovih robota koji koriste RAPID kod. Za razliku od *RoboDK*-a u ovom softveru moguća je izrada virtualnog kontrolera. Virtualni kontroler se ponaša jednako kao i stvarni, a to znači da ono gibanje robota koje se vidi u simulaciji na računalu odgovara gibanju koje bi robot izvodio u stvarnosti. Upravo iz tog razloga, svaka je simulacija verificirana u ovome programu.

Na početku rada dobivena je robotska ćelija u *RobotStudio*-u koju je također trebalo modificirati. Prvi korak koji je bilo potrebno napraviti jest učitati sistemski modul. Sistemski se moduli koriste za definiranje podataka i rutina specifičnih za sustav, a nisu u užem smislu povezani s tehnološkim procesom [15]. Nakon što je učitan sistemski modul, u softver je učitan i programski modul koji sadrži putanju za obradu ispitnog uzorka.



Slika 29 Prikaz putanje alata u *RobotStudio*-u

Slika 29 prikazuje putanju alata u *RobotStudio*-u. U stanicu je učitan i 3D model alata koji je *drag and drop*-om postavljen na robot (ABB IRB 6660-205/1.9). Nakon toga u stanicu je učitan 3D model stezne naprave s nedeformiranim ispitnim uzorkom te je isti pridružen okretno nagibnom stolu. Steznu napravu bilo je potrebno zarotirati oko vlastite Z osi kako bi ona bila pravilno orijentirana u odnosu na robot. Posljednji korak koji je bilo potrebno napraviti jest zarotirati okretno nagibni stol za 30 stupnjeva kao što je to bio slučaj i u *RoboDK*-u. Krajnji rezultat dan je na slici 30.



Slika 30 Simulacija obrade referentne ploče

Na slici 30 prikazana je putanja alata za referentnu (idealnu) ploču. Budući da je tema rada korekcija putanje alata prema odstupanju oblika pripremka, u RAPID kodu bilo je potrebno izdvojiti točke putanje alata u kojima je alat u zahvatu s obratkom. U tim točkama korigirat će se putanja alata. Prvi korak u tom postupku bio je dodavanje komentara ″!Pocetak″ i ″!Kraj″ na dio koda u kojem će se alat nalaziti u punom zahvatu s obrađivanom površinom ispitnog uzorka (slika 31). Linija koda ispod ″! Kraj″ predstavlja gibanje alata tokom kojeg on izlazi iz zahvata s obratkom (pomak po Z osi alata) stoga je ona izostavljena.

```
33        ! OPERATION NAME : Tool Change.1
34        ! Start generation of : Tool Change.1
35        ! TOOLCHANGEBEGINNING
36        ! TOOLCHANGEEND
37        ! End of generation of : Tool Change.1
38        ! OPERATION NAME : Curve Following.1
39        ! Start generation of : Curve Following.1
40        Mirka 70.0;
41        ! Show T1_IRB6660_FCU
42        MoveAbsJ [[-10.192300,-11.190500,21.683600,122.941000,11.751400,-126.161000],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
43        ! Pocetak
44        MoveL [[80.000,105.000,0.000],[0.00000000,0.00000000,1.00000000,0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
45        MoveL [[355.000,105.000,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
46        MoveL [[362.028,105.381,0.000],[0.00000000,-0.00000000,1.00000000,0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
47        MoveL [[368.973,106.520,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
48        MoveL [[375.755,108.403,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
49        MoveL [[382.293,111.008,0.000],[0.00000000,-0.00000000,1.00000000,0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
50        MoveL [[388.511,114.304,0.000],[0.00000000,-0.00000000,1.00000000,0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
51        MoveL [[394.336,118.254,0.000],[0.00000000,0.00000000,1.00000000,0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
52        MoveL [[399.700,122.810,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
53        MoveL [[404.541,127.920,0.000],[0.00000000,-0.00000000,1.00000000,0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
54        MoveL [[408.800,133.523,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
55        MoveL [[412.428,139.553,0.000],[0.00000000,0.00000000,1.00000000,0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
56        MoveL [[415.383,145.941,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
...
124       MoveL [[412.428,460.447,0.000],[0.00000000,-0.00000000,1.00000000,0.00000000],[0,2,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
125       MoveL [[408.800,466.477,0.000],[0.00000000,-0.00000000,1.00000000,0.00000000],[0,2,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
126       MoveL [[404.541,472.080,0.000],[0.00000000,-0.00000000,1.00000000,-0.00000000],[0,2,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
127       MoveL [[399.700,477.190,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000],[0,2,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
128       MoveL [[394.336,481.746,0.000],[0.00000000,-0.00000000,1.00000000,-0.00000000],[0,2,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
129       MoveL [[388.511,485.696,0.000],[0.00000000,0.00000000,1.00000000,0.00000000],[0,2,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
130       MoveL [[382.293,488.992,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000],[0,2,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
131       MoveL [[375.755,491.597,0.000],[0.00000000,-0.00000000,1.00000000,-0.00000000],[0,2,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
132       MoveL [[368.973,493.480,0.000],[0.00000000,0.00000000,1.00000000,0.00000000],[0,2,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
133       MoveL [[362.028,494.619,0.000],[0.00000000,-0.00000000,1.00000000,-0.00000000],[0,2,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
134       MoveL [[355.000,495.000,0.000],[0.00000000,-0.00000000,1.00000000,0.00000000],[0,2,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
135       MoveL [[80.000,495.000,0.000],[0.00000000,0.00000000,1.00000000,0.00000000],[0,2,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
136       ! Kraj
137       MoveL [[80.000,495.000,100.000],[0.00000000,-0.00000000,1.00000000,0.00000000],[0,2,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
138       ! End of generation of : Curve Following.1
139   ENDPROC
140
141 ENDMODULE
142
```

Slika 31 Izdvajanje točaka u RAPID kodu

6. FORMIRANJE MJERNOG POSTAVA

Za analizu odstupanja oblika pripremka koristio se softver *GOM Inspect Pro*. U nastavku poglavlja opisana su ručna i automatizirana metoda formiranja mjernog postava.

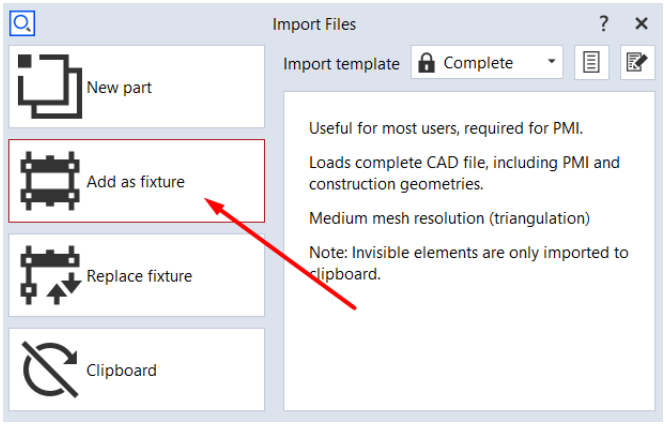
6.1. Ručno postavljenje

U softver je potrebno učitati dva modela od kojih će jedan biti postavljen kao referentni (*Nominal*) i on će predstavljati idealni predmet. Drugi model će predstavljati stvarni, realni predmet (*Actual*) te će na njemu biti izmjerena sva odstupanja. Za potrebe rada korišten je i treći model koji je predstavljao steznu napravu.

Kao referentni model koristio se CAD model nedeformiranog ispitnog uzorka, a kao stvarni predmeti koristili su se uzorci koji se sastoje od deformirane ploče i dijelova stezne naprave (slika 25).

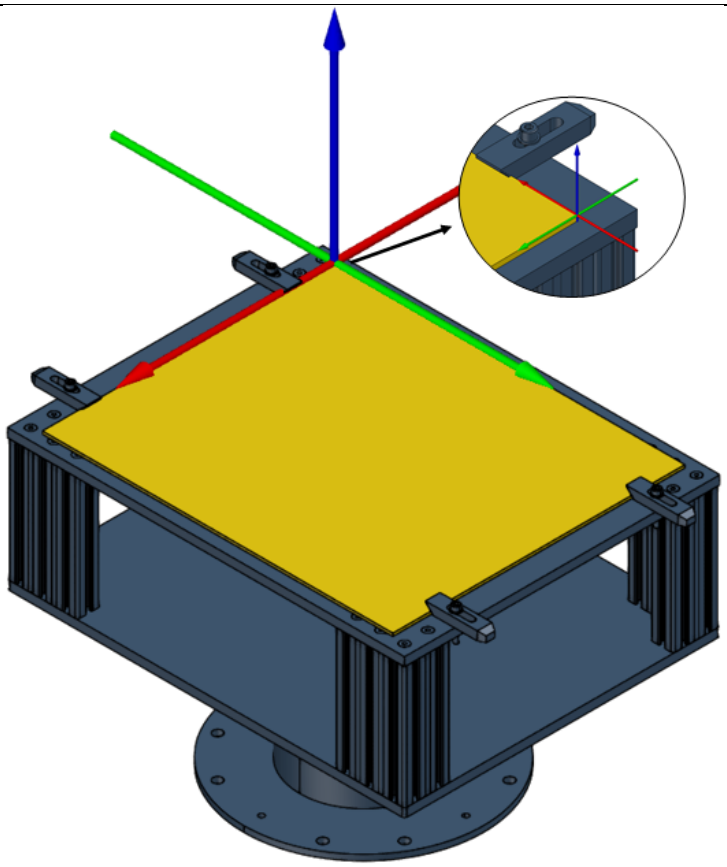
6.1.1. Učitavanje stezne naprave, referentne ploče i ispitnog uzorka u GOM Inspect Pro

Kako bi se navedeni modeli učitali u *Gom Inspect Pro*, potrebno je na alatnoj traci odabrati *File / Import* i zatim odabrati željenu datoteku. Prva datoteka koja se učitala u softver bila je stezna naprava. Prilikom učitavanja stezne naprave odabrana je opcija *Add as fixture* (slika 32).



Slika 32 Opcija koja omogućava da se učitana datoteka prikazuje kao stezna naprava

Zatim je u softver učitan nedeformirani ispitni uzorak te je u programu zadano da isti predstavlja *CAD body* (*Nominal*). Dobiveni sklop predstavlja idealni CAD model (slika 33). Nakon toga u program je učitan deformirani ispitni uzorak. Koordinatni sustav nedeformiranog ispitnog uzorka definiran u CAD/CAM sustavu Catia V5R21 ujedno predstavlja i globalni koordinatni sustav u formiranom mjernom postavu. Iz toga slijedi da se globalni koordinatni sustav preklapa s nul-točkom obrade.

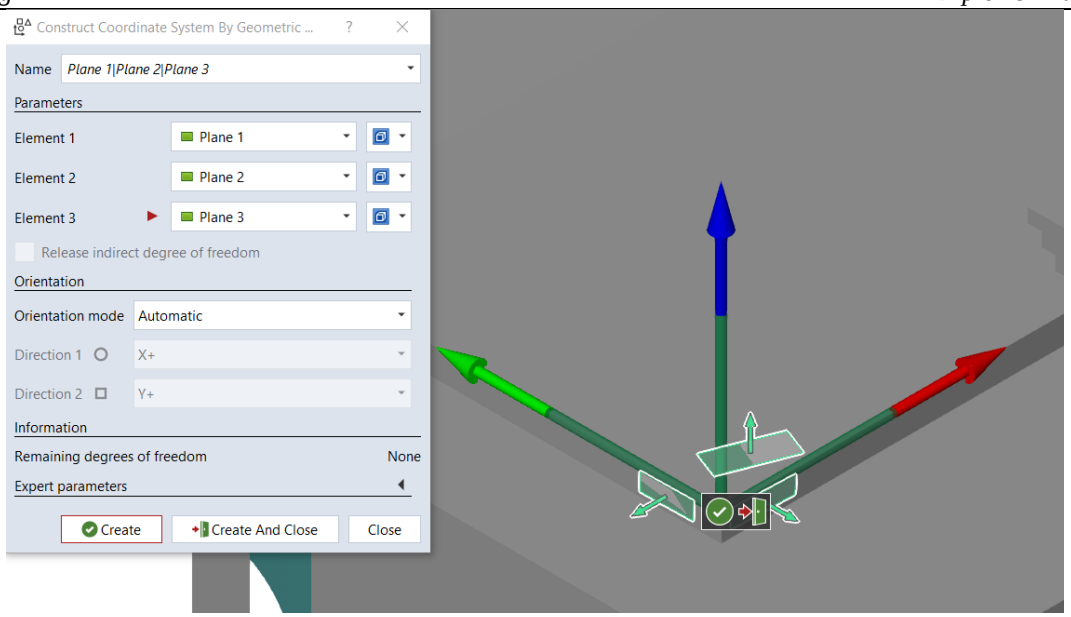


Slika 33 Stezna naprava i referentna ploča

6.1.2. Poravnanje

Ovaj je korak vrlo bitan kako bi se osiguralo da dobivena odstupanja odgovaraju isključivo odstupanju na deformiranom modelu. Za glavno poravnanje koristila se naredba *Alignment By Cordinate Systems.* Kako bi se ovo poravnanje moglo izvršiti, na deformiranom ispitnom uzorku najprije je potrebno definirati koordinatni sustav.

Koordinatni sustav je definiran pomoću tri ravnine. Svaka je ravnina konstruirana pomoću tri točke. Prilikom konstruiranja ravnina potrebno je voditi brigu o tome da su točke odmaknute jedna od druge (izbjegavati koncentriranje točaka na neku malu površinu) kako bi se dobila ravnina koja dovoljno točno predstavlja neku plohu skeniranog predmeta. Kada su konstruirane ravnine moguća je izrada koordinatnog sustava (slika 34).

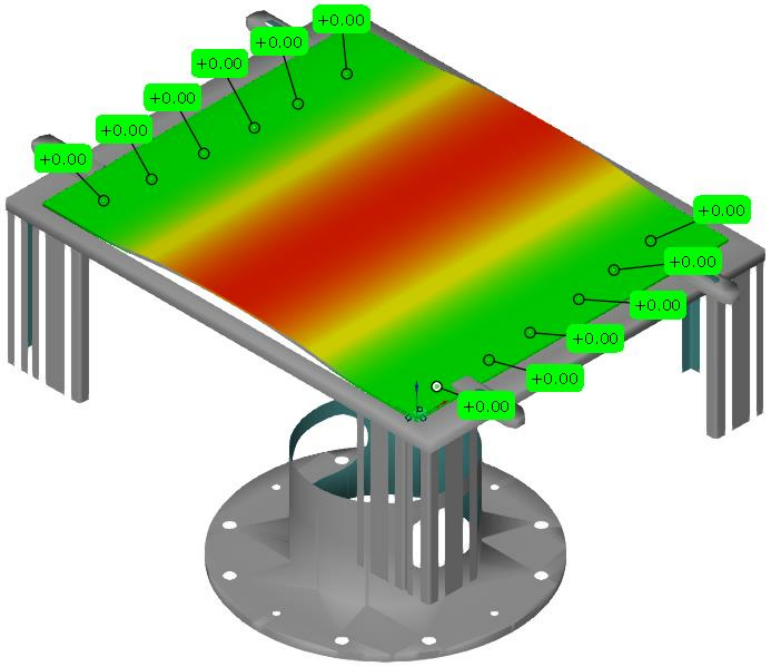


Slika 34 Koordinatni sustav deformiranog ispitnog uzorka

Konstruirani koordinatni sustav na uzorku se nalazi na istom mjestu gdje je smješten i globalni koordinatni sustav. Kada je konstruiran koordinatni sustav uzorka, u izborniku se odabire *Operations/ Alignment / Main Alignment / By Coordinate Systems* kako bi se konstruirani koordinatni sustav poravnao s globalnim.

Cilj ovog poravnanja bio je postići da je dio digitaliziranog (skeniranog) objekta koji predstavlja steznu napravu idealno poravnat (nema odstupanja) sa steznom napravom učitanom u softveru, a da se sva odstupanja skena od CAD modela manifestiraju na ploči. U ovom, simuliranom, slučaju to je moguće dok bi se u stvarnom slučaju nastojalo dobiti što bolje preklapanje idealnog i skeniranog modela.

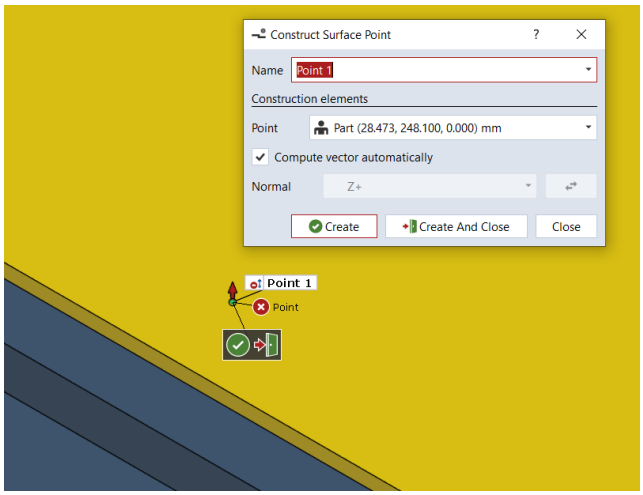
Kako bi se to provjerilo uključena je opcija *Surface Comparison On Actual.* Ova opcija omogućava kvalitetnu i brzu kontrolu odstupanja. Na vizualan način omogućava operateru da vrlo brzo uoči područja koja su izvan nekih zadanih tolerancija. Osim vizualnog prikazivanja grešaka, greške se mogu i izmjeriti pomoću *„Deviation labels“*. Kada je navedena naredba aktivirana, na deformiranoj ploči postavljene su naljepnice za odstupanje kako bi se dobila brojčana vrijednost eventualne greške (slika 35). Iz slike 35 se može vidjeti da ravna područja deformiranog ispitnog uzorka (slika 23) ne odstupaju u odnosu na idealni model uzorka. Jedina odstupanja koja se pojavljuju su ona na dijelu ploče koji je deformiran. Na ovaj način je potvrđena ispravnost poravnanja.



Slika 35 Provjera poravnanja

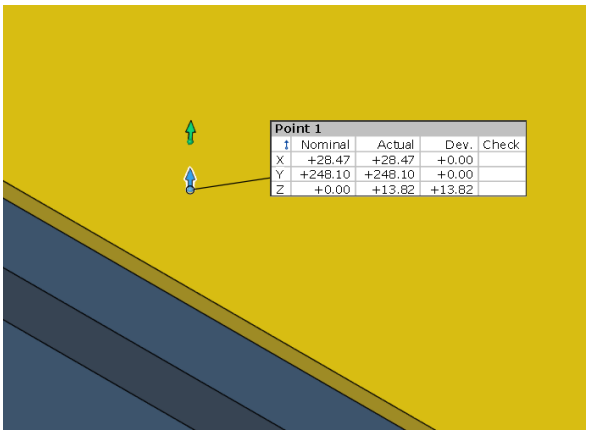
6.1.3. Konstruiranje točaka na površinu CAD modela i uzorka

Nakon što su deformirani i referentni CAD model poravnati na željeni način, idući je korak konstruiranje točaka iz putanje alata prema NC programu. U tu svrhu korištena je naredba *Construct Surface Point* koja konstruira točku zajedno s pripadajućim vektorom normale (Slika 36). U izborniku se nudi opcija da softver sam odredi smjer tog vektora ili da mu se on ručno zada (npr, u pozitivnom smjeru osi z globalnog koordinatnog sustava). Za potrebe ovoga rada ostavljeno je da softver sam odredi smjer normale. Točka je postavljena na CAD modelu (*Nominal*).



Slika 36 Točka s pripadajućim vektorom normale na idealnoj ploči

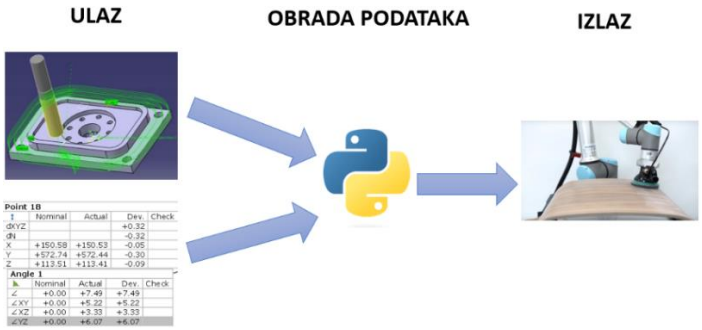
Kako bi se ta točka s pripadajućim vektorom normale preslikala na deformirani uzorak, ranije stvorenoj točki potrebno je definirati *Measuring Principle*. U radu je korišten *„Measuring Principle: Intersection With Mesh“*. Slika 37 prikazuje konstruiranu točku na deformiranom ispitnom uzorku s pripadajućim vektorima smjera (zeleno). Na slici se također vidi i odstupanje točke na uzorku u sve tri osi. Do tih podataka dolazi se na način da se pomoću tipke *Control* na tipkovnici i desnog klika miša otvara izbornik u kojem je potrebno odabrati koji se podaci žele prikazati.



Slika 37 Točka na deformiranom ispitnom uzorku

6.2. Automatizirano postavljanje mjernog sustava

GOM Inspect Pro na temelju ulaznih podataka kao što su CAD model idealnog oblika uzorka i digitaliziranog uzorka izračunava odstupanja koja predstavljaju neku izlaznu vrijednost. Zamisao ovoga rada je da se stvori sustav koji bi na osnovi učitanih točaka iz NC programa za obradu idealnog dijela te podataka o odstupanjima pripremka iz *GOM Inspect Pro*-a korigirao putanju alata s obzirom na izmjerena odstupanja (Slika 38). Za obradu podataka korišten je programski jezik *Python*. Prije nego se krenulo u izradu koda, u sustav je dodana *numpy* biblioteka za programski jezik *Python* koja dodaje podršku za rad s vektorima i matricama [16].



Slika 38 Povezivanje podataka iz CAM i GOM sustava s robotom

6.2.1. Automatiziranje poravnanja idealnog i deformiranog modela uzorka

Prvi korak koji se automatizirao bilo je poravnanje. *GOM Inspect Pro* ima mogućnost snimanja korisničkih operacija u niz naredbi (eng. macro) koje se generiraju u obliku skripte u programskom jeziku *Python.* Uređivanjem snimljene skripte ona se može prilagoditi drugim zadacima [17]. Kako bi se automatiziralo poravnanje, uključeno je snimanje naredbi te je ponovljen postupak opisan u poglavlju 6.1.2. Krajnji rezultat prikazan je na slici 39. Funkcijom *gom.script.primitive.create_plane_by_3_ points* konstruira se ravnina (pomoću tri točke), a funkcijom *gom.script.cs.create_element_by_geometric*_elements pomoću konstruiranih ravnina stvara se koordinatni sustav. Funkcijom u retku 38 definira se poravnanje prema koordinatnim sustavima.

```
2
3   import gom
4   gom.script.cad.hide_element (elements=[gom.app.project.parts['Part'].nominal])
5
6   gom.script.cad.show_element (elements=[gom.app.project.parts['Part'].actual])
7
8   MCAD_ELEMENT=gom.script.primitive.create_plane_by_3_points (
9       name='Plane 1',
10      point1={'interpolated': True, 'normal': gom.Vec3d (-1.0, 4.130759879e-15, 0.0), 'point': gom.Vec3d (-6.323830348e-13, 15.92296948, -3.491071113), 'target': gom.app.proje
11      point2={'interpolated': True, 'normal': gom.Vec3d (-1.0, 4.130759879e-15, 0.0), 'point': gom.Vec3d (-6.536993169e-13, 8.387984196, -0.6832540947), 'target': gom.app.proj
12      point3={'interpolated': True, 'normal': gom.Vec3d (-1.0, 4.130759879e-15, 0.0), 'point': gom.Vec3d (-6.75015599e-13, 4.241525165, -3.746964549), 'target': gom.app.projec
13      properties=gom.Binary ('eAHFmGuMVGcZx3+LFBBLkd6EhuhkWQWhszd2KQy0bLlJdaErpVAb7TiXs7tT2meGmVkWamgHjWm9pTFG25gYTcTWhATDF6uNpI2J0dQEYxqlokHL4kfpNbUREiTgvk/73l3zlz2gsY4hJ13z
14
15  MCAD_ELEMENT=gom.script.primitive.create_plane_by_3_points (
16      name='Plane 2',
17      point1={'interpolated': True, 'normal': gom.Vec3d (-4.110428123e-15, -1.0, 0.0), 'point': gom.Vec3d (6.106159775, -3.552713679e-14, -3.697141974), 'target': gom.app.proj
18      point2={'interpolated': True, 'normal': gom.Vec3d (-4.110428123e-15, -1.0, 0.0), 'point': gom.Vec3d (10.69751785, -4.263256415e-14, -1.152323003), 'target': gom.app.proj
19      point3={'interpolated': True, 'normal': gom.Vec3d (-4.110428123e-15, -1.0, 0.0), 'point': gom.Vec3d (20.30561366, -8.526512829e-14, -4.340321273), 'target': gom.app.proj
20      properties=gom.Binary ('eAHFmGuMlGcVx3+LQBFLcXuzNEYnClOEZm8sWxigbLlJ7UI3dAvYqONc3t2d7uzM7MzsLovBDmrSmrQhfrA1fmkTtW2yCYZPRVPSJ15iNMEPptq08sFLYqKYmn4QlBTM/zzvszOzM3tBYxzCz
21
22  MCAD_ELEMENT=gom.script.primitive.create_plane_by_3_points (
23      name='Plane 3',
24      point1={'interpolated': True, 'normal': gom.Vec3d (0.0, 0.0, 1.0), 'point': gom.Vec3d (7.481916564, 12.27457741, 0.0), 'target': gom.app.project.parts['Part'].actual},
25      point2={'interpolated': True, 'normal': gom.Vec3d (0.0, 0.0, 1.0), 'point': gom.Vec3d (9.617415668, 5.379808535, 0.0), 'target': gom.app.project.parts['Part'].actual},
26      point3={'interpolated': True, 'normal': gom.Vec3d (0.0, 0.0, 1.0), 'point': gom.Vec3d (15.22291823, 10.29211857, -7.105427358e-15), 'target': gom.app.project.parts['Part
27      properties=gom.Binary ('eAHFmFtsm2cZx38tXVdKu5LuwFpOVhpYWOecmqat2y1ZT3Rb2kVdlpYJsHz4knhxbNd2km2Q6jKJDQk0cbENcQGIw4YUqahXA78p3CAhJhUkBIxNu0BM44LCYBdsILVF/+f93tiOnUNBCFeNX
28
29  MCAD_ELEMENT=gom.script.cs.create_element_by_geometric_elements (
30      additional_constraints={'use_fixed_opening_angle': 'true'},
31      computation_mode='pure-geometry',
32      datum_1_elements=[gom.app.project.actual_elements['Plane 1']],
33      datum_2_elements=[gom.app.project.actual_elements['Plane 2']],
34      datum_3_elements=[gom.app.project.actual_elements['Plane 3']],
35      inclusion=False,
36      properties=gom.Binary ('eAHFmE1sXFcVx38JaepGTVO3BZqA6MgxNLQ2e+w4bjJJSZo0JS1Ja7VpUiFg9Gbm2R4yX5k2x3FR6QQWFAmpYsGHugGJT8l8UFYFBAIWIGARFgioiroAIbEgqKgLWiE1Qf9z3/V782U7IMSzbl
37
38  CAD_ALIGNMENT=gom.script.alignment.create_by_csys (
39      actual_coordinate_system=gom.app.project.actual_elements['Plane 1|Plane 2|Plane 3'],
40      name_expression='Alignment by coordinate systems',
41      parent_alignment=gom.app.project.parts['Part'].original_alignment,
42      target_coordinate_system=gom.app.project.nominal_elements['system_global_coordinate_system'])
```

Slika 39 Skripta za automatiziranje poravnanja

6.2.2. Automatizirano stvaranje točaka na površini CAD modela

Idući korak bio je automatizirati stvaranje točaka s pripadajućim vektorima normala na idealnom CAD modelu ispitnog uzorka. Slika 40 prikazuje dio koda koji iz RAPID koda čita linije koje se nalaze između komentara „! Pocetak“ i „! Kraj“ (slika 31) te ih pomoću *.append* funkcije dodaje u listu 1stTockePutanjeZahvat. Te linije predstavljaju koordinate i orijentaciju alata tijekom kojeg je on u zahvatu s obratkom. Na samom početku koda definirana je lokacija datoteke početnog RAPID koda iz kojeg se izvlače točke putanje alata (varijabla FILE). Također, na početku koda uvezena je biblioteka u kojoj su definirane specifične funkcije koje će se pozivati u raznim fazama glavnoga koda (*libLeginRepalust*).

```
import gom
import libLeginRepalust
FILE="C:\DIPLOMSKI\kod\RobotStudio_RAPID_brusenje.mod"
STR_POCETAK="! Pocetak"
STR_KRAJ="! Kraj"

lstTockePutanjeZahvat=[]

with open(FILE) as infile:
    copy = False
    for line in infile:
        if line.strip() == STR_POCETAK:
            copy = True
            continue
        elif line.strip() == STR_KRAJ:
            copy = False
            continue
        elif copy:
            lstTockePutanjeZahvat.append(line)

```

Slika 40 Dio koda koji čita RAPID putanje od komentara "! Pocetak" do "! Kraj"

Slika 41 prikazuje dio koda koji izdvaja koordinate točaka iz putanje alata i postavlja ih na referentnu ploču. Na početku se ispisuje linija iz RAPID koda tako da se pritom uklone svi namjerno ili slučajno postavljeni razmaci na početku i na kraju linije (metoda *.strip*()).

```
point_nr=1
for line in lstTockePutanjeZahvat:
    print("\n")
    print("*"*80)
    print(line.strip())
    tocke=libLeginRepalust.parse_rapid_points(line)
    print("KOORDINATE:", tocke[0])
    print("KUTEVI_Q:", tocke[1])
    print("KUTEVI_RAD:", tocke[2])
    print("KUTEVI_DEG:", tocke[3])
    print("NORM KUTEVI", tocke[4])

    print("*" * 80)
    print("\n")


    koord_x = tocke[0][0]
    koord_y = tocke[0][1]
    koord_z = tocke[0][2]

    jed_vec=libLeginRepalust.quat2jv(tocke[1][0], tocke[1][1], tocke[1][2], tocke[1][3], "z")
    jed_vec_i = jed_vec[0]*-1.0
    jed_vec_j = jed_vec[1]*-1.0
    jed_vec_k = jed_vec[2]*-1.0

    MCAD_ELEMENT=gom.script.primitive.create_surface_point (
        name="Point %s" % (str(point_nr)),
        point={'interpolated': True, 'normal': gom.Vec3d (jed_vec_i, jed_vec_j,jed_vec_k), 'point': gom.Vec3d (koord_x, koord_y, koord_z), 'target': gom.a

    print("Added Point %s: (x,y,z)=%.8f %.8f %.8f  ANGLES: %.8f %.8f %.8f" % (str(point_nr), koord_x, koord_y, koord_z, jed_vec_i, jed_vec_j, jed_vec_k))
    point_nr+=1
```

```
********************************************************************************
MoveL [[80.000,105.000,0.000],[0.00000000,0.00000000,1.00000000,0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_
KOORDINATE: [80.0, 105.0, 0.0]
KUTEVI_Q: [0.0, 0.0, 1.0, 0.0]
KUTEVI_RAD: (0.0, 0.0, 3.141592653589793)
KUTEVI_DEG: [0.0, 0.0, 180.0]
NORM KUTEVI [0.0, 0.0, 1.0]
********************************************************************************
```

Slika 41 Dio koda koji izdvaja koordinate točaka iz putanje i postavlja ih na referentnu ploču

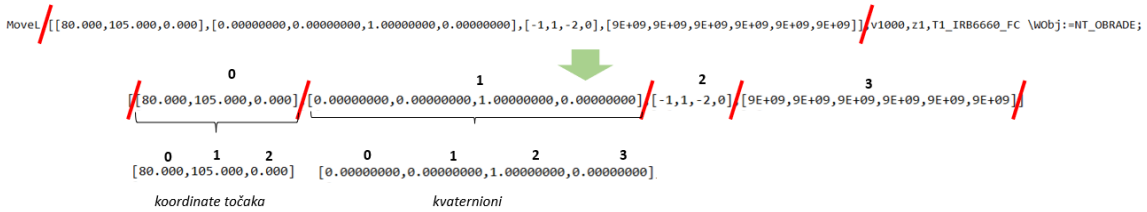
Zatim se iz biblioteke *libLeginRepalust* poziva funkcija *parse_rapid_points* (slika 42). Cilj ove funkcije je iz jedne linije RAPID koda izdvojiti koordinate točaka (vrha alata) i orijentaciju alata zapisanu pomoću kvaterniona.

```
def parse_rapid_points(rapidline):

    _sve_koordinate=find_substr2(rapidline, start_delim="[", end_delim="]")

    _sve_koordinate = _sve_koordinate.split("],[")
    _koordinate_tocaka=_sve_koordinate[0][1:].split(",")
    _kutevi_tocaka_quart=_sve_koordinate[1].split(",")

    _koordinate_tocaka = [float(tocka) for tocka in _koordinate_tocaka]
    _kutevi_tocaka_quart = [float(tocka) for tocka in _kutevi_tocaka_quart]

    _kutevi_tocaka_rad = euler_from_quaternion(_kutevi_tocaka_quart[0], _kutevi_tocaka_quart[1], _kutevi_tocaka_quart[2], _kutevi_tocaka_quart[3])
    _kutevi_tocaka_jed = [kut/math.pi for kut in _kutevi_tocaka_rad]
    _kutevi_tocaka_euler =  [math.degrees(kut) for kut in _kutevi_tocaka_rad]

    return (_koordinate_tocaka, _kutevi_tocaka_quart, _kutevi_tocaka_rad, _kutevi_tocaka_euler, _kutevi_tocaka_jed)
```

Slika 42 *parse_rapid_points*

Slika 43 prikazuje kako navedena naredba izdvaja koordinate točaka i kvaternione. Najprije se iz cijele linije RAPID koda izdvoje podaci između prve i posljednje uglate zagrade. Nakon

toga dobivena se linija podijeli u listu, a kao argument podijele zadali su se simboli ″],[″. U tako stvorenoj listi na prvom mjestu (0) nalaze se podaci o koordinatama točaka, a na drugom mjestu (1) nalaze se podaci o kvaternionima. Kako je podatke o koordinatama točaka potrebno odvojiti, ponovila se naredba *.split* ali se sada kao argument podjele definirao zarez. Postupak se ponavlja i za kvaternione. Na kraju dobiveni se podaci pretvaraju u realne brojeve (podatak tipa *float*).



Slika 43 *parse_rapid_points* na primjeru jedne linije RAPID koda

Nakon toga u glavnome kodu definirane su X, Y i Z koordinate točaka (linije od 37 do 39) na način da se one iščitavaju sukladno brojevima na slici 43. Nakon toga u glavnom kodu se iz *libLeginRepalus* biblioteke poziva *quat2jv* (slika 44) . Navedena funkcija služi za pretvaranje kvaterniona iz originalnog RAPID koda u jedinični vektor koji predstavlja orijentaciju alata u smjeru razmatrane osi [18]. U ovom dijelu koda odabrana je Z os.

```
86  def quat2jv(q1, q2, q3, q4, chsVec):
87      """
88      q1 -> w
89      q2 -> x
90      q3 -> y
91      q4 -> z
92      chsVec -> "x", "y", "z"
93      """
94      if chsVec == "x":
95          i = 1-2*pow(q3,2)-2*pow(q4,2)
96          j = 2*q2*q3+2*q1*q4
97          k = 2*q2*q4-2*q1*q3
98      elif chsVec == "y":
99          i = 2*q2*q3-2*q1*q4
100         j = 1-2*pow(q2,2)-2*pow(q4,2)
101         k = 2*q3*q4+2*q1*q2
102     elif chsVec == "z":
103         i = 2*q2*q4+2*q1*q3
104         j = 2*q3*q4-2*q1*q2
105         k = 1-2*pow(q2,2)-2*pow(q3,2)
106     else:
107         i = float("NaN")
108         j = float("NaN")
109         k = float("NaN")
110
111     return(i,j,k)
112
```

Slika 44 *quat2jv*

ABB-ovi roboti koriste kvaternione za definiranje orijentacije vrha alata. Kvaternion je broj koji ima tri imaginarne i jednu realnu komponentu. On predstavlja matematičku veličinu koja se koristi za računanje i opisivanje prostornih transformacija. Prednost kvaterniona naspram uobičajeno korištenih Eulerovih kutova je ta što se upotrebom kvaterniona prilikom izračuna izbjegava singularnost. Također, kvaternioni su jednoznačno definirani dok se za Eulerove kutove koriste razne konvencije i ako se koristi kriva konvencija dobije se drukčiji rezultat. Nedostatak kvaterniona je što nisu intuitivni.

Projekcije jediničnog vektora množe se s -1 jer Z os alata gleda iz alata prema obratku, a u *GOM Inspect Pro*-u normala površine gleda iz obratka prema alatu. Zatim se dobivene vrijednosti (koordinate točaka i projekcije jediničnih vektora) koriste unutar naredbe *gom.script.primitive.create_surface_point*, kojom *GOM Inspect Pro* kreira točku na obratku zajedno s pripadajućim vektorom normale.

6.2.3. Automatizirano stvaranje točaka na površini uzorka

Kako bi se automatizirao postupak stvaranja točaka s pripadajućom normalom na uzorku (*Actual*-u), najprije se uključuje snimanje naredbi. Nakon toga svim točkama na idealnom uzorku dodjeljuje se *„Measuring Principle: Intersection With Mesh“*. Slika 45 prikazuje naredbu dobivenu snimanjem makro funkcije.

```
import gom
MCAD_ELEMENT=gom.script.inspection.measure_by_intersection_with_mesh (
    check_plausibility=True,
    elements=gom.ElementSelection ({'category': ['key', 'elements', 'part', gom.app.project.parts['Part'], 'explorer_category', 'nominal', 'object_family', 'geometrical_element', 'type', 'inspection_surface_point']}),
    properties=gom.Binary {'eAHFmF1sW2cZx38tXRfKupJtjHVCYKUBwjYnTppmrdvRrh+jg3aLtrabJsA6tk8SU3/VdppmqPQULhj5pIkLGOIGJD6lSkW9QAMEghskhFC4QMA0tAsEEhcUbdoFm5DWov/znjc+ju3GBSEcJX19zvs+n//n/zznnKhV0qd2jZ/j6JGHDx95ghONoNRqZ
```

Slika 45 Skripta za automatizirano stvaranje točaka na površini uzorka

6.2.4. Korekcija putanje alata

Iz postavljenih točaka na CAD modelu deformiranog uzorka moguće je dobiti vrijednosti odstupanja deformirane ploče. Kada su dobiveni podaci o odstupanjima deformirane ploče, krenulo se u izradu koda koji korigira putanju alata. Slika 46 prikazuje dio koda koji iščitava koordinate točaka na referentnom (xNom, yNom, zNom) odnosno deformiranom (xAct, yAct, zAct) uzorku te duljine projekcija vektora normale (jediničan vektor) na referentnom (iNom, jNom i kNom) odnosno deformiranom (iAct, jAct, kAct) uzorku. Te se vrijednosti ispisuju radi kontrole i pohranjuju se u podatak tipa rječnik (eng. *dictionary*).

Slika 47 prikazuje dio koda koji iščitava linije iz RAPID koda koje se nalaze između ″! Pocetak″ i ″! Kraj″ te ih zakomentira kako bi u korigiranom RAPID kodu ostao zapis originalne linije (Slika 54). Nakon toga, zbog kontrole, ta se linija ispisuje na konzolu.

Na slici 48 dan je dio koda koji iz linije u RAPID kodu izdvaja podatke o kvaternionima (slika 43) te ih ispisuje na konzolu. Iz biblioteke se ponovno poziva funkcija *quat2jv* (slika 44). Sada je odabrana X os koja će se kasnije koristiti kao dodatni vektor za definiciju orijentacije. Iz *numpy* biblioteke poziva se funkcija *np.zeros* koja se koristi za stvaranje matrice određenih dimenzija i tipa podataka, u ovom slučaju vektora (1x3). Podaci koji se upisuju u ovaj vektor dobiveni su *quat2jv* funkcijom.

```python
import gom
import numpy as np
import libLeginRepalust
FILE="C:\DIPLOMSKI\kod\RobotStudio_RAPID_brusenje.mod"
OUTPUT_FILE="C:\DIPLOMSKI\kod\RobotStudio_RAPID_brusenje_updated.mod"
REPORT_FILE="C:\DIPLOMSKI\korekcija_putanje.csv"
STR_POCETAK="! Pocetak"
STR_KRAJ="! Kraj"

lstPtData=[]
ptNo=1
ptName = "BLANK"
while True:
    try:

        ptName = "Point %s" % str(ptNo)
        print(ptName)

        xNom = gom.app.project.inspection[ptName].center_coordinate.x
        yNom = gom.app.project.inspection[ptName].center_coordinate.y
        zNom = gom.app.project.inspection[ptName].center_coordinate.z

        iNom = gom.app.project.inspection[ptName].normal.x
        jNom = gom.app.project.inspection[ptName].normal.y
        kNom = gom.app.project.inspection[ptName].normal.z

        xAct = gom.app.project.actual_elements[ptName].center_coordinate.x
        yAct = gom.app.project.actual_elements[ptName].center_coordinate.y
        zAct = gom.app.project.actual_elements[ptName].center_coordinate.z

        iAct = gom.app.project.actual_elements[ptName].normal.x
        jAct = gom.app.project.actual_elements[ptName].normal.y
        kAct = gom.app.project.actual_elements[ptName].normal.z

        print("-"*50)

        print("Processing point: %d (%s)" % (ptNo, ptName))
        print("Nominal coordinates:", xNom, yNom, zNom)
        print("Nominal normal:", iNom, jNom, kNom)
        print("Actual coordinates:", xAct, yAct, zAct, )
        print("Actual normal:", iAct, jAct, kAct)
        print("\n")


        dictPtData={'id': ptNo,
                    'Xnom': xNom, 'Xact': xAct,
                    'Ynom': yNom, 'Yact': yAct,
                    'Znom': zNom, 'Zact': zAct,
                    'iNom': iNom, 'iAct': iAct,
                    'jNom': jNom, 'jAct': jAct,
                    'kNom': kNom, 'kAct': kAct,
                    }
        lstPtData.append(dictPtData)


    except Exception as ex:
        print("Zadnja tocka/kut: %d (%s)" %(ptNo, ptName))
        print(str(ex))
        break
    else:
        ptNo += 1

for pt in lstPtData:
    print(pt)
```

Slika 46 Dio koda za iščitavanje koordinata točaka i projekcija vektora normala te njihovo spremanje u tablicu

```python
updateData = False
numLineToUpdate=1
with open(FILE, 'r') as inFile, open(OUTPUT_FILE, 'w') as outFile:
    for line in inFile:
        if line.strip() == STR_POCETAK:
            updateData = True
            outFile.write(line)
            continue
        elif line.strip() == STR_KRAJ:
            updateData = False
            outFile.write(line)
            continue

        if updateData:

            _leading_spaces=len(line) - len(line.lstrip())
            outFile.write("%s ! ORIGINAL LINE: %s" %("\t" * _leading_spaces, line.lstrip()))

            print("ORIGINAL LINE:", line.strip())
```

Slika 47 Dio koda koji iščitava linije iz originalnog RAPID koda

```python
_tocke = libLeginRepalust.parse_rapid_points(line)
_orig_quat=_tocke[1]
print("Original quaterniuons:", _orig_quat)
_orig_norm_x = libLeginRepalust.quat2jv(_orig_quat[0], _orig_quat[1], _orig_quat[2], _orig_quat[3], "x")

xVecNormNom = np.zeros((3), dtype=float)
xVecNormNom[0] = _orig_norm_x[0]
xVecNormNom[1] = _orig_norm_x[1]
xVecNormNom[2] = _orig_norm_x[2]
```

Slika 48 Dodatni vektor za definiciju orijentacije

Prve tri linije koda na slici 49 služe kako bi se došlo do podataka o koordinatama točaka na idealnom modelu uzorka. U prvoj liniji koda dodaje se ″+ 2″ zato što program u liniji traži ″[[″ te kada nađe te simbole on i njih uključi u rezultat pretraživanja. Iz tog razloga potrebno je pomaknuti granicu za dva mjesta kako bi se izdvojili samo podaci o koordinatama točaka. Dobiveni se podaci zatim ispisuju. Iduće tri linije služe kako bi se došlo do podataka o kvaternionima te se i oni ispisuju.

Iz rječnika se iščitavaju i ispisuju podaci (dobiveni iz GOM *Inspect Pro*-a) o odgovarajućoj točki. Ponovno se naredbom *np.zeros* stvara vektor (1x3) koji definira Z os koordinatnog sustava vrha alata čiji smjer odgovara vektoru normale iščitane iz GOM *Inspect Pro*-a, ali je suprotne orijentacije. Iz tog razloga vektoru je potrebno promijeniti orijentaciju množenjem njegovih komponenti s -1.0.

```
_srch_tocke_start = line.find("[[") + 2
_srch_tocke_end = line.find("]")
_srch_koordinate_tocaka_str=line[_srch_tocke_start:_srch_tocke_end]
print("PT   ==> START:", _srch_tocke_start, "END:", _srch_tocke_end, "DATA:",_srch_koordinate_tocaka_str)


_srch_quart_start=line[_srch_tocke_end:].find("[") + _srch_tocke_end + 1
_srch_quart_end = line[_srch_quart_start:].find("]") + _srch_quart_start
_srch_karternioni_str= line[_srch_quart_start:_srch_quart_end]
print("ANGL ==> START:", _srch_quart_start, "END:", _srch_quart_end, "DATA:", _srch_karternioni_str)


_matchingPt = [i for i in lstPtData if i['id'] == numLineToUpdate][0]

print("UPDATED POINT DATA:", _matchingPt)

zVecNormAct = np.zeros((3), dtype=float)
zVecNormAct[0] = _matchingPt["iAct"] * -1.0
zVecNormAct[1] = _matchingPt["jAct"] * -1.0
zVecNormAct[2] = _matchingPt["kAct"] * -1.0
```

Slika 49 Zapis Z vektora dobiven pomoću podataka iz GOM Inspect Pro-a

Kada su definirani jedinični vektor koji predstavlja Z osi koordinatnog sustava vrha alata i dodatni X vektor za određivanje rotacije koordinatnog sustava oko Z-osi, u glavnom kodu (slika 50) poziva se *jv2rotmtx* (slika 51). *jv2rotmtx* najprije pomoću vektorskog produkta (*np.cross*) jediničnog vektora Z osi i dodatnog vektora za definiciju orijentacije (stara X os) izračunava Y vektor a zatim vektorskim produktom vektora Z-osi i upravo dobivenim vektorom Y-osi dobiva se novi X vektor. Nakon toga X i Y vektor se normaliziraju kako bi se dobili jedinični vektori, dok je vektor Z-osi već očitan iz GOM *Inspect Pro*-a kao jedinični vektor. Podaci o projekcijama jediničnih vektora upisuju se u matrice rotacije. Kada se popuni matrica rotacije, u glavnom kodu se poziva *rotmtx2quat* (slika 52). *rotmtx2quat* pretvara matricu rotacije u kvaternione [21]. Nakon toga ispisuju se dobiveni podaci. Završni dio koda dan je na slici 53. Ovaj dio koda zapisuje nove podatke o koordinatama točaka i kvaternionima u novi RAPID kod (slika 54).

```
rotMtxAct = libLeginRepalust.jv2rotmtx(zVecNormAct, xVecNormNom)

_act_quat = libLeginRepalust.rotmtx2quat(rotMtxAct)

print("ROT Data:")
print("UV xNOM:", xVecNormNom)
print("UV zACT:", zVecNormAct)
print("RM:\n", rotMtxAct)
print("Q ORIG:", _orig_quat)
print("Q ACT:", _act_quat)
```

Slika 50 Dio glavnog koda za izračun korigiranih vrijednosti kvaterniona

```
def jv2rotmtx(vecNorm2Surf, vecOri):
    """
    vecNorm -> vektor normale na povrsinu (aktualna z-os)
    vecOri  -> dodatni vektor za definiciju orijentacije (stara x-os)
    """

    zVec = vecNorm2Surf
    yVec = np.cross(zVec, vecOri)
    xVec = np.cross(yVec, zVec)

    xVecNorm = xVec / np.sqrt(pow(xVec[0], 2)+pow(xVec[1], 2)+pow(xVec[2], 2))
    yVecNorm = yVec / np.sqrt(pow(yVec[0], 2)+pow(yVec[1], 2)+pow(yVec[2], 2))
    zVecNorm = zVec / np.sqrt(pow(zVec[0], 2)+pow(zVec[1], 2)+pow(zVec[2], 2))

    rotMtx = np.matrix([[xVecNorm[0], yVecNorm[0], zVecNorm[0]],
                                      [xVecNorm[1], yVecNorm[1], zVecNorm[1]],
                                [xVecNorm[2], yVecNorm[2], zVecNorm[2]]])
```

Slika 51 kod funkcije *jv2rotmtx*

```
def rotmtx2quat(rotMtx):

    tr = rotMtx[0, 0] + rotMtx[1, 1] + rotMtx[2, 2]

    if tr > 0:
        s = np.sqrt(tr+1.0) * 2 # S=4 * qw
        q1 = 0.25 * s
        q2 = (rotMtx[2, 1] - rotMtx[1, 2]) / s
        q3 = (rotMtx[0, 2] - rotMtx[2, 0]) / s
        q4 = (rotMtx[1, 0] - rotMtx[0, 1]) / s
    elif (rotMtx[0, 0] > rotMtx[1, 1]) & (rotMtx[0, 0] > rotMtx[2, 2]):
        s = np.sqrt(1.0 + rotMtx[0, 0] - rotMtx[1, 1] - rotMtx[2, 2]) * 2 # s=4 * qx
        q1 = (rotMtx[2, 1] - rotMtx[1, 2]) / s
        q2 = 0.25 * s
        q3 = (rotMtx[0, 1] + rotMtx[1, 0]) / s
        q4 = (rotMtx[0, 2] + rotMtx[2, 0]) / s
    elif rotMtx[1, 1] > rotMtx[2, 2]:
        s = np.sqrt(1.0 + rotMtx[1, 1] - rotMtx[0, 0] - rotMtx[2, 2]) * 2 # s=4 * qy
        q1 = (rotMtx[0, 2] - rotMtx[2, 0]) / s
        q2 = (rotMtx[0, 1] + rotMtx[2, 0]) / s
        q3 = 0.25 * s
        q4 = (rotMtx[1, 2] + rotMtx[2, 1]) / s
    else:
        s = np.sqrt(1.0 + rotMtx[2, 2] - rotMtx[0, 0] - rotMtx[1, 1]) * 2 # s=4 * qz
        q1 = (rotMtx[1, 0] - rotMtx[0, 1]) / s
        q2 = (rotMtx[0, 2] + rotMtx[2, 0]) / s
        q3 = (rotMtx[1, 2] + rotMtx[2, 1]) / s
        q4 = 0.25 * s

    quatOut = [q1, q2, q3, q4]

    return quatOut
```

Slika 52 kod funkcije *rotmtx2quat*

```
162             _repl_koordinate_tocaka_str= "%.3f,%.3f,%.3f" % (_matchingPt["Xact"], _matchingPt["Yact"], _matchingPt["Zact"])
163             print("REPL PT DATA:", _srch_koordinate_tocaka_str, "->", _repl_koordinate_tocaka_str)
164
165             _repl_karternioni_str= "%.8f,%.8f,%.8f,%.8f" % (_act_quat[0], _act_quat[1], _act_quat[2], _act_quat[3])
166             print("REPL ANGLE DATA:", _srch_karternioni_str, "->", _repl_karternioni_str)
167
168             repl_line = line.replace(_srch_koordinate_tocaka_str, _repl_koordinate_tocaka_str)
169
170
171             repl_line = repl_line.replace(_srch_karternioni_str, _repl_karternioni_str)
172
173             print("UPDATED LINE:",repl_line.lstrip())
174
175             outFile.write(repl_line)
176
177             print("\n")
178
179             numLineToUpdate +=1
180         else:
181             outFile.write(line)
182
183 print ("DONE")
```

Slika 53 ažuriranje podataka o koordinatama točaka i kvaterniona u RAPID kodu

```
! Pocetak
 ! ORIGINAL LINE: MoveL [[80.000,105.000,0.000],[0.00000000,0.00000000,1.00000000,0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
MoveL [[80.000,105.000,1.002],[-0.00000000,0.00000000,0.99955455,0.02984466],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
 ! ORIGINAL LINE: MoveL [[355.000,105.000,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
MoveL [[355.000,105.000,1.002],[-0.00000000,0.00000000,0.99955455,0.02984466],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
 ! ORIGINAL LINE: MoveL [[362.028,105.381,0.000],[0.00000000,-0.00000000,1.00000000,0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
MoveL [[362.028,105.381,1.025],[-0.00000000,0.00000000,0.99955455,0.02984466],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
 ! ORIGINAL LINE: MoveL [[368.973,106.520,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
MoveL [[368.973,106.520,1.093],[-0.00000000,0.00000000,0.99955455,0.02984466],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
 ! ORIGINAL LINE: MoveL [[375.755,108.403,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
MoveL [[375.755,108.403,1.205],[-0.00000000,0.00000000,0.99955455,0.02984466],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
 ! ORIGINAL LINE: MoveL [[382.293,111.008,0.000],[0.00000000,-0.00000000,1.00000000,0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
MoveL [[382.293,111.008,1.371],[-0.00000000,0.00000000,0.99937787,0.03526872],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
 ! ORIGINAL LINE: MoveL [[388.511,114.304,0.000],[0.00000000,-0.00000000,1.00000000,0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
MoveL [[388.511,114.304,1.604],[-0.00000000,0.00000000,0.99937787,0.03526872],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
 ! ORIGINAL LINE: MoveL [[394.336,118.254,0.000],[0.00000000,0.00000000,1.00000000,0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
MoveL [[394.336,118.254,1.899],[-0.00000000,0.00000000,0.99917172,0.04069251],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
 ! ORIGINAL LINE: MoveL [[399.700,122.810,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
MoveL [[399.700,122.810,2.271],[-0.00000000,0.00000000,0.99917172,0.04069251],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
 ! ORIGINAL LINE: MoveL [[404.541,127.920,0.000],[0.00000000,-0.00000000,1.00000000,0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
MoveL [[404.541,127.920,2.737],[-0.00000000,0.00000000,0.99893619,0.04611389],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
 ! ORIGINAL LINE: MoveL [[408.800,133.523,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
MoveL [[408.800,133.523,3.293],[-0.00000000,0.00000000,0.99867121,0.05153463],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
 ! ORIGINAL LINE: MoveL [[412.428,139.553,0.000],[0.00000000,0.00000000,1.00000000,0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
MoveL [[412.428,139.553,3.949],[-0.00000000,0.00000000,0.99837685,0.05695325],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
 ! ORIGINAL LINE: MoveL [[415.383,145.941,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
MoveL [[415.383,145.941,4.708],[-0.00000000,0.00000000,0.99805302,0.06237116],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
 ! ORIGINAL LINE: MoveL [[417.631,152.611,0.000],[0.00000000,0.00000000,1.00000000,0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
MoveL [[417.631,152.611,5.549],[-0.00000000,0.00000000,0.99801049,0.06304814],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
 ! ORIGINAL LINE: MoveL [[419.144,159.484,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
MoveL [[419.144,159.484,6.419],[-0.00000000,0.00000000,0.99825884,0.05898551],[-1,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],v1000,z1,T1_IRB6660_FC \WObj:=NT_OBRADE;
```

Slika 54 Izgled korigiranog RAPID koda (dio koji opisuje gibanje alata)

7. SIMULACIJA KORIGIRANE PUTANJE ALATA

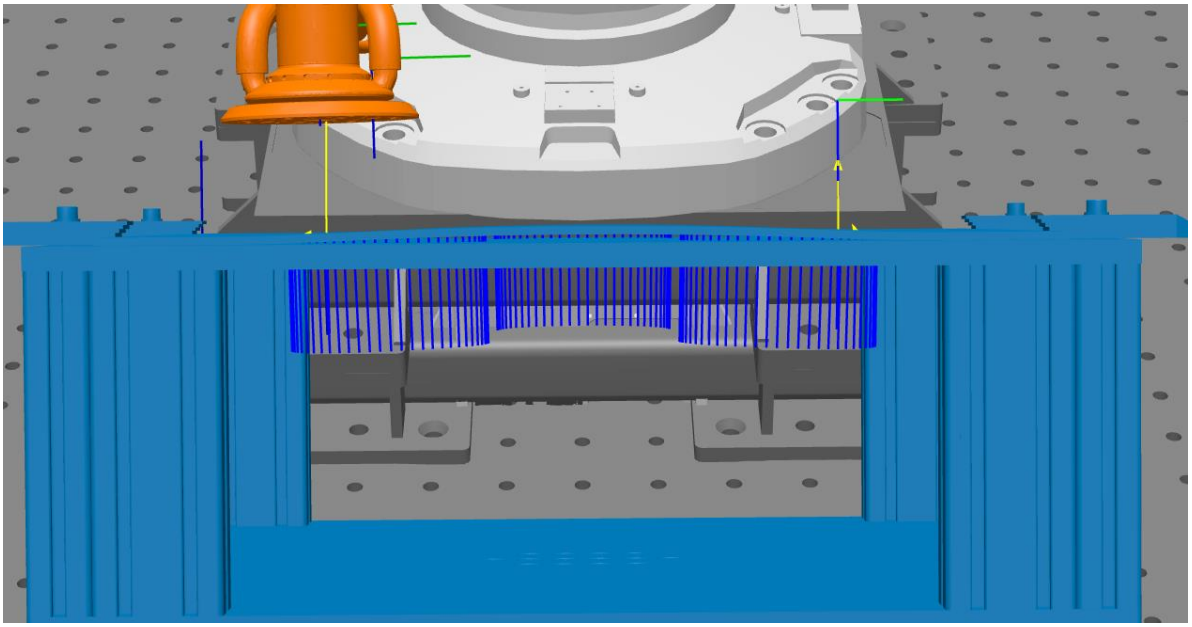
Kada je napravljen kod za korekciju putanje alata, u *GOM Inspect Pro-u* učitani su uzorci s različitim stupnjevima odstupanja oblika (5 mm i 15 mm). Pomoću koda korigirala se putanja alata te se ona simulirala u *RobotStudio*-u.

7.1. Uzorak s maksimalnim odstupanjem od 5 mm

Prvi uzorak koji se učitao u *GOM Inspect Pro* bio je onaj kod kojega najveća deformacija ispitnog uzorka iznosi 5 mm. Slika 55 prikazuje položaj točaka na referentnom (plavo) i deformiranom (zeleno) ispitnom uzorku kao i pripadajuće vektore normala.

Slika 55 Položaj točaka na referentnoj (plavo) i deformiranoj (zeleno) ploči te pripadajući vektori normala

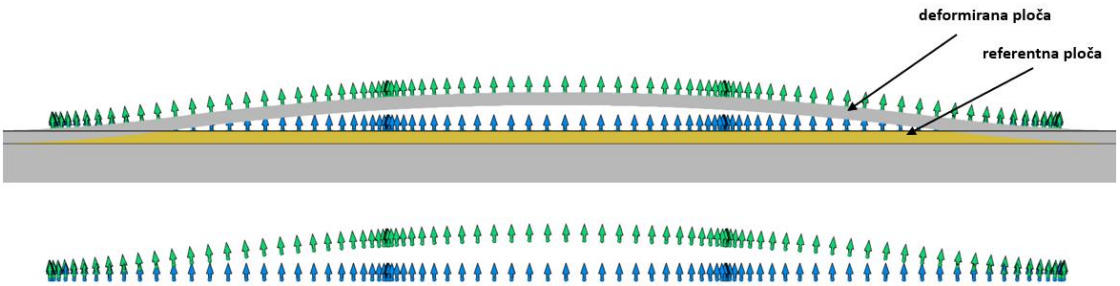
Kada je generirana korigirana putanja alata, ona je učitana u *RobotStudio* kako bi se verificiralo gibanje alata po korigiranoj putanji (slika 56). Na slici 56 može se vidjeti da su točke korigirane putanje alata zamaknute u smjeru Z osi (na slici plavo obojena os) što znači da dio koda koji korigira položaj točaka radi. Također se sa slike se može uočiti da je Z os alata nagnuta u nekoliko točaka što znači da radi i dio koda koji korigira orijentaciju, odnosno nagib alata.



Slika 56 Korigirana putanja alata prikazana u RobotStudio-u

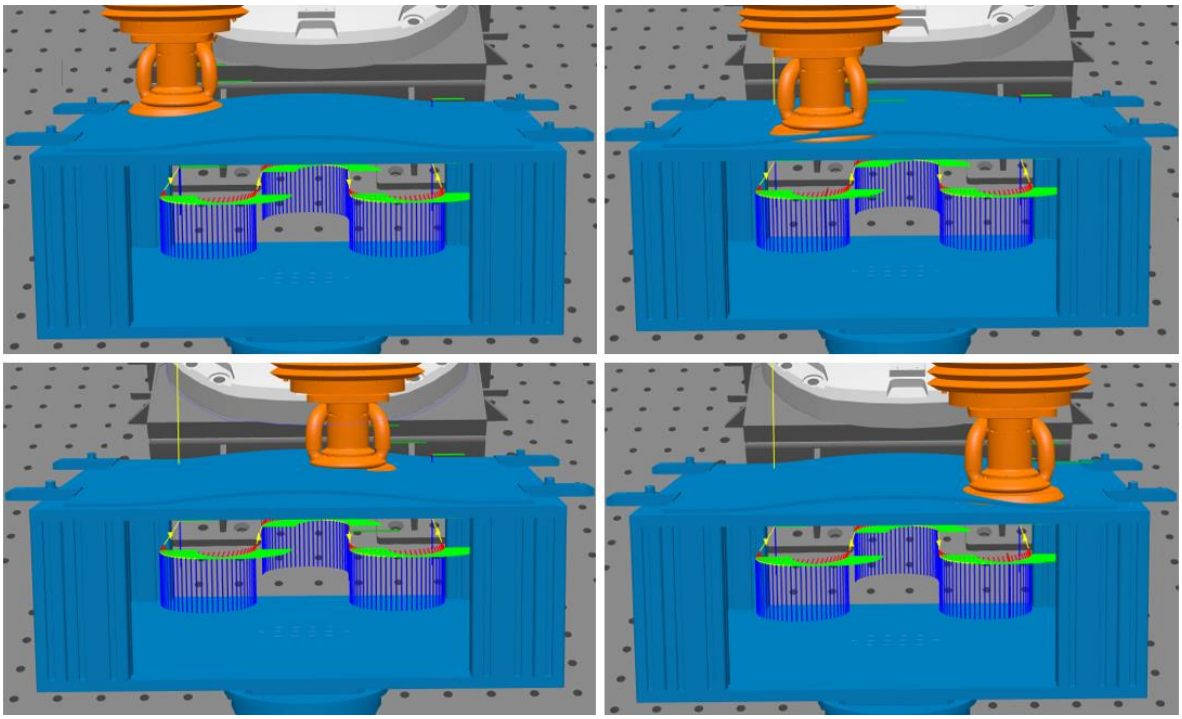
7.2. Uzorak s maksimalnim odstupanjem od 15 mm

Kako bi se bolje vidjela razlika između originalne i korigirane putanje alata, izrađen je uzorak kojemu najveće odstupanje deformirane ploče iznosi 15 mm. Slika 57 prikazuje uzorak učitan u *GOM Inspect Pro*-u na kojem se nalaze korigirane točke putanje alata s pripadajućim vektorima normala (obojene u plavo).



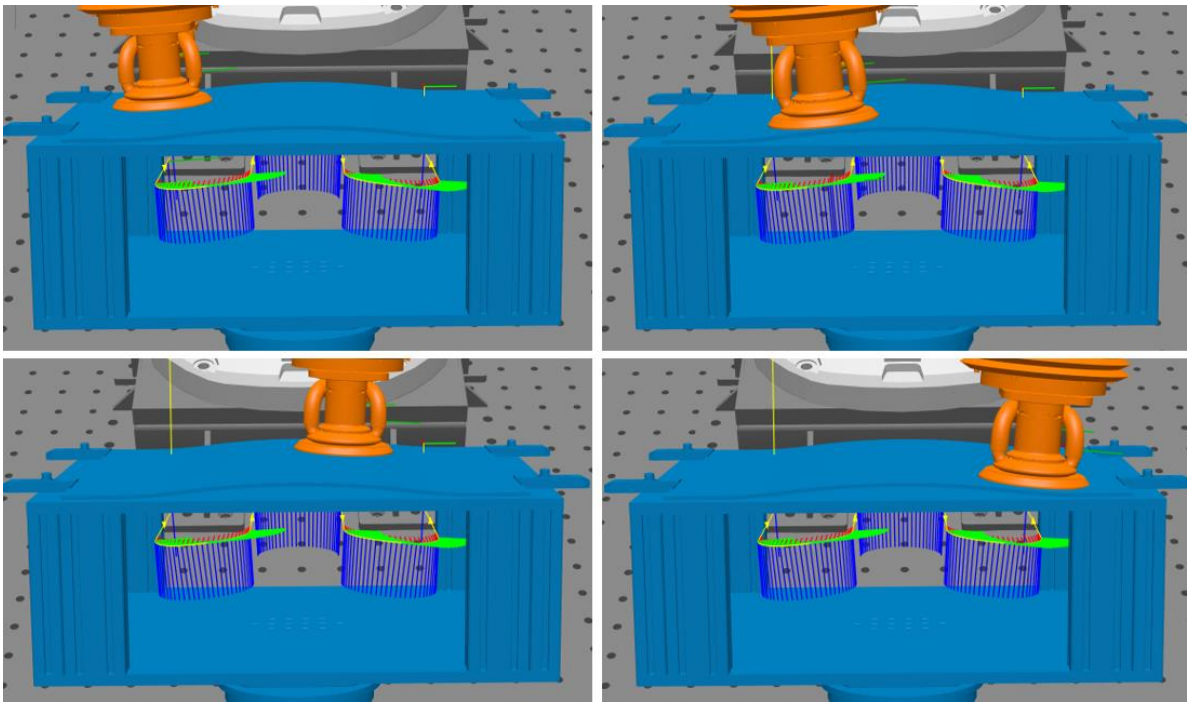
Slika 57 Položaj točaka i pripadajući vektori normala na referentnoj (plavo) i deformiranoj (zeleno) ploči

Slika 58 prikazuje nekorigiranu putanju alata u *RobotStudio*-u. Iz slike se vidi da u području deformacije ploče dolazi do prodora alata u ispitni uzorak.



Slika 58 Nekorigirana putanja alata u RobotStudio-u

Slika 59 prikazuje korigiranu putanju alata. Iz putanje se mogu jasno vidjeti nagibi alata (plave linije) u svakoj točki. U slučaju korigirane putanje, alat ne ulazi u prodor s ispitnim uzorkom nego prati njegov oblik. Pokretanjem simulacije verificirala se ispravnost korigirane putanje.



Slika 59 Korigirana putanja alata u RobotStudio-u

8. ZAKLJUČAK

Uređaji za beskontaktno 3D skeniranje u proizvodnji se najčešće koriste na kraju proizvodnog procesa u svrhu kontrole kvalitete gotovog proizvoda. U ovome radu ta se tehnologija koristila prije samog procesa brušenja u svrhu korekcije putanje alata prema odstupanjima oblika. Budući da nije bilo moguće izraditi stvarno deformirane ispitne uzorke te ih skenirati, ispitni su se uzorci napravili u programu Catia V5 V5R21. Navedeni su se ispitni uzorci koristili kao simulirani rezultati digitalizacije. U softveru Catia V5 V5R21 napravljena je i putanja alata na temelju koje se pomoću *RoboDK*-a generirao početni RAPID kod. Zatim se u *GOM Inspect Pro*-u formirao mjerni postav koji se sastoji od CAD modela stezne naprave, idealnog ispitnog uzorka i simuliranih rezultata digitalizacije. U sklopu tog mjernog postava napravljena su dva koda u programskom jeziku *Python*. Prvi kod služi za čitanje točaka putanje alata iz originalnog RAPID koda te postavljanje tih točaka s pripadajućim vektorima normala na idealni ispitni uzorak. Kada se te točke preslikaju na deformirani ispitni uzorak pokreće se drugi kod koji na temelju analize odstupanja projiciranih točaka i normala na deformiranom ispitnom uzorku korigira putanju alata. Simulacijom korigirane putanje alata u *RobotStudio*-u verificirala se njena ispravnost.

Budući da je korigirana putanja alata isprobana samo simulacijom u *RobotStudio*-u, idući korak bio bi izvesti te pokuse u stvarnosti. Također, na stvarnim uzorcima bilo bi moguće napraviti i analizu površine te utvrditi kako korigirana i nekorigirana putanja alata utječu na njena svojstva. Isto tako moglo bi se istražiti na koji način korigirana putanja alata utječe na vijek trajanja alata.

LITERATURA

[1] https://www.3dsourced.com/guides/what-is-3d-scanning-guide/ (03.10.2023.)

[2] https://www.aniwaa.com/guide/3d-scanners/3d-scanning-technologies-and-the-3d-scanning-process/ (03.10.2023.)

[3] https://en.wikipedia.org/wiki/Structured-light_3D_scanner (04.10.2023.)

[4] https://3space.com/3d-scanning-technology-comparison/ (04.10.2023.)

[5] https://www.3dnatives.com/en/laser-3d-scanner-vs-structured-light-3d-scanner-080820194/# (04.10.2023.)

[6] https://www.artec3d.com/learning-center/laser-3d-scanning (04.10.2023.)

[7] https://www.sculpteo.com/blog/2015/11/11/scanning-for-3d-printing-using-photogrammetry/ (04.10.2023.)

[8] https://search.abb.com/library/Download.aspx?DocumentID=ROB0053EN_F&LanguageCode=en&DocumentPartId=&Action=Launch (05.10.2023.)

[9] https://new.abb.com/products/3HAC020536-013/irb-6660?HR (05.10.2023.)

[10] https://library.e.abb.com/public/91504b95a1734759a3e5b3eb5fc20560/3HAC028207%20PS%20IRB%206660-en.pdf (06.10.2023.)

[11] https://new.abb.com/products/robotics/application-equipment-and-accessories/workpiece-positioners/irbp-a (06.10.2023.)

[12] https://new.abb.com/products/robotics/robots/articulated-robots/irb-4600 (06.10.2023.)

[13] Požgaj,MD. *Integracija beskontaktnog uređaja za digitalizaciju oblika i mjerenje topologije površine*. Zagreb: Fakultet strojarstva i brodogradnje; 2023.

[14] https://www.fadi-ims.com/MS1-ATOS-5X.html (06.10.2023.)

[15] https://forums.robotstudio.com/discussion/499/system-module (14.11.2023.)

[16] https://en.wikipedia.org/wiki/NumPy (19.11.2023.)

[17] https://www.zeiss.nl/metrologie/producten/software/inspect/python-interface.html (19.11.2023.)

[18] https://www.euclideanspace.com/maths/geometry/rotations/conversions/matrixToQuaternion/index.htm (19.11.2023.)

PRILOZI

I. CD-R disk